Budapest University of Technology and Economics
Department of Telecommunications and Media Informatics

Optimal Resource Pooling over Legacy Equal-Split Load Balancing Schemes

Krisztián Németh

Ph.D. Dissertation

Supervised by

Gábor Rétvári, Ph.D.

Senior Research Fellow

Budapest, Hungary
2018

Abstract

Communication networks are increasingly, and by now almost completely based on the Internet Protocol. These networks are continuously being enhanced, posing challenges to protocol designers, device vendors and network operators. One of the problems they have to face is related to traffic splitting.

Splitting traffic flows to different data paths can serve as the basis for load balancing at several places of an end-to-end connection. Unfortunately, by allowing only equal division amongst the parallel resources, existing technologies often cannot realize the optimal traffic splitting, which can have serious negative consequences on the network performance.

In this dissertation I present a flexible and efficient traffic splitting method that is incrementally deployable and fully compatible with practically all existing protocols and data planes. My proposal is called Virtual Resource Allocation (VRA) and, as the name implies, it is based on setting up virtual resources alongside existing ones, as for example virtual links parallel to the physical links. This way one can trick the legacy equal traffic splitting technology into realizing the required non-equal division over the physical media.

In my dissertation I propose several VRA schemes, give theoretical bounds on their performance, and also show that the full-fledged VRA problem is NP-complete in general. I also provide algorithms, both optimal methods and quick heuristics, that solve the different VRA problems. I use IP Traffic Engineering with OSPF routing as an example application of the concept. My simulations show that VRA has huge practical potential as it allows approaching an ideal traffic split using only a very limited set of virtual resources.

Kivonat

A kommunikációs hálózatok napjainkra szinte teljesen az Internet Protokollt használják. E hálózatokat folyamatosan fejlesztik, amely során a protokollok tervezőinek, a hálózati eszközök gyártóinak és a szolgáltatóknak egyre újabb feladatokat kell megoldaniuk. Az egyik ilyen probléma a hálózati forgalom szétosztásával kapcsolatos.

A forgalom különböző adatutak közötti megosztása a terheléskiegyenlítés alapjául szolgálhat egy végponttól végpontig tartó kapcsolat különböző részein is. A jelenlegi technológiák azonban gyakran nem képesek az optimális szétosztás megvalósítására, mivel csak egyenletesen tudják a forgalmat a párhuzamos erőforrások között szétosztani, ez pedig nagyon hátrányosan befolyásolhatja a hálózatok teljesítményét.

Disszertációmban bemutatok egy rugalmas és hatékony forgalomszétosztó módszert, amely fokozatosan is bevezethető és teljesen kompatibilis gyakorlatilag az összes meglévő protokollal és csomagtovábbítási rendszerrel. Javaslatomat virtuális erőforrásfoglalásnak (Virtual Resource Allocation) neveztem el, és, ahogy a neve is mutatja, a meglévő valódi erőforrások melletti virtuális erőforrások létrehozásán alapul: például virtuális útvonalakat hozhatunk létre a valódiakkal párhuzamosan. Ezzel a megoldással rábírhatjuk a meglévő, egyenletesen szétosztó rendszereket arra, hogy a megkívánt, nem egyenlő arányban osszák el a forgalmat a valódi eszközök között.

Értekezésemben több különböző virtuális erőforrás-foglalási sémát javaslok, elméleti korlátokat adok a teljesítményükre, és azt is megmutatom, hogy a teljes feladat általánosságban NP-teljes. A problémákhoz optimális megoldási algoritmusokat valamint gyors heurisztikákat is adok. A javaslatom bemutatására példaként az OSPF útvonalválasztással megvalósított IP forgalomelvezetés (Traffic Engineering) témakörét választottam. A bemutatott szimulációm eredményei alátámasztják a javasolt módszer létjogosultságát, megmutatva, hogy az ideális osztásarány nagyon jól megközelíthető már kisszámú virtuális erőforrás alkalmazásával is.

Acknowledgements

It was a long and winding road that led me here. It would have been very hard, if not impossible, to walk this road alone. Fortunately several people accompanied me along the way. Some precious ones are not with us anymore, sadly, including my father, and some have been born since then. I am grateful to everyone who helped and encouraged me to finish what I have started back in 1998.

First of all, I would like to thank my ex-girlfriend Viki (who is now my wife) for her invaluable, continuous and unconditional support. Thank you so much, Viki. I am also thankful to our children, Eszter and Roland, without whom, as the joke says, this dissertation could have been finished many years earlier. While this is certainly not true, without them this place would be a lot lonelier and much less cheerful. My parents have helped me in several ways, and again, all I can do is thank them for all.

In the University I had several leaders, colleagues as well as students who helped me in various ways throughout the years, for which I am grateful. Having Gábor Rétvári become my scientific advisor was definitely a breakthrough in my progress towards the degree. With his professional lead I was able to carry out this work within a few years. I am also indebted to Attila Kőrösi who helped me with my math-related questions. His quick way of thinking has always fascinated me. I thank my previous advisors, István Cselényi, Krzysztof Szarkowicz and Sándor Székely for supporting me, which also helped me staying at the University for these mostly happy years. Thank you, gentlemen.

It is impossible to name all those co-workers who helped and encouraged me when I was writing this dissertation. József Bíró, Gyula Csopaki, Edit Halász, and Tamás Henk are definitely on this (alphabetical) list, but it is really not possible to include everybody here as there are so many of them. Please rest assured that if you supported me, I noticed and appreciate it. I thank you.

Last but not least, for the simulation evaluation I used “superman”, the High Performance Computing Cluster of the University. With the help of this excellent machine my simulations have been completed within weeks, instead of years. This infrastructure was supported by the grant TÁMOP-4.2.2.B-10/1–2010-0009.

Köszönetnyilvánítás

Hosszú utat jártam be, mire idáig eljutottam. Ezen az úton végigmenni nehéz, sőt minden bizonnyal lehetetlen lett volna egyedül. Szerencsére azonban sokan velem tartottak menet közben. Néhányan már nem lehetnek köztünk – sajnos édesapám sem –, néhányan pedig időközben születtek. Hálás vagyok mindazoknak, akik biztattak és segítettek, hogy befejezzem, amit még 1998-ban elkezdtem.

Legelőször szeretném megköszönni volt barátnőmnek, Vikinek (akit legtöbben bizonyára úgy ismernek, mint a feleségem) a felbecsülhetetlen értékű folyamatos és odaadó támogatását. Viki, nagyon köszönöm! Szintén hálás vagyok gyerkőceinknek, Eszternek és Rolandnak, akik nélkül – ahogy mondani szokás – ez a disszertáció évekkel korábban elkészülhetett volna. Ez természetesen csak vicc, de az biztos, hogy nélkülük ez a hely sokkal magányosabb és szomorkásabb lenne. A szüleim rengeteget segítettek, minden tekintetben, én pedig továbbra sem tehetek egyebet, mint hogy megköszönöm nekik.

Az Egyetemen sok főnököm, kollégám és hallgatóm is segített mind személyes, mind szakmai tanácsokkal az évek során, amelyekért nagyon hálás vagyok. Az, hogy Rétvári Gábor a konzulensem lett, valódi áttörést jelentett a munkámban. Az ő kiváló témavezetői munkája segítségével pár év alatt el tudtam készíteni azt, amit e disszertációban leírtam. Szintén hálával tartozom Kőrösi Attilának, aki segített megválaszolni a matematikával kapcsolatos kérdéseimet. Az elképesztő gyors gondolkodása mindig lenyűgözött. A korábbi témavezetőim, Cselényi István, Szarkowicz Krzysztof és Székely Sándor szintén sokat tettek értem, ami ahhoz is hozzájárult, hogy az Egyetemen maradjak ezekre a javarészt vidám évekre. Köszönöm, uraim.

Lehetetlenség felsorolni azokat a munkatársakat, akik segítettek és biztattak a disszertációm megírása közben. Bíró József, Csopaki Gyula, Halász Edit és Henk Tamás bizonyosan rajta vannak ezen a (névsor szerinti) listán, de igazán nem lehet

mindenkit név szerint megemlíteni, mert oly sokan vannak. Ha segítettél nekem, észrevettem és hálás vagyok érte. Köszönöm.

Végül, de nem utolsósorban, a szimulációs vizsgálatokhoz az Egyetem „superman" nevű szuperszámítógépét használtam. Ennek a csodás gépnek a segítségével a szimulációim hetek alatt futottak le, egyébként mindez évekig tartott volna. Ezen infrastruktúrát a TÁMOP-4.2.2.B-10/1–2010-0009 projekt támogatta.

Contents

1 Introduction **1**

1.1 Related Work 4

1.2 Organization 9

2 Virtual Resource Allocation Overview **10**

2.1 Traffic Engineering, OSPF-TE 11

2.2 Resource Bounds 12

2.3 Overview of VRA Optimization Strategies 13

2.3.1 OSPF Weight Optimization 14

2.3.2 Overlay Optimization 14

2.3.3 Peer-Local Optimization 15

2.3.4 Peer-Global Optimization 16

2.3.5 Summary of the Optimization Strategies 17

2.4 Other Use Cases 17

3 Overlay Optimization **18**

3.1 VRA-1N-1D Problem Definition 20

3.2 Bounds on the Error 21

3.3 Optimal Solution of VRA-1N-1D 22

3.4 Other Problem Formulations 27

3.4.1 Application for WCMP 29

4 Peer-Local Optimization **30**

4.1 VRA-1N-mD Problem Definition 32

4.2 Attributes of VRA-1N-mD and VRA-1N-mD-Unlimited 35

4.2.1 Completeness of the Problems . . . 35
4.2.2 Bounds on the Error . . . 36
4.3 Unlimited Number of Links . . . 37
4.3.1 Consistency of a VRA-1N-mD-Unlimited Problem . . . 37
4.3.2 Notes on the Types of the Solution . . . 42
4.3.3 An LP-based Iterative Solution . . . 44
4.3.4 A Special Case: Positive Matrix G . . . 46
4.4 Limited Number of Links . . . 47
4.4.1 An ILP-based Iterative Solution . . . 47
4.4.2 A Direct ILP Formulation . . . 49
4.4.3 A Heuristic Solution . . . 51

5 Peer-Global Optimization **53**
5.1 VRA-PGO Problem Definition . . . 53
5.2 Optimal Solution . . . 55
5.3 Computational Complexity . . . 59
5.3.1 NP-Completeness of VRA-PGO . . . 59
5.3.2 Inapproximability of VRA-PGO . . . 68

6 Numerical Evaluation **76**
6.1 Examined Algorithms . . . 76
6.1.1 Global Optimization . . . 77
6.1.2 OSPF Weight Optimization . . . 78
6.1.3 Implementation Aspects . . . 78
6.2 Simulation Scenarios . . . 79
6.3 Simulation Results . . . 81
6.3.1 Resource Consumption . . . 83

7 Conclusions **85**
7.1 Summary . . . 85
7.2 Possible Future Work . . . 87

Bibliography **92**

Index **100**

Appendix A List of Theses **102**

Appendix B List of Problem Definitions **106**

Appendix C Auxiliary Proofs **108**

C.1 Proof of Theorem 16 108

C.2 Proof of Theorem 18 114

C.3 Addendum on Computation Complexity 120

C.3.1 X3C Reduction 120

C.3.2 Good Simultaneous Approximation Reduction 123

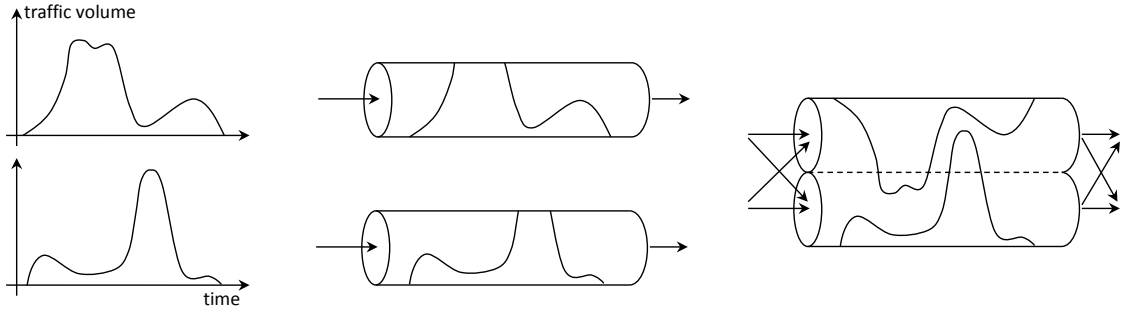
Appendix D Publications **126**

Chapter 1

Introduction

Unity is strength. Treating separate network resources as one and sharing it among users is a technique inherent to the Internet. This scheme, often called the Resource Pooling Principle [1], can be observed at several aspects of today's networks. Examples of this principle include multipath routing, multihoming, Ethernet Link Aggregation Groups [2], load balancing between application level servers (such as web-servers or database servers), load balancing in Traffic Engineering. Content Delivery Networks [3] are also a form of resource pooling, just as cloud storage and cloud computing [4]. To realize these services, data centers are being installed rapidly, often utilizing parallel paths, which are, in many of the cases, asymmetric in capacity [5]. Furthermore, several new concepts, such as network virtualization and Software Defined Networking (SDN) [6] appeared in the recent years, which also take advantage of the pooling principle in order to optimally exploit the network resources.

This list is far from being comprehensive, yet it shows the versatility of scenarios where resources are pooled. There are several reasons to do so. First, its inherent redundancy increases the robustness against component failures. Second, by dynamically allocating more resources for a temporal peak usage higher level services can be offered on the same infrastructure, utilizing statistical multiplexing. Third, having a greater freedom to couple demands and resources more efficient network utilization can be achieved along with a more scalable service.



(a) Demands (b) Without resource pooling (c) With resource pooling

Figure 1.1: Resource pooling example

A simple example of resource pooling is shown in Fig. 1.1[1]. If the links and the demands are coupled in a one-to-one fashion (Fig. 1.1(b)) then lossless transmission is not possible. With pooled resource usage (Fig. 1.1(c)), however, both demands can be forwarded without data loss. The latter scenario is also more robust against failures: if one link goes down, without pooling the service is completely denied for the related demand. Shared usage of the links, however, guarantees some level of service for both demands in case of a single link failure.

The implementation of resource pooling, however, is challenging as the load balancers can often split the incoming demands only roughly equally amongst the resources (because it is simple and scalable this way). As an illustration, a load balancer between two web-servers typically splits the incoming requests in half, which heavily hinders the overall performance if one of the back-end servers are for instance twice as powerful as the other. Likewise, in routing protocols such as OSPF [7] or IS-IS [8] Equal-Cost Multipath (ECMP) is used to distribute the traffic over the shortest paths with the same cost. ECMP, however, is only able to split traffic between these paths uniformly, even if they have different capacities, which poses a giant barrier when aspiring to an optimal Traffic Engineering [5, 9, 10, 11].

As a solution, I introduce a technique called Virtual Resource Allocation (VRA) to realize optimal resource pooling over legacy equal-split load balancing schemes. The basic idea of VRA is to virtually multiply the available parallel resources so that the load balancing system sees a greater number than what actually exists. The virtual resources are then grouped and assigned to the physical ones, thereby tricking

[1]This figure is based on Fig. 2 of paper [1].

the legacy equal splitting technology into approximating the required non-equal load division over the existing media.

To continue the previous example, one can install two virtual machines on the more powerful web-server and present them, along with the unmodified less powerful server, to the load balancer. It then sees three servers, and by realizing equal split between them the higher capacity one will eventually end up with 2/3 part of the total load, as desired. In a similar fashion, installing virtual links or paths alongside the physical ones (which, in practice, can be carried out via some administrative settings), ECMP's equal-split limitation can be amended. If, for example a $25\% - 75\%$ traffic proportion is desired on two, equal cost shortest paths A and B, then by installing two virtual paths parallel to B, and presenting these four to ECMP, it will happily realize the expected traffic split rate.

The engineering problem to solve in VRA is then to come up with an optimal setting of virtual resources so that a predefined non-equal traffic split ratio is approximated sufficiently with limited resource usage. Furthermore, placing VRA in a broader scope, other, network-wide goals can be targeted as well.

One advantage of my VRA proposition is that it is incrementally deployable, since it is perfectly fine to set up virtual resources only at a subset of the network nodes. Moreover, unlike most other proposals, VRA is fully compatible with existing hardware or software components in the network. Finally, VRA is extremely efficient, as my numerical results indicate that by adding only a small set of virtual resources the ideal traffic split ratio can be very well approximated, resulting in substantial performance gain.

Later on in this dissertation Traffic Engineering (TE) in IP networks will be used to introduce the VRA proposal. The idea is to set up virtual links alongside the existing ones and present them to OSPF, as described above. This way near-optimal TE can be achieved without any hardware or software modification on the network infrastructure. Let me emphasize, however, that TE is just a descriptive example application of the VRA concept, and its possible fields of usage are much broader. To name one other use case, in certain SDN-based scenarios VRA can be used for rule table optimization.

1.1 Related Work

In this section I briefly overview the most relevant publications.

Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks [12] targets the same problem as I do in my TE example: overcoming the equal split limitation of the OSPF/IS-IS routing protocols. Their basic idea is to modify the forwarding table and this way controlling the set of shortest paths assigned to different routing prefix entries. If this is done properly, the ideal traffic split ratio can be approximated without changing the routing protocols or the forwarding mechanism of the routers. However, unlike at VRA, in this proposal the control path of the routers are affected by the modified way of the forwarding table maintenance.

Penalizing Exponential Flow-spliTting (PEFT [13]) is a proposal for a provably optimal traffic engineering using link state protocols and hop-by-hop forwarding. Dissimilarly from VRA, here the traditional operation of the routing protocols is modified: not only the shortest paths are used but all of them, and the amount of traffic on a path depends on the total path length. PEFT provides quick and optimal traffic engineering, but at the expense of modifying the routing basics and using unlimited number of next hops.

Weighted Cost Multipath (WCMP [5]) targets unequal traffic splits at data centers. It assumes SDN-capable switches, yet the installed rules are based on longest prefix matching, just like in the case of a traditional routing protocol. The WCMP proposal assigns weights to each egress port in a multipath group, and realizes traffic split proportional to the weights by essentially adding several duplicated entries to the multipath forwarding table. The total number of table entries is constrained, as in the VRA case. On the other hand, using SDN rules enables the designers to treat each demand separately, avoiding the unfortunate coupling of independent demands, which happens in my VRA-1N-mD problem presented in Section 4.1.

WCMP is therefore similar to my work both in terms of the goal and the applied technology (longest prefix match-based forwarding). The similarities continue if we look at the WCMP problem formulation. Their fundamental mathematical problem is perfectly identical to my VRA-1N-1D case (see Sec. 3.1): find a set of integers that sums to a low number and their relative quotients approximate a given ratio. Even

the error function is essentially the same across the two papers: the maximum of ratios of the actual and the intended traffic per output ports.

While paper [5] proposes heuristic solutions only, my algorithms, which can directly be applied to the WCMP problems, provide optimal solutions for this case and not just approximations. Furthermore, my proposals can achieve this with comparable or even smaller computational complexity.

Fibbing [14] is a fresh proposal aiming to compound the advantages of traditional routing protocols (primarily: scalability and robustness) and the ones of SDN-based routing (easy manageability and flexibility). The basic idea of Fibbing is to inject fake nodes and links through standard routing protocol messages, thereby effectively "lying" to other participants of the routing. The applicability of Fibbing for per destination load balancing with uneven splitting ratios has recently been demonstrated [15]. If a routing has a single shortest path for a destination and two parallel paths are to be used with equal traffic share, advertisements of a fake node and a fake link has to be injected to the network. Likewise, if for example 33%–67% traffic ratio is to be achieved in an unequal load sharing case then two fake nodes and links have to be advertised. My VRA-1N-1D algorithm (Sec. 3.1) can be used with Fibbing to find the best approximation of an arbitrary split ratio using bounded number of fake entities. Nevertheless, it is yet unknown if the operators will favor the advantages provided by Fibbing over the extra abstraction level it requires.

Niagara [10] is another SDN-based proposal, which provides flexible traffic splitting between load balancers. Its goal is inherently the same as mine: to divide the incoming traffic (towards different servers in this case) according to a given ratio. This is achieved by a set of rules for selecting the next hop, taking into account the destination IP address and some of the least significant bits of its source IP address, too. The goal is to approximate the given split ratio with a small number of rules.

The underlying mathematical problem is similar to mine: try to approximate a given ratio by the sum of fractions of small integers. In VRA I will mostly use fractions with common denominator, Niagara uses sums of $1/2^i$. I will aim for a small denominator, in the Niagara case the number of terms in the sum should be kept low. With some clever enhancements to Niagara negative terms can be allowed in the sum as well (e.g., $1/4+1/16-1/64$) and by sharing rules among different traffic aggregates

the number of rules can be further lowered. This latter case is somewhat similar to my VRA-1N-mD (see Sec. 4.1) problem.

Niagara, just like WCMP, can treat different demands independently of each other, and by using the source address as well for rule creation it can achieve a more concise rule table than WCMP. As shown in Sec. 5.1 of [10], NIAGARA utilizes a greedy algorithm to minimize the total number of rules for multiple aggregates. I have adopted this idea in Alg. 3.4 (p. 28), and proven to be optimal in Sec. 3.4. Niagara seems to be a promising and powerful tool. On the other hand, VRA poses much lower requirements on the network, and is therefore more easily deployable.

COYOTE [16] is recent proposal applying the VRA concept. It is designed to be a readily deployable TE scheme for robust and efficient network utilization. COYOTE takes as input a capacitated network, and a set of traffic demands with a source and destination node and a traffic volume range (so-called "uncertainty bounds"), within which the traffic amount can change arbitrary. It then calculates static traffic splitting ratios that are optimized with respect to all scenarios within the uncertainty bounds. These ratios are then realized in the network by combining the Fibbing way of injecting fake protocol messages and some of my algorithms to optimize the number of the fake entities.

Data centers are a special and nowadays very important field of application of load balancing. In a data center high speed interconnections are required between the nodes, which are typically realized by parallel data paths [17]. These paths, on the other hand, are often asymmetric in capacity [5], demanding an adequate load balancing technique. There are several techniques and proposals for this purpose and I highlight some of the most important ones below.

ECMP [18] and the flowlet-based Flare [19] are traditional choices in data centers to realize multipathing. They, however, can only split the traffic evenly and not proportionally to the available resources if the parallel paths are asymmetric in capacity. One possible workaround is to handle the large "elephant" flows separately, like Hedera [20] does. WCMP is also intended to be used in data centers, just like Presto [21], which realizes load balancing in the soft network edge (virtual switches) and DRB [22], which is a per-packet routing algorithm for data centers. FlowBender [23] tackles the issue at the flow level, by taking advantage of the congestion notifications. CONGA [24] and HULA [25] operate with flowlets, handling the traffic

at a finer granularity. Both use an explicit feedback mechanism. For CONGA it is signaled leaf-to-leaf, while HULA uses periodic probes with a distance vector style distribution of the network utilization information. DRILL [26] slightly modifies the data plane to be able to handle very short term congestions. Finally, LetFlow [27] takes advantage of the realization that the size of the flowlets changes automatically with the traffic conditions on their path. Using this elastic property LetFlow can reach nearly as good performance as the ones that use information on the global traffic conditions of the data center, like CONGA, but by making very simple decisions based on local information only.

The *actual realization in the data plane* of a predefined (equal or non-equal) splitting rate is out of the scope of my dissertation, but such a functionality is naturally required for the traffic splitting. This subproblem is not straightforward, either. A packet-based solution can very accurately achieve the required splitting ratio, but it may cause reordering of datagrams of a single flow, which could cause disturbances in the higher network layers [19]. Using network flows as the unit of splitting, on the other hand, may be a less exact solution [28]. Accordingly, an intermediate approach, the so-called flowlet-based splitting has been proposed [19] and it is now available in commercial routers and servers [29]. Another—in a sense packet based—option is the increasingly popular Multipath TCP (MPTCP, [30]), which has recently been incorporated into the flowlet-aware HULA in the MP-HULA proposal [31].

The packet-to-interface mapping is usually realized via hashing. In its simpler form hashing only supports equal split, but a table-based hashing can be used to implement unequal split ratios [32]. Algorithm 3.1 (see p. 25) could actually be used to compute the optimal thresholds or indexes in such algorithms (see Sec. IV.B in [32]), but if the number of bins is much larger than the number of outgoing links then simpler approximation algorithms would suffice as well.

Some related work focus on the *utility of splitting*. Papers [33] and [34] examine the performance gap between a splitting and a non-splitting scenario under various assumptions. If traffic splitting is allowed it is often constrained, just like in my work. In [35] the number of paths available for a demand is limited, which is similar to my Overlay Optimization problem discussed in Chapter 3. However, there the authors aim for network utility maximization, which means to maximize the aggregate utility of each source-destination pair, where the utility function is a concave, increasing,

Topic	Papers referenced	Related contribution
Optimal Traffic Engineering	Achieving near-optimal... [12], PEFT [13], COYOTE [16]	Thesis Groups 1, 2, 3 in Chapters 3, 4, 5
Flexible traffic splitting	Achieving near-optimal... [12], WCMP [5], Niagara [10], Fibbing [14] (partially), COYOTE [16] (partially), Utility of splitting: [33], [34], [35], [36]	Thesis Group 1 in Chapter 3
Multipath support in data centers	WCMP [5], Flare [19], Hedera [20], Presto [21], DRB [22], Flowbender [23], CONGA [24], HULA [25], DRILL [26], LetFlow [27], MP-HULA [31]	Thesis Group 1 in Chapter 3
Realization in the data plane	Summary: [28], Hashing performance: [32]	Not in the immediate scope of this work

Table 1.1: Summary of Related Papers

continuous function of the throughput. I have, on the other hand, worked with fixed demands for the source-destination pairs, where transferring less is unsatisfactory and transferring more is superfluous. This corresponds to a non-continuous utility function. Accordingly, I have aimed for maximum link load minimization, which is a fundamentally different objective from network utility maximization. Finding the optimal traffic values for the sources and the best routing simultaneously while obeying the path cardinality constraint is practically intractable so the authors had to revert to heuristic solutions.

A follow-up of this work is presented in [36], where instead of the number of paths the split ratio granularity is constrained. This is again similar to my Overlay Optimization, however in [36] the demands are split into exactly p parts whereas in my work it is at most p ($E < Q$ with my notations). Furthermore, their objective is still the network utility maximization while mine is the maximal link load minimization.[2] Due to these dissimilarities the results, while being similar, are not directly comparable; nevertheless, they complement each other in the field of unequal traffic splitting.

Table 1.1 shows a short summary for this Related Work section indicating the association between the references and the different parts of this dissertation.

[2]Note that [36] also refers our paper [37], and points out this difference.

1.2 Organization

This dissertation consists of seven main chapters and an appendix. After this Introduction, Chapter 2 introduces the VRA concept with some simple examples. It also covers the different possible constraint types and the potential optimization strategies. Chapters 3, 4 and 5 carry my main theoretical results about different versions of the original problem: Overlay Optimization, Peer-Local and Peer-Global Optimizations, respectively. The numerical evaluation, which includes my algorithms as well as the existing best-practice solution, is presented in Chapter 6. Chapter 7 concludes this dissertation, also incorporating a section on possible future work.

After the Bibliography and the Index, in Appendix A the list of my theses is presented. Appendix B lists the problem definitions given throughout this work. Appendix C contains proofs that were too long to fit into the main text without disturbing its readability. Finally, Appendix D lists my scientific publications.

Chapter 2

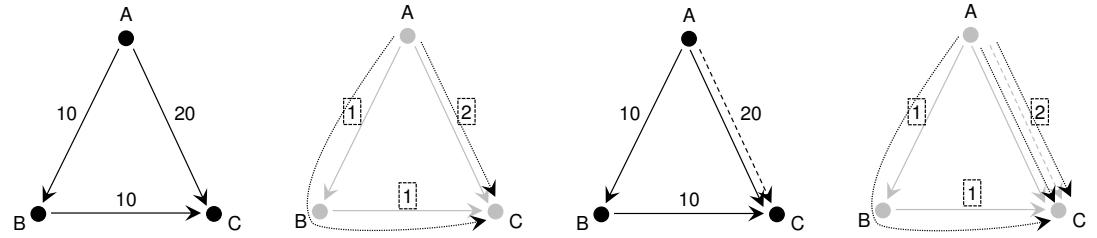
Virtual Resource Allocation Overview

In this section I present an overview of the Virtual Resource Allocation concept, using Traffic Engineering as a descriptive example.

The idea behind VRA is fairly simple and is best explained by a small sample problem. Consider the triangular network shown in Fig. 2.1(a). Suppose we would like to transfer 30 units of traffic from A to C without overutilizing any of the links. Using stock OSPF would allow us to set the link weights[1], thereby we could easily create two equal cost shortest paths (i.e. paths with minimal total cost/weight): $A-B-C$ and $A-C$, by using for example the weights shown in Fig. 2.1(b). On the other hand, OSPF ECMP only allows splitting the traffic equally between the shortest paths implying a 150% load on links $A-B$ and $B-C$.

If, however, we could set up a virtual link on top of the existing link $A-C$ and expose it to OSPF (see Fig. 2.1(c)), it would happily split the traffic in three, sending one third on path $A-B-C$ and the rest on physical link $A-C$ (Fig. 2.1(d)). Naturally, installing a virtual link over physical link $A-C$ does not change its capacity, it only enables OSPF ECMP to use its full potential. The link weights would also remain unchanged, and the new virtual link would have the same weight as the respective physical one. By this simple administrative intervention we can route the traffic through this network without exceeding the link capacities.

[1]Link weights are also often called link costs, link metrics or SPF (Shortest Path First) metrics. I will use these terms interchangeably.



(a) Original topology with link capacities (b) Link weights and shortest paths (c) Adding a virtual link (d) Link weights and shortest paths

Figure 2.1: A triangular network. Demand: $A \rightarrow C : 30$

There are several possible ways to set up a virtual link parallel to an existing one. These options include Ethernet Virtual LANs (VLANs), IP-IP tunnels, Generic Routing Encapsulation (GRE) tunnels, etc. The exact method of setting up this Layer 2 connection is out of the scope of this dissertation, I only focus on the effect of the virtual links on the network performance.

2.1 Traffic Engineering, OSPF-TE

Traffic Engineering (TE) [38] is the scientific area of performance management in operational communication networks. Several methods exist for assigning traffic flows to data paths and thereby approximating optimal network utilization, perhaps the most well-known being MPLS [39] with RSVP-TE [40] (MPLS-TE [38]). However, for a network with N nodes and a full traffic matrix it requires N^2 label switched paths. Either for this or for other reasons some operators are reluctant to deploy MPLS-TE in their network.

A less demanding alternative is *OSPF Traffic Engineering (OSPF-TE)*. Its basic idea is to adjust the administrative link costs so that the shortest paths calculated by OSPF will map exactly to the ones chosen by the administrator. These paths may be a result of an adequate linear program or some related heuristics [41]. The link costs can be inferred from the dual of a similar linear program [42]. There is, however, a fundamental problem with this solution. It may result in several parallel equal cost shortest paths, with different amount of traffic to be transmitted over them. On

the other hand OSPF ECMP provides only even split[2] and consequently for certain networks the quality of OSPF-TE can become arbitrarily poor compared to optimal TE [9].

Finding the best link weight configuration for a network with OSPF ECMP (a process called "OSPF weight optimization") is not straightforward, either. It is well known to be NP-hard [9], and as a recent study revealed even approximating it by a computationally efficient algorithm within *any* constant ratio is infeasible [43]. Still, there are proposed weight optimization heuristics that perform well in real-life scenarios [44, 45].

2.2 Resource Bounds

We shall see that generally a network using VRA performs better as the number of applicable virtual links grows. In practice, however, the number of next hops one can provision for a particular destination entry in the routing table is typically limited by the OSPF implementation, in line with Section 16.8 of the OSPF RFC [7]. For example, in many Cisco, Ericsson and Juniper routers this limit is adjustable, but the maximum allowed setting is 16 [46, 47, 48]. Similar limits exist for other routing protocol implementations, like EIGRP, IS-IS, RIP [46].

In other use cases similar bounds may exist. For example in SDN the number of rules to be installed have got a maximal value, too. For WCMP and Niagara this limit is in the order of several hundreds to several thousands. This is much larger than the number of allowed next hops in OSPF ECMP, but the important point is that there exists an upper bound.

On that ground, I shall also study the form of VRA, when an upper limit is given on the applicable resources. I will examine the following three models, which are important from theoretical and/or practical point of view:

1. *Unlimited Resources.* In this simple case we pose no upper bound on the maximum usable resources. Certainly, this approach is not directly applicable in real life

[2]The rationale for OSPF ECMP supporting equal-splitting only is out of the scope of this work. Nevertheless, two likely reasons are: (1) the link metrics do not contain the necessary information about the required split ratio, and (2) allows simpler hardware implementation in the data plane [32].

settings, but in some scenarios solving this simpler problem will lead to the solution of the more complicated ones.

2. Bounded Total Resources. In this model the total number of resources, including the real and virtual ones, are limited. For example in Sec. 3.1 the total number of outgoing links used for a demand is bounded by a constant Q. In this case, as Q is fixed, the higher the number of used physical links, the smaller the number of allowed virtual links.

3. Bounded Virtual Resources. Here the number of virtual resources is limited by a constant R, which is independent of the number of existing physical resources.

2.3 Overview of VRA Optimization Strategies

As shown above, installing virtual resources (virtual links in this particular case) in a pure OSPF ECMP environment can reduce the congestion in a network. It is not straightforward, however, *where* and *how many* virtual links to install in order to attain the ideal TE scenario. I propose several virtual resource allocation strategies, each aiming to answer these questions.

Unless otherwise stated, throughout this dissertation the applied metric of TE optimality is the widely adopted *Maximal Link Utilization (MLU)*. The link utilization is defined as the link traffic volume divided by the link capacity, and the MLU is the maximum of this measure over all the links of the network. The goal is of course to minimize the MLU.

The different VRA approaches are introduced using the example network shown in Fig. 2.2. The only demand to route is from A to D, with a volume of 35 units of traffic. Clearly, the optimal solution fully utilizes all the links, resulting in MLU of 1.0. On the other hand, achieving this in an OSPF routing environment is not a trivial task. Let us limit the number of allocated virtual links per node to $R = 4$ (cf. *Bounded Virtual Resources* in the previous section); or the total number of usable next hops for a destination to $Q = 6$ (*Bounded Total Resources* model), which are identical in this case.

This example is used in the following subsections to enumerate different TE strategies, which are discussed in depth in the upcoming chapters.

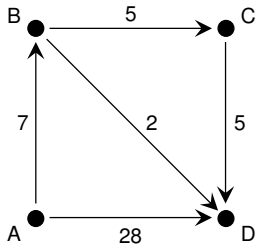


Figure 2.2: A capacitated example network. Demand: $A \to D : 35$

2.3.1 OSPF Weight Optimization

In this simplest scenario we are not utilizing any kind of virtual resources. Consequently, our degree of freedom is limited to setting the link weights, so that running OSPF ECMP over the shortest paths will generate optimal result (realizing OSPF-TE). I will use this case as one of the reference points in my evaluation.

In the current example the best we can do is having one minimum cost path, $A - D$, by setting for example all link weights to 1. This causes MLU of $35/28 = 1.25$. (See Table 2.1 for the MLUs of this example.)

2.3.2 Overlay Optimization

In this important scenario I suppose that a set of source–destination tunnels are already set up; yet, splitting the traffic between these tunnels have to be done somehow. Having MPLS-TE with a Path Computation Element (PCE) or some other kind of advanced means at our disposal, this problem can be tackled relatively easily. As another example, Cisco Express Forwarding (CEF) allows traffic splitting roughly proportionally to the MPLS tunnel bandwidth [49].[3] In many cases, however, no such tool is at hand, only the pre-allocated tunnels are present, and OSPF has to be used to transfer the traffic through this overlay network. In this scenario we can still achieve a near-optimal traffic distribution, by presenting virtual resources, in this case virtual paths (i.e., virtually multiplicated tunnels), to OSPF.

To examine this scenario, suppose we have three paths (tunnels) already set up, as shown in Fig. 2.3. If proportional load sharing was not implemented and these paths

[3]Note that even this implementation uses fractions of small integers to approximate the desired split ratio, just as I propose in this dissertation. In CEF the Bounded Total Resources model is used with $Q = 16$.

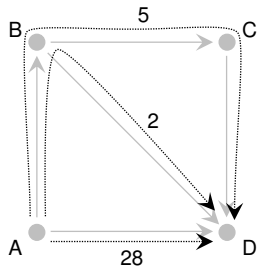


Figure 2.3: Paths for Overlay Optimization

were exposed to OSPF, it would split the traffic in three equal portions, resulting in a utilization factor of $(35/3)/2 \approx 5.833$ on link $B - D$.

Nevertheless, we could apply virtual paths to optimize the traffic. We would put the maximum allowed 4 virtual paths to the one hop path $A - D$, resulting in an $5 : 1 : 1$ traffic split ratio. In this case the MLU (still on $B - D$) would drop to $(35/7)/2 = 2.5$.

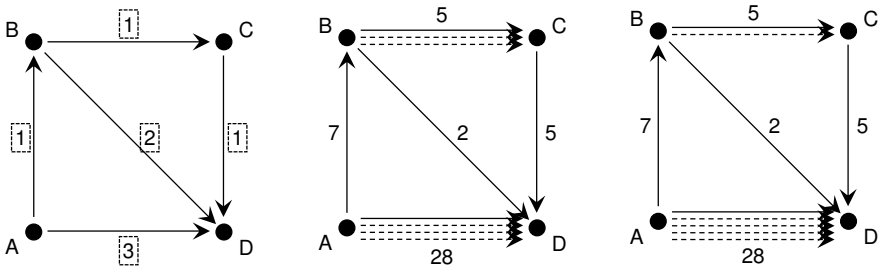
We are even better off if we allow not to utilize some of the paths at all. In this case we could disregard path $A-B-D$, and use the 4 virtual links to $A-D$, splitting the traffic in $5 : 1$. Now the maximal link utilization is on link $B - C$: $(35/6)/5 \approx 1.167$.

In this scenario traffic splitting occurs within an overlay network of pre-allocated tunnels, implying the name *Overlay Optimization*. I will refer to the latter case, where not all the paths are used, as *Overlay Optimization with Path Exclusion*.

2.3.3 Peer-Local Optimization

In this scenario there is no overlay network and traffic optimization takes place directly on the physical infrastructure. During the optimization the link weights are adjusted, and virtual links are set up.

We first compute the optimal routing, which is trivial in our example: fully utilize each link. Next, the link weights have to be set accordingly, meaning in this case that each of the three paths shown in Fig. 2.3 would be a shortest path. Fig. 2.4(a) shows a sufficient weight allocation. Finally, for each node where traffic splitting occurs, the desired split should be approximated by applying virtual links. This is done for each node individually, independently on the other nodes, hence the name *Peer-Local Optimization*.



(a) Link Weights (b) Peer-Local Optimization (c) Peer-Global Optimization

Figure 2.4: Peer-Local and Peer-Global Optimization

In our case at node A the $7:28$ split can be achieved exactly by adding 3 virtual links to $A-D$ (see Fig. 2.4(b)). At node B, however, the $5:2$ ratio cannot be perfectly realized by using only 4 virtual links (5 would be enough, though). The best we can do is allocating 2 virtual links to $B-C$. Thus the optimal virtual link allocation results in $(7 \cdot 3/4)/5 = 1.05$ utilization on links $B-C$ and $C-D$, which is the MLU in this case as well.

2.3.4 Peer-Global Optimization

There is a fundamental problem with the previous approach: as the errors of the local optimizations propagate downstream with the data flows they encounter other imperfect splitting points, whereby local errors can enlarge or weaken each other's effect. The unintended effects of this kind of cascade errors can be avoided by minimizing the errors concurrently, in a centralized manner. This is what I call *Peer-Global Optimization*.

In this case we determine both the link weights and the number of applied virtual links for each physical link simultaneously. Just as with the previous approaches, at this point I do not detail how to do it, only describe its potential. In the example network the optimal global solution uses the same link weights as Peer-Local Optimization (see Fig. 2.4(a)) and the virtual link provisioning is plot in Fig. 2.4(c). With this allocation the MLU will be on link $A-D$: $(35 \cdot 5/6)/28 \approx 1.042$.

Notice that this result is better than the one for Peer-Local Optimization. This is because we sacrificed the otherwise realizable perfect split at node A in order to

Optimization strategy	MLU
OSPF Weight Optimization (no virtual links)	1.25
Overlay Optimization	2.5
Overlay Optimization with Path Exclusion	1.167
Peer-Local Optimization	1.05
Peer-Global Optimization	1.042
Optimal Solution	1

Table 2.1: Performance of the different optimization strategies

lower the error downstream, at node B. This trick would hardly be possible using local considerations only.

2.3.5 Summary of the Optimization Strategies

The maximal link utilization of the different optimization strategies for our simple rectangular example network are shown in Table 2.1.

Although the main purpose of this example was to give a quick insight into the different TE approaches, the listed results also suggest that OSPF-TE enhanced by VRA may result in considerably better network performance than pure OSPF-TE. Actually, in this case Peer-Global Optimization approaches the theoretical optimum by only 4%—without using any kind of advanced traffic engineering technology. More realistic numerical studies are presented in Chapter 6.

2.4 Other Use Cases

It is important to emphasize again that the deployment of VRA algorithms are not limited to Traffic Engineering. Several other use cases are possible, including SDN rule optimization for WCMP or minimization of Fibbing virtual link and node numbers, as discussed in Sec. 1.1. See also Sec. 3.4.1 for application of VRA for WCMP.

In the following chapters I explain in detail, formalize and analyze these VRA optimization approaches. For the sake of simplicity in delivery I will continue to use TE as the scenario for VRA, keeping in mind that most of my algorithms can be used more generally.

Chapter 3

Overlay Optimization

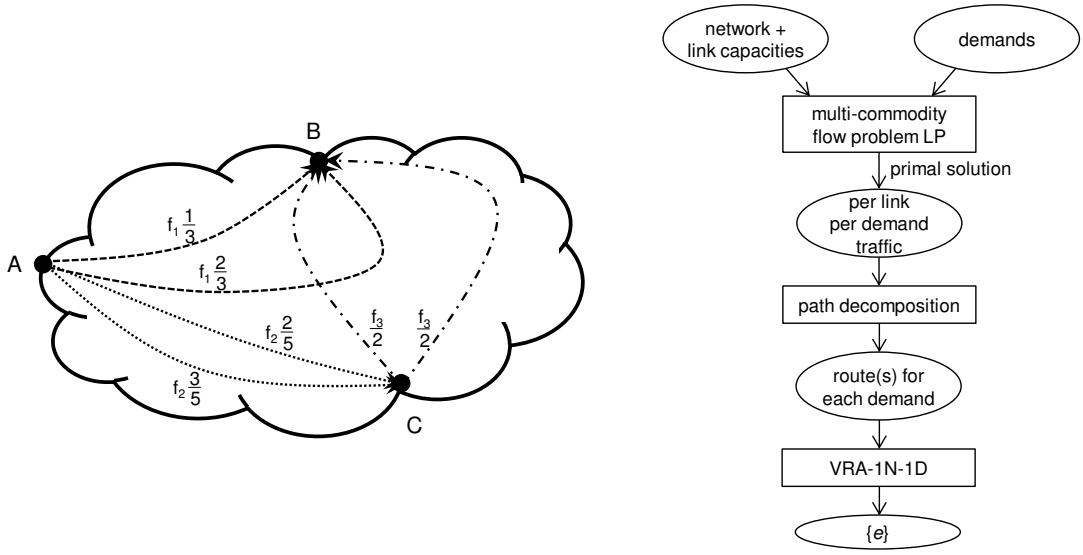
This chapter is devoted to Overlay Optimization [37], which has been shortly introduced in Sec. 2.3.2. The basic assumption is that end-to-end tunnels are already set up, using MPLS-TE for example, and OSPF ECMP is deployed on top of this overlay.

A sample scenario is plot in Fig. 3.1(a). In this simple transit network there are three edge routers A, B and C, and a full mesh MPLS overlay is realized between them containing two paths per router pair. This MPLS overlay, in turn, is seen as an IP topology deployed on top, which runs plain OSPF as a routing protocol. Easily, if the ideal traffic splitting ratios are like the ones given in the figure then this traffic allocation is impossible to implement with ECMP. With my proposed technique, however, we can set up 4 virtual links[1] (one between $A-B$ and three between $A-C$) to obtain exactly the required splitting.

The beauty of Overlay Optimization is that traffic splitting only occurs at the source nodes, meaning that the demands can be treated separately from each other. For example, adding virtual links between A and B does not affect the transmission of the other demands in any way.

Overlay Optimization can also be used in the more general case, when only a capacitated network and the demands are given, and we can assume the ability of setting up (possible parallel) end-to-end tunnels. In this case we first have to calculate a set of end-to-end tunnels, then use the VRA Overlay Optimization method over these paths. The major steps of my proposed technique are shown in Fig. 3.1(b).

[1]The phrase "virtual path" could be more appropriate in this case, but for simplicity I continue to call them virtual links.



(a) Overlay Optimization example

(b) Overlay Optimization with tunnel/route setup

Figure 3.1: Overlay Optimization

The LP (Linear Program) for the multi-commodity flow problem, which allows branching [50], can be solved in polynomial time and supplies the per link per demand traffic volumes as the primal solution. The next step is to combine these per link traffic volumes into end-to-end routes: generally there is more than one route for a demand, each with a possibly different traffic share. This is called *path decomposition* (or subflow decomposition), which can be done in several ways, resulting in higher or lower total number of routes. There are polynomial time algorithms for the decomposition, like *SimPol* proposed in [51], but finding the minimal number of routes is a strongly NP-hard problem [52], which cannot even be efficiently approximated better than some fixed constant [53].

The final, and for now the most important step is denoted by VRA-1N-1D in the figure, which stands for "VRA for One Node, One Demand". Indeed, as explained above, here each demand can be treated separately, and splitting only occurs at their sources. This means that the VRA problem can be decomposed into D independent VRA-1N-1D problems, easing the underlying mathematical problem substantially.

Using the method summarized in Fig. 3.1(b) I will be able to compare Overlay Optimization with the other techniques, as described in the Evaluation section.

Notation	Description
k	number of outgoing links used
$g_1, g_2, \ldots, g_k \in \mathbb{Z}^+$	desired traffic volume per outgoing links
$G_0 = \sum_{i=1}^{k} g_i$	total traffic volume
$h_1, h_2, \ldots, h_k$	actual traffic volume per outgoing link
$U_i = h_i/g_i$	error on the ith outgoing link
$U = \max_i U_i$	error of a virtual resource allocation
$e_1, e_2, \ldots, e_k$	number of allocated links (physical and virtual together)
$E = \sum_{i=1}^{k} e_i$	total number of allocated links
$Q \in \mathbb{Z}^+$	upper bound on the total number of links

Table 3.1: VRA-1N-1D notations

The rest of this chapter is devoted to solving the VRA-1N-1D problem, which is a crucial step of Overlay Optimization. I start with a precise problem definition.

3.1 VRA-1N-1D Problem Definition

We are given a single node, where the traffic of a single demand has to be split. It is supposed to use k outgoing links (or paths/tunnels, but for simplicity I will use "link" in the remainder of this section), each with $g_1, g_2, \ldots, g_k$ desired traffic volume (see Table 3.1 for a list of notations). We can safely suppose that $g_i \in \mathbb{Z}^+$.[2] Furthermore let $G_0 = \sum_{i=1}^{k} g_i$.

Our objective is to share the traffic over the outgoing links using OSPF ECMP such that the actually emerging $h_1, h_2, \ldots, h_k$ subflow values are as close as possible to the nominal $g_1, g_2, \ldots, g_k$ subflow volumes. Here "closeness" between the ith subflows is defined as the per link error $U_i = h_i/g_i$, and the ultimate error metric to be minimized (U) is the maximum of the per link errors.[3]

Note that this time I compare the actual traffic to the desired traffic volume, not to the link capacities (MLU). The reason for this is that VRA-1N-1D is just a part

[2]Using integers for the traffic volumes simplifies the analytical study of the problem. On the other hand, it is not a real restriction, as any rational division ratio can be expressed this way. The absolute values of the volumes can be varied by changing the unit, so this should not cause problem, either. For example 1.5 Mbps is not an integer value, but 1500 kbps certainly is.

[3]Note that I refer to U_i and U as "errors", but in fact they represent actual-to-required traffic ratios. Usually zero or close-to-zero errors are preferred, but in this case $U = 1$ is the ideal condition.

of Overlay Optimization, which only focuses on realizing the traffic split rate given by $\{g_i\}$. This approach also makes VRA-1N-1D reusable in other resource pooling schemes, like WCMP (SDN) and Fibbing, as described in Sec. 1.1. Certainly the MLU metric can be used for Overlay Optimization as a whole, as shown in Chapter 6.

To reach our objective, I apply virtual links parallel to the physical ones. Let e_i denote the total number of (virtual and physical) links in place of the ith physical link, and $E = \sum_i e_i$ the total number of allocated links. To save space, I only examine the case without link disabling possibilities (i.e., no "Path Exclusion": $e_i > 0$).

Applying the equal-split principle of OSPF ECMP we get:

$$h_i = G_0 \frac{e_i}{\sum_{j=1}^{k} e_j} = \frac{G_0 e_i}{E}, \qquad i = 1 \dots k$$

and

$$U = \max_i \frac{h_i}{g_i} = \max_i \frac{G_0 e_i}{E g_i} \ .$$

As described in Sec. 2.2, the total number of outgoing links for a demand is limited in the practical router implementations. Consequently, I use the *Bounded Total Resources* model here, requiring $E \leq Q$.

The following formal definition summarizes this section:[4]

Problem 1, VRA-1N-1D. *Given k, $\{g_i\}$ and Q, find $\{e_i\}$ that minimizes U such that $\sum_i e_i \leq Q$.*

3.2 Bounds on the Error

Let us examine the theoretical bounds on the error of VRA-1N-1D, starting with a lower limit:

Lemma 1. $U \geq 1$.

Proof. By contradiction: $U < 1$ means $\forall i : G_0 e_i/(E g_i) < 1$, i.e., $G_0 e_i < E g_i$. Summing these over i yields: $G_0 \sum_i e_i < E \sum_i g_i$, i.e.: $G_0 E < E G_0$. □

Easily, if the number of links is unlimited, $U = 1$ can always be reached:

[4]For a comparative list of problem definitions see Appendix B.

Lemma 2. *If $Q \geq G_0$ then $\exists\{e_i\}$ for which $U = 1$.*

Proof. Using $e_i = g_i$ means $E = G_0$ and $U_i = G_0 e_i/(Eg_i) = 1 \ \forall i$. □

Corollary 3. *If $Q = \infty$ then $\exists\{e_i\}$ for which $U = 1$.*

There is a simple upper bound on the error, which will be useful later on:

Lemma 4. $U \leq G_0$.

Proof. Since $g_i \geq 1$ (as $g_i \in \mathbb{Z}^+$) and $e_i \leq E$, $\forall i$: $U_i = G_0 e_i/(Eg_i) \leq G_0$. □

A stronger upper bound is:

Lemma 5. $U \leq \frac{G_0}{\min_i g_i}$.

Proof. Similarly to the previous proof, $\forall i$: $U_i = G_0 e_i/(Eg_i) \leq G_0/g_i$. Therefore $U = \max_i U_i \leq \max_i G_0/g_i = G_0/\min_i g_i$. □

This latter bound is also applicable if the desired traffic split ratio is given by real numbers in the form of $\{\gamma_i\}$: $\sum_i \gamma_i = 1$. ($\gamma_i = g_i/G_0$ can be used if integer g_is are given.) In this case the bound is $U \leq 1/\min_i \gamma_i$.

The final remark on the error limits is about large G_0s:

Lemma 6. *If G_0 is unbounded and E is bounded by a finite Q then U can be arbitrarily high for any $Q > 2$.*

Proof. Let $k = 2$, $g_1 = 1$, $g_2 = x$, so that $x > Q$ ($x \in \mathbb{Z}$). Then $G_0 = x + 1$ and the optimal allocation of links is $e_1 = 1$, $e_2 = Q - 1$. The link traffic volumes are $h_1 = (x + 1)/Q$, $h_2 = (x + 1)(Q - 1)/Q$. The errors are $U_1 = (x + 1)/Q$, $U_2 = (x + 1)(Q - 1)/(Qx)$. Since $(Q - 1)/x < 1$, $U_2 < U_1$, meaning that $U = U_1$, i.e.: $U = (x + 1)/Q$, which can be arbitrary high, as Q is fixed and x is unbounded. □

3.3 Optimal Solution of VRA-1N-1D

Now I answer the question: which virtual link allocation minimizes the error? As there are only finite link allocations due to the constraint $E \leq Q$, an exhaustive search might be a possibility. To check its validity first let us cont the number of possible allocations:

Lemma 7. *The number of possible VRA-1N-1D allocations is* $\binom{Q}{k}$.

Proof. We have to dispense $Q-k$ virtual links between $k+1$ places: the first k places are the k physical links, and the last one is a place for the unused virtual links (allowing $E < Q$). This is a combination with repetitions with the number of possibilities being: $\binom{(Q-k)+(k+1)-1}{(k+1)-1} = \binom{Q}{k}$. □

For a given Q and for $1 \leq k \leq Q$, $\binom{Q}{k}$ is the largest for $k = \lfloor Q/2 \rfloor$. For example the largest number of possible VRA-1N-1D allocations for $Q = 30$ is $\binom{30}{15} \approx 1.55 \cdot 10^8$; for $Q = 50$ it is $\binom{50}{25} \approx 1.26 \cdot 10^{14}$. A well-known lower bound on binomial coefficients is: $\left(\frac{Q}{k}\right)^k \leq \binom{Q}{k}$. For the "worst case", i.e., $k = Q/2$ this lower bound is $2^{Q/2} = (\sqrt{2})^Q$, an exponential function.

This means for small Q values (like $Q \leq 30 \ldots 50$) an exhaustive search may be feasible, but not for much larger ones. For the given example of OSPF-TE $Q \leq 16$ is probably enough in most practical cases, but VRA-1N-1D can be used in other scenarios, where Q could be in the order of thousands as well (like SDN, where Q represents the maximal rule number). Therefore a more computationally efficient solution is necessary.

First I present an Integer Linear Program (see LP 3.1 on the following page), which solves the problem. The idea is to minimize the error U, by requiring for each link error:

$$U_i = \frac{h_i}{g_i} = \frac{e_i G}{E g_i} \leq \alpha \qquad i = 1 \ldots k \tag{3.1}$$

and minimizing α. Variables y_i help to find the optimal E: $y_i = 1$ if $E = i$ and $y_i = 0$ otherwise, which is enforced by constraints (3.3) and (3.4). The (3.5) system of constraints is only effective if $E = j$, and then results in inequality (3.1). The second term of the objective function (3.2) ensures that if there are several optimal solutions, then the solver would choose the one with the smallest number of links altogether. Although this would not be necessary as long as $\sum e_i \leq Q$, it is somewhat nicer to have it.

Due to the nature of the problem, LP 3.1 must contain integer variables. This, on the other hand, means that it is not guaranteed to run in polynomial time. Therefore I present an iterative algorithm that can solve the link allocation problem in pseudo-polynomial running time.

LP 3.1 VRA-1N-1D

$$
\begin{aligned}
\textit{variables: } & e_i && (e_i \geq 1,\ e_i \in \mathbb{Z},\ i = 1 \dots k) \\
& y_j && (y_j \in \{0,1\},\ j = 1 \dots Q) \\
& \alpha && (\alpha \in \mathbb{R}) \\
\textit{constants: } & Q && (Q \in \mathbb{Z}^+) \\
& g_i && (g_i \in \mathbb{Z}^+,\ i = 1 \dots k) \\
& G && = \sum_{i=1}^{k} g_i \\
& r && \text{(a small number, e.g., } 10^{-5}) \\
& M && \text{(a large number, e.g., } 10^{5})
\end{aligned}
$$

$$\textit{objective: } \min \alpha + r \sum_{i=1}^{k} e_i \tag{3.2}$$

$$\textit{constraints: } \sum_{j=1}^{Q} y_j = 1 \tag{3.3}$$

$$\sum_{i=1}^{k} e_i = \sum_{j=1}^{Q} j y_j \tag{3.4}$$

$$\frac{e_i G}{g_i} \leq \alpha j + M(1 - y_j),\ \ i = 1 \dots k,\ j = 1 \dots Q \tag{3.5}$$

Let us start with Algorithm 3.1 on the next page that checks for a given α, k, $\{g_i\}$ and E whether or not it is possible to assign the links with $U \leq \alpha$. If the assignment is feasible then it also provides a solution and indicates if it is the only solution. I prove that Algorithm 3.1 provides correct result:

Theorem 8. VRA-1N-1D-Fixed-E *can be solved with* $U \leq \alpha$ *if and only if* $\sum_{i=1}^{k} x_i \geq E$, *where* $x_i = \left\lfloor \frac{\alpha g_i E}{G_0} \right\rfloor$.

Proof. For any correct solution $\{e_i\}$, for all $i = 1 \dots k$:

$$\alpha \geq U = \max_{j=1\dots k} \frac{h_j}{g_j} \geq \frac{h_i}{g_i} = \frac{G_0 e_i}{E g_i}\ ,$$

thus $\alpha g_i E / G_0 \geq e_i$ and since $e_i \in \mathbb{Z}$,

$$\frac{\alpha g_i E}{G_0} \geq \left\lfloor \frac{\alpha g_i E}{G_0} \right\rfloor = x_i \geq e_i\ . \tag{3.8}$$

Algorithm 3.1 VRA-1N-1D-Fixed-E

Input: α, k, $\{g_i\}$, E
Output: $feasible$, $single\_solution$, $\{e_i\}$

for $i \leftarrow 1 \dots k$ **do**

$$x_i \leftarrow \left\lfloor \frac{\alpha g_i E}{G_0} \right\rfloor \tag{3.6}$$

end for
if $\sum_{i=1}^{k} x_i < E$ **then**
  $feasible \leftarrow$ `false`
else if $\sum_{i=1}^{k} x_i = E$ **then**
  $feasible \leftarrow$ `true`
  $single\_solution \leftarrow$ `true`
else
  $feasible \leftarrow$ `true`
  $single\_solution \leftarrow$ `false`
end if
if $feasible =$ `true` **then**
  Solve the following set of equations to find an $\{e_i\}$:

$$\sum_{i=1}^{k} e_i = E;\ \ 1 \le e_i \le x_i\ \ (e_i \in \mathbb{Z},\, i = 1 \dots k) \tag{3.7}$$

end if

So if $\sum x_i \geq E$, then we can find e_i values such that (3.7) is satisfied, and then due to (3.8) we will have a valid assignment, where $U \leq \alpha$.

On the other hand, if $\sum x_i < E$, then we cannot find e_is such that (3.7) is satisfied, and $U \leq \alpha$. To see this, suppose the opposite. Then (3.8) still must be true, and then the supposed $\sum_{i=1}^{k} x_i < E$ contradicts $\sum e_i = E$ in (3.7). □

As calculating x_is according to (3.6) is simple (i.e., $O(1)$) and solving (3.7) can be done in $O(k)$, Algorithm 3.1 has a complexity of $O(k)$.

Next, I utilize a binary search framework to find the minimal α for which there is a feasible solution of algorithm VRA-1N-1D-Fixed-E, given g_is and E. To do so, we need a lower and an upper bound on U. For this, I shall use 1 and G_0, respectively, as given by Lemmas 1 and 4. (Lemma 5 could have been used, too.) To stop the iteration, we also need a lower bound on $|U - U'|$, where U and U' belong to two different link allocations, $\{e_i\}$ and $\{e'_i\}$. The following lemma helps:

Algorithm 3.2 VRA-1N-1D-Bin-Search

Input: k, $\{g_i\}$, E
Output: $\{e_i\}$, U
  $lower \leftarrow 1.0$
  $upper \leftarrow G_0$
  while $upper - lower \geq 1/(G_0E)$ **do**
    $\alpha \leftarrow (upper + lower)/2$
    if VRA-1N-1D-FIXED-E(α, $\{g_i\}$, E) finds a solution **then**
      $upper \leftarrow \alpha$
    else
      $lower \leftarrow \alpha$
    end if
  end while
  $\{e_i\}$ $\leftarrow$VRA-1N-1D-FIXED-E(α, $\{g_i\}$, E) {Lower limits are not valid settings, upper limits are valid. We need a valid setting}
  Calculate U from $\{g_i\}$ and $\{e_i\}$

Lemma 9. *Consider two different link allocations, $\{e_i\}$ and $\{e'_i\}$, both using a total number of E links. For the the associated errors, U and U', if $U \neq U'$ then $|U - U'| \geq 1/(G_0E)$.*

Proof. We can suppose $U > U'$. Then

$$U - U' = \max_{1\leq i\leq k} U_i - \max_{1\leq j\leq k} U'_j \geq \min_{1\leq i,j\leq k,\, U_i > U'_j} U_i - U'_j =$$
$$= \min \frac{G_0 e_i}{E g_i} - \frac{G_0 e'_j}{E g_j} = \min \frac{G_0}{E} \frac{e_i g_j - e'_j g_i}{g_i g_j} \geq \frac{G_0}{E} \frac{1}{G_0^2} = \frac{1}{G_0 E} \ ,$$

since at the last inequality $e_i g_j - e'_j g_i$ is a positive integer and $g_i g_j \leq G_0^2$. □

Note that it is indeed possible, that $U_i = U'_j$. An example for this is the following: $k = 2, g_1 = 1, g_2 = 1, E = 3$. One possible allocation is $e_1 = 2, e_2 = 1$, where $U_1 = 4/3$; another possible allocation is $e'_1 = 1, e'_2 = 2$, where $U'_2 = 4/3$. This has no effect on the lower limit given in Lemma 9: if these Us happen to be the optimal errors for two different allocations, just like in our example, then finding the optimal α results in $\sum x_i > E$, meaning that there are more than one optimal solutions.

The binary search method is described in Algorithm 3.2. This algorithm runs in $\log(G_0^2 E)$ steps, yielding an overall $O(k \log(G_0^2 E))$ polynomial complexity.

Algorithm 3.3 VRA-1N-1D

Input: k, $\{g_i\}$, Q
Output: $\{e_i\}$, $U(E)$, U

```
best_U ← G_0 + 1.0
for E ← k ... Q do
    {current_e_i}, current_U ← VRA-1N-1D-Bin-Search({g_i}, E)
    if current_U < best_U then
        best_U ← current_U
        {best_e_i} ← {e_i}
    end if
    U(E) ← best_U {used in Sec. 3.4}
end for
U ← best_U
{e_i} ← {best_e_i}
```

What remains is to find the value of E that yields the smallest error for the given Q. This is done by the simple Algorithm 3.3. Note that this is theoretically not a polynomial time algorithm as its complexity is $O(Qk\log(G_0^2Q))$, which is not polynomial in the size of Q (i.e., $\log(Q)$). Yet, this complexity is low enough, so the algorithm is easily tractable for the practical use cases.

3.4 Other Problem Formulations

The algorithms for Problem VRA-1N-1D described above can be used with minor modifications to answer a set of related questions. I list three such problems here along with the proposed solutions.

Problem 2, VRA-1N-1D-Link-Min. *Given k, $\{g_i\}$, and $U_{lim} \geq 1$, find $\{e_i\}$ that minimizes the total number of links (E) such that $U \leq U_{lim}$.*

The solution is simple. Consider $U(E)$ generated by Alg. 3.3. For the input Q of Alg. 3.3 use G_0. It is a weakly decreasing function, whose domain is a subset of the positive integers. The solution of the problem is E, where $U(E-1) > U_{\text{lim}}$ and $U(E) \leq U_{\text{lim}}$, or k, if $U(k) \leq U_{\text{lim}}$.

The number of loop cycles until this algorithm finds the suitable E depends on U_{lim} and on the shape of $U(E)$. Nevertheless, Lemma 2 guarantees that in the worst case $E = G_0$ is suitable, thus the complexity is $O(G_0k\log(G_0^3))$.

Algorithm 3.4 N-VRA-1N-1D

Input: $\{k_i\}$, $\{g_{ij}\}$, Q
Output: $\{e_{ij}\}$, $\{Q_i\}$, U
  for $i \leftarrow 1 \dots N$ **do** {initialization}
    $U_i, U_i(E) \leftarrow$ VRA-1N-1D(k_i, $\{g_{ij}\}$, Q)
    $Q_i \leftarrow k_i$
  end for
  while $\sum Q_i < Q$ **do** {greedy algorithm}
    $i = \arg\max U_i$ {if there is more than one such i, select any of them}
    $Q_i \leftarrow Q_i + 1$
  end while
  for $i \leftarrow 1 \dots N$ **do**
    $\{e_{ij}\} \leftarrow$ VRA-1N-1D(k_i, $\{g_{ij}\}$, Q_i)
  end for

The next, somewhat harder problem is about solving several VRA-1N-1D problems concurrently, so that the total number of physical and virtual links of all the nodes together is limited and the goal is to minimize the error over all the links of all the nodes:

Problem 3, Parallel-VRA-1N-1D. *Given k_n ($n = 1 \dots N$), $\{g_{ni}\}$, and Q, find $\{e_{ni}\}$ that minimizes $U_{max} = \max U_n$ such that $\sum_n E_n = \sum_n \sum_i e_{ni} \leq Q$.*

Although in my OSPF-TE example this problem is not directly addressed, in other use cases, just like WCMP, where the total rule number is constrained, this is indeed a valid an important question. A simple greedy algorithm (similar to the one mentioned for Niagara in Sec. 5.1 of [10]) provides an optimal solution, as shown in Algorithm 3.4. We use Q_1, Q_2, $\dots Q_N$ links for each problem, such that $\sum Q_i = Q$.

Theorem 10. *Algorithm 3.4 finds an optimal solution to Problem 3.*

Proof. I prove by contradiction. Suppose Alg. 3.4 results in $\{Q_i\}$ and U, and yet there is $\{R_i\}$ with $U' < U$, such that $\sum R_i \leq \sum Q_i = Q$. Because of the last condition, and because $\exists i : R_i \neq Q_i$ there is at least one j, such that $R_j < Q_j$. Due to the nature of the algorithm and the nonincreasing property of U_j: $U_j(Q_j - 1) \geq U \geq U_j(Q_j)$. On the other hand $U' \geq U_j(R_j) \geq U_j(Q_j - 1) \geq U$, which contradicts $U' < U$. □

The complexity of Algorithm 3.4 is $O(NQk \log(G_0^2 Q) + Q)$, where $k = \max k_i$, $G_0 = \max G_{0i}$. Note that the algorithm could be implemented more efficiently by

calculating $U_i(E)$ only when it is needed for the computation of arg max, resulting in a complexity of $O((Q+N)k\log(G_0^2Q))$.

Finally, here is another variant of Parallel-VRA-1N-1D, minimizing the total number of links over several nodes with a given error limit:

Problem 4, Parallel-VRA-1N-1D-Link-Min. *Given k_n ($n = 1 \dots N$), $\{g_{ni}\}$, and U_{lim}, find $\{e_{ni}\}$ that minimizes $\sum_n E_n = \sum_n \sum_i e_{ni}$ such that $U_{max} = \max U_n \leq U_{lim}$.*

This problem can trivially be decomposed into N independent VRA-1N-1D-Link-Min problems, which can be solved as described above at Problem 2.

3.4.1 Application for WCMP

The concept of Virtual Resource Allocation and the related algorithms presented in this dissertation has high application potential in several parts of communication networks. An example is the enhancement of Weighted Cost Multipathing (WCMP, [5]).

Sections 3.2.1–3.2.3 of paper [5] propose heuristic solutions. My algorithms described in Sec. 3.3 and 3.4 (especially Alg. 3.3 and 3.4) could directly be applied to the WCMP problems as well and they always provide optimal solutions, not just approximations. The computational complexity of these algorithms are:

- Find a solution with minimal error if the maximal number of links/rules is given (Problem 1). WCMP (with our notation): $O(k^2 + k(G_0 - Q))$, VRA: $O(Qk\log(G_0^2Q))$.
- Find a solution with minimal links/rules if the maximal error is given (Problem 2). WCMP: $O(k(G_0 - k))$, VRA: $O(kG_0\log(G_0^3))$.
- Simultaneously minimize the error for N problems using maximum a total number of Q links/rules (Problem 3). WCMP: $O((\sum_N G_{0i} - Q) \cdot \sum_N k_i(G_{0i} - k_i))$, VRA (using $k = \max_i k_i$, $G_0 = \max_i G_{0i}$): $O((Q+N)k\log(G_0^2Q))$.

Although these numbers are not directly comparable, if $G_0 \gg Q, k$, which is a very possible scenario, my algorithms run either similarly to, or faster than, the ones proposed for WCMP, and unlike those mine are guaranteed to provide optimal solutions.

Chapter 4

Peer-Local Optimization

Peer-Local Optimization has been briefly introduced in Sec. 2.3.3 and is described in full detail in this chapter. In this scenario we are given a capacitated network and a set of demands (see Fig. 4.1). The optimization task is to determine for each link a weight and the number of parallel virtual links, which, if fed together to OSPF, will result in minimal MLU (Maximal Link Utilization). In other words, Peer-Local Optimization provides input for OSPF-TE (see Section 2.1) enhanced by VRA.

Just as before, here we have a limit on the number of usable links per node as well, using the *Bounded Total Resources* model, described in Sec. 2.2. In this scenario, however, the limit exists *per node per demand*, in line with the router constraint that a single traffic flow cannot be split onto too many outgoing links. As an example consider the capacitated network in Fig. 4.2(a) with two demands: $A \to E : 30$, $A \to F : 40$. Clearly, for optimal routing all the links have to be fully utilized, requiring a traffic split of $2 : 1$ and $1 : 3$ for the demands at node A. Suppose we are allowed to use at most $Q = 4$ outgoing links per node per demand. We can reach an optimal solution by setting up virtual links as shown in Fig. 4.2(b). Although this way six links are leaving node A, none of the demands are split onto more than four, obeying the limit.

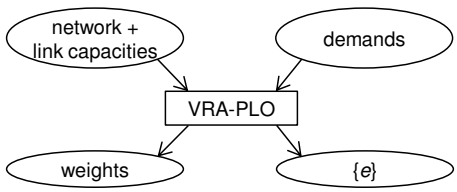
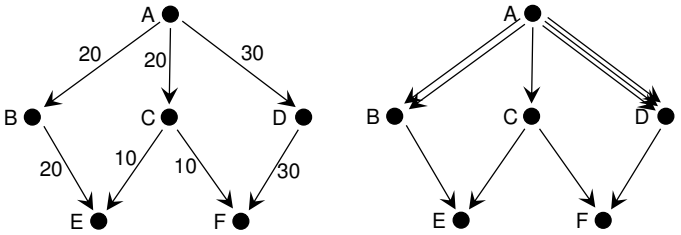
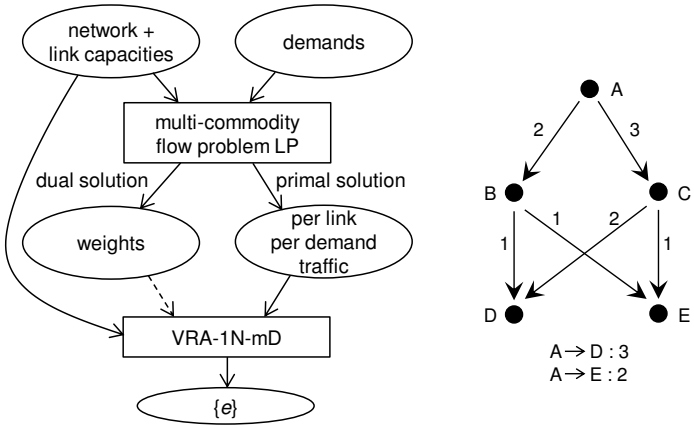


Figure 4.1: Virtual Resource Allocation–Peer-Local Optimization



(a) A network with link capacities (b) Installed virtual links

Figure 4.2: Multi-demand constraint example. Demands: $A \to E : 30$, $A \to F : 40$



(a) Peer-Local Optimization operation (b) VRA-1N-mD example

Figure 4.3: Peer-Local Optimization

As introduced in Sec. 2.3.3, Peer-Local Optimization works at the node level. The overview of its operation is shown in Figure 4.3(a). The first step is the same as for Overlay Optimization: solving a multi-commodity LP with splittable flows, which can be done in polynomial time. The primal solution provides the per link per demand traffic volumes, just as before, but in this case I also extract the dual solution. These contain the link weights (minus a constant r) [42], necessary for OSPF-TE. The next and final step is to solve the VRA-1N-mD problem for each node independently. VRA-1N-mD stands for "Virtual Resource Allocation for One Node and multiple Demands" and provides locally optimal virtual link settings, as detailed later in this chapter.

It has to be noted, however, that the link weights gained this way are not always ready to use by OSPF ECMP. The good news is that according to these weights each

link that has nonzero traffic from a demand will be part of a shortest path between the corresponding source and destination nodes. The bad news, however, is that the opposite direction is not true: there can be links belonging to a shortest path of a demand that have zero traffic from that demand. It has been proven in [54] that by carefully changing the link weights (without modifying the primal solution) this effect can be minimized (which the authors call *minimal shortest path representation*), but the ideal case, where all links of each shortest path have nonzero traffic for the corresponding demand (called *perfect shortest path representation*), is not achievable in general.

Unfortunately, VRA-1N-mD cannot handle a non-perfect shortest path representation, where a link on a shortest path has zero traffic from the corresponding demand. One workaround to this problem is what I followed in my simulation evaluation (Chapter 6), to divert a minimal amount of traffic to these links. This is denoted by the dashed arrow in Fig. 4.3(a).

In VRA-1N-mD we deal with a single node and several demands routed through it. Consider the example shown in Fig. 4.3(b) with the link capacities and the demands. Suppose identical link weights. It is easy to see that at node A the first flow requires a 33%–67% percent split, while the second one needs a 50%–50% divide for optimal network performance. These requirements are contradictory: for the first demand a virtual link along $A - C$ is preferred, while the second is routed best without any virtual link. Clearly, both cannot be done at the same time, meaning that there is no perfect solution. Nevertheless, a setup with the minimal error can indeed be selected. In the remainder of this chapter we examine, how.

4.1 VRA-1N-mD Problem Definition

The formal definition of the VRA-1N-mD problem, using the notations summarized in Table 4.1, is as follows. For a network node A we are given a matrix

$$G = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1k} \\ g_{21} & g_{22} & \cdots & g_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ g_{D1} & g_{D2} & \cdots & g_{Dk} \end{bmatrix}, \tag{4.1}$$

Notation	Description
k	number of outgoing links used
D	number of demands
$G = (g_{ij}) \in \mathbb{Z}^{D \times k}$	desired traffic volume per demand per outgoing link ($g_{ij} \geq 0$)
$\Gamma = (\gamma_{ij}) \in \mathbb{R}^{D \times k}$	row-normalized version of matrix G
$\Sigma = (\sigma_{ij}) \in \{0,1\}^{D \times k}$	$\sigma_{ij} = 0$ if $g_{ij} = 0$, $\sigma_{ij} = 1$ otherwise
$G_i = \sum_{j=1}^{k} g_{ij}$	total traffic volume of demand i
h_{ij}	actual traffic volume per demand per outgoing link
e_1, e_2, …, e_k	number of allocated links (physical and virtual together)
$E_i = \sum_{j=1}^{k} e_j \sigma_{ij}$	total number of parallel links on shortest paths for demand i
$U_{ij} = e_j/(\gamma_{ij} E_i)$	per demand per link error (only where $\gamma_{ij} > 0$)
$U = \max_{\gamma_{ij}>0} U_{ij}$	per node error
$Q \geq E_i \ (\forall i), \ Q \in \mathbb{Z}^+$	upper bound on the number of usable links per demand

Table 4.1: VRA-1N-mD notations

representing how demands $1 \ldots D$ arriving at the node should be split among outgoing links $1 \ldots k$: g_{ij} is the traffic volume that belongs to demand i and should be sent out on link j. g_{ij}s are non-negative integers. We can assume for simplicity that G contains no all-zero rows or columns. Later on I will also use the row-normalized and the signum versions of G:

$$\gamma_{ij} = \frac{g_{ij}}{\sum_{n=1}^{k} g_{in}}; \ \ \sigma_{ij} = \begin{cases} 0, & \text{if } g_{ij} = 0 \\ 1, & \text{if } g_{ij} > 0 \end{cases} .$$

As an example I show these matrices for the simple instance shown in Fig 4.3(b):

$$G = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix}, \ \ \Gamma = \begin{bmatrix} \frac{1}{3} & \frac{2}{3} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}, \ \ \Sigma = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} .$$

We set up e_j number of parallel links (including the physical and virtual ones) for outgoing link j of node A. Our goal is to find $e_1, \ldots, e_k$, such that an error metric is minimized.

An important question is if we allow disabling a physical link instead setting up parallel links, in a similar fashion to Overlay Optimization with Path Exclusion. Formally it would mean setting the number of links to zero on an outgoing link,

i.e., having $e_j = 0$ for some j. Whether or not this can be performed is a network administration issue, and is outside the scope of this work. Nevertheless, due to space constraints, in the rest of this dissertation I suppose that such link disabling is not permitted (i.e., $e_j > 0$).

Let $E_i = \sum_{j=1}^{k} e_j \sigma_{ij}$ be the total number of parallel links on the shortest paths for the ith demand (i.e., we only sum those e_js, where the corresponding g_{ij} is not zero), and $G_i = \sum_{j=1}^{k} g_{ij}$ be the offered load for the demand.

According to ECMP's equal-split rule, the per demand traffic volume on an outgoing link is:

$$h_{ij} = \frac{e_j G_i}{E_i} \ .$$

For the same reasons as described at VRA-1N-1D, here we introduce the *per demand per link error*, defined as the ratio of the transmitted traffic and the offered volume on a given outgoing link j, for a given demand i, but it is only defined if the offered traffic is non-zero:

$$U_{ij} = \frac{h_{ij}}{g_{ij}} = \frac{G_i e_j}{g_{ij} E_i} = \frac{e_j}{\gamma_{ij} E_i} \quad (\forall g_{ij} > 0) \ .$$

The *per node error* (or shorty just *error*) is defined as the maximum of the per link per demand errors:

$$U = \max_{i,j:\gamma_{ij}>0} \frac{e_j}{\gamma_{ij} E_i} \ ,$$

which we aim to minimize in the rest of this chapter.

To complete the previous example, suppose that $e_1 = 2$, $e_2 = 3$, which yields:

$$(U_{ij}) = \begin{bmatrix} \frac{6}{5} & \frac{9}{10} \\ \frac{4}{5} & \frac{6}{5} \end{bmatrix} \ .$$

This results in $U = 6/5$, which can be shown to be actually the minimal error for the given problem.

Like in the Overlay Optimization Section, below I will also study the problem variant with limited number of links used for traffic splitting. I am using the *Bounded Total Resources* model (Sec. 2.2), but here the upper limit is applied on a per demand basis (by requiring $E_i \leq Q$ for all $i = 1 \ldots D$), as detailed for Fig. 4.2.

To sum up, we can formulate the problem as follows.

Problem 5, VRA-1N-mD. *Given k, D, G, and Q, find $\{e_j\}$ such that $E_i \leq Q$ for all $i = 1 \dots D$ and U is minimal.*

As a first approach, however, I will examine a simplified problem variant:

Problem 6, VRA-1N-mD-Unlimited. *Given k, D, and G find $\{e_j\}$ such that U is minimal.*

4.2 Attributes of VRA-1N-mD and VRA-1N-mD-Unlimited

After the definition, first I examine the problems without trying to solve them yet. In this section I present an observation about the completeness of the problems and statements about the possible values of the error metric.

4.2.1 Completeness of the Problems

I present an important finding about the completeness of the problems:

Theorem 11. *Any nonnegative matrix with at least one non-zero element in each row and in each column can be the matrix G of a VRA-1N-mD or a VRA-1N-mD-Unlimited problem, which is the result of Peer-Local Optimization in a suitable network.*

Proof. For a given matrix G, I construct a network where after the Peer-Local Optimization process (see Fig. 4.3(a)) at a certain node the required splitting ratios are exactly as given in G.

Let G be in the form of (4.1). The corresponding capacitated network is shown in Fig. 4.4. There are altogether d demands in the system, the ith going from S_i to D_i, i.e., there is no traffic between S_i and D_j if $i \neq j$. For each demand i the desired splitting ratio is given by g_{ij} $(j = 1 \dots k)$. Note that if for some i, j $g_{ij} = 0$ then the corresponding link can be omitted from the network.

Trivially, at node A Peer-Local Optimization will result in a VRA-1N-mD/VRA-1N-mD-Unlimited problem with the given matrix G. □

This result is significant as it allows us to focus on the matrices only, instead of the possibly much more complex networks.

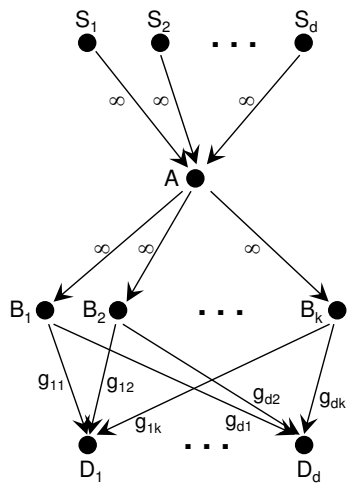


Figure 4.4: Capacitated network corresponding to a given matrix G

4.2.2 Bounds on the Error

Let us now see the theoretical bounds on the error of VRA-1N-mD and VRA-1N-mD-Unlimited. The first statement is analogous to Lemma 1, giving a lower limit:

Lemma 12. $U \geq 1$.

Proof. I will prove a stronger claim: in every row of G there is an element for which the per link per demand error is at least one. The proof is by contradiction: suppose this is not the case for the ith row. This means that for all $j = 1 \dots k$, where $\sigma_{ij} = 1$: $U_{ij} = e_j/(\gamma_{ij}E_i) < 1$, that is $e_j/E_i < \gamma_{ij}$. Summing these yields

$$\sum_{j=1\dots k:\sigma_{ij}=1} \frac{e_j}{E_i} < \sum_{j=1\dots k:\sigma_{ij}=1} \gamma_{ij}$$

meaning that $E_i/E_i < 1$, which is clearly a contradiction. □

Note that $U = 1$ is attainable for some matrices G by properly selecting $\{e_j\}$, but for other Gs there is no such $\{e_j\}$ setting. Examples and detailed discussion on this topic is available in Sec. 4.3.1.

Next, I give an upper bound on the error:

Lemma 13.

$$U \leq \frac{1}{\min_{i,j:\gamma_{ij}>0} \gamma_{ij}}$$

Proof.

$$U_{ij} = \frac{e_j}{E_i \gamma_{ij}} = \frac{e_j}{(\sum_{n=1}^{k} e_n \sigma_{in})\gamma_{ij}} \leq \frac{1}{\gamma_{ij}} \ ,$$

from which the theorem follows. □

Lemma 14. *There is no universal (G-independent) upper bound on the error.*

Proof. For any $M > 0$ integer the following matrix G results in $U = M$:

$$G = \begin{bmatrix} 1 & 2M-1 \\ 2M-1 & 1 \end{bmatrix}, \ \Gamma = \begin{bmatrix} \frac{1}{2M} & \frac{2M-1}{2M} \\ \frac{2M-1}{2M} & \frac{1}{2M} \end{bmatrix} \ .$$

Easily, the optimal solution is $e_1 = e_2 = 1$, for which $U = U_{11} = U_{22} = \frac{1/2}{1/(2M)} = M$. □

As in this optimal solution the total number of virtual links is zero, this theorem is valid even if the number of virtual links is not limited in any way, i.e., it is valid for both the VRA-1N-mD and the VRA-1N-mD-Unlimited problems. Likewise, the other two bounds on the error given above are also valid for both problem variants.

4.3 Unlimited Number of Links

After exploring the theoretical limits, I look for an optimal link allocation. I start with a simplified version of the problem, where the maximum number of link constraint is relaxed, i.e., we allow unlimited number of parallel links to be used simultaneously, as defined in Problem 6, VRA-1N-mD-Unlimited. Notice that in this case the problem is practically defined by the matrix G itself. Accordingly, I will use the problem instance and the matrix interchangeably in this section.

4.3.1 Consistency of a VRA-1N-mD-Unlimited Problem

Definition 1. A matrix G is *consistent* if and only if $U = 1$ can be achieved with a suitable $\{e_j\}$, allowing unlimited number of parallel links.

Here are some examples on consistency:

Example 1. The following G is consistent, and $U = 1$ for the given $\{e_j\}$:

$$G = \begin{bmatrix} 1 & 2 & 4 & 0 & 0 \\ 0 & 3 & 6 & 1 & 0 \\ 0 & 0 & 30 & 5 & 1 \end{bmatrix}$$
$$e = \begin{bmatrix} 15 & 30 & 60 & 10 & 2 \end{bmatrix}$$

Example 2. This matrix is also consistent:

$$G = \begin{bmatrix} 0 & 1 & 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 3 & 7 \\ 5 & 3 & 0 & 6 & 0 & 0 \end{bmatrix}$$
$$e = \begin{bmatrix} 5 & 3 & 14 & 6 & 3 & 7 \end{bmatrix}$$

Example 3. This matrix is inconsistent:

$$G = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix}$$

Example 4. This matrix is also inconsistent:

$$G = \begin{bmatrix} 1 & 2 & 4 & 0 & 0 \\ 0 & 3 & 6 & 1 & 0 \\ 0 & 0 & 30 & 5 & 1 \\ 2 & 0 & 0 & 7 & 0 \end{bmatrix}$$

An Algorithm for Deciding Consistency. Algorithm 4.1 decides whether a given G is consistent, and if it is, it also supplies an $\{e_j\}$ for which $U = 1$. Note that the algorithm could be extended by dividing the resulting vector e by the GCD of its coordinates to have a "nicer" result, but as we supposed unlimited total number of available links, this is not necessary.

Algorithm 4.1 Deciding Consistency of a Matrix G

Input: G
Output: $consitent, \{e_j\}$

```
1: function VectorMergeAndCheck(v1,v2) {Check for all the coordinates of v1 and v2 if they are either equal or one of them is 0. If true, merge the non-zero elements of v2 into v1. Requires identical vector sizes.}
2:     for all coordinates i do
3:         if v1[i] = 0 then
4:             v1[i] ← v2[i]
5:         else if v2[i] ≠ 0 and v1[i] ≠ v2[i] then
6:             return false
7:         end if
8:     end for
9:     return true
10: end function

11: function IsConsistent(G) {Decides whether G is consistent. If true, also returns an {e_j} for which U = 1}
12:     if G is empty or G has only one row then
13:         e ← G
14:         return true, e
15:     end if
16:     e ← 1st row of G
17:     remove 1st row from G
18:     while G is not empty do
19:         found ← false
20:         for all rows i in G do
21:             for all coordinates j of the ith row of G do
22:                 if G_ij ≠ 0 and e_j ≠ 0 then
23:                     found ← true
24:                     exit the innermost for all loop
25:                 end if
26:             end for
27:             if found = true then
28:                 exit the innermost for all loop
29:             end if
30:         end for
31:         if found = true then
32:             f ← ith row of G
33:             remove ith row from G
```

```
34:             p ← e_j
35:             q ← f_j
36:             d ← GCD(p, q)
37:             p ← p/d
38:             q ← q/d
39:             e ← q · e {multiply a vector by a scalar}
40:             f ← p · f {multiply a vector by a scalar}
41:             if VectorMergeAndCheck(e, f)=false then
42:                 return false
43:             end if
44:         else{the remainder of G is independent of the already processed one}
45:             cons, new_e ←IsConsistent(G)
46:             if cons = false then
47:                 return false
48:             end if
49:             VectorMergeAndCheck(e, new_e)
50:             return true, e {exiting the main loop}
51:         end if
52:     end while
53:     return true, e
54: end function
```

The idea of Algorithm 4.1 is to build the vector e by iterating through the rows of matrix G. We start with e being the first row of G (lines 16-17). Then, we search for a row in G, which has at least one non-zero element in a column, where e is also nonzero (lines 18-30). If found, then we remove it from G and check if that row is "compatible" with e, in the sense, that the ratio of the coordinates, where each vector is non-zero, is identical. If they are not compatible, then G is inconsistent (lines 31-43). If they are, then we extend e with the non-zero elements of the given row of G (lines 41, 1-10). We repeat this until there are no more rows of G that have non-zero elements, where e is non-zero. We then re-run the whole algorithm on the remainder of G and merge the resulting e with the current one, if possible (lines 44-51).

Complexity of Algorithm 4.1. Examining the loops, it can be seen that at most $d^2k + 4dk$ steps are required, so the complexity of the algorithm is $O(d^2k)$.

LP for Deciding Consistency. Linear Program 4.1 also solves this problem in polynomial time. There is a feasible solution for the LP if and only if the matrix is

LP 4.1 Deciding Consistency

$$
\begin{aligned}
\textit{variables: } & e_j && (e_j \geq \epsilon,\ e_j \in \mathbb{R},\ j = 1 \dots k) \\
\textit{constants: } & g_{ij} && (g_{ij} \geq 0,\ g_{ij} \in \mathbb{Z},\ i = 1 \dots d,\ j = 1 \dots k) \\
& \epsilon && (\epsilon > 0) \\
\textit{objective: } & - && \\
\textit{constraints: } & e_{j_1} g_{ij_2} = e_{j_2} g_{ij_1} && (\forall i = 1 \dots d,\ j_1, j_2 = 1 \dots k :\ g_{ij_1} g_{ij_2} \neq 0)
\end{aligned}
$$

consistent. The role of ϵ is only to provide $e_j > 0$, as the resulting vector e could be multiplied by any constant and still be valid. A disadvantage of this LP-based approach is that it returns real numbers as the solution, which are not trivial to map to integer number of links.

The same problem could also be solved as an Integer LP (requiring $e_j \in \mathbb{Z}^+$), but it could result in non-polynomial running time. An intermediate solution would be to use a rational solver, but in that case having a polynomially limited running time is also not straightforward.

Note that LP 4.1 might look at first sight as a homogeneous system of linear equations, but as the variables are required to be positive, the classical solutions, like the Gaussian elimination, are not applicable here.

Consistency and Errors. Finally, I present a lemma on the nature of the errors that are related to a consistent matrix G.

Lemma 15. *If a matrix G is consistent then for an $\{e_j\}$ for which $U = 1$: $U_{ij} = 1$ $\forall i, j : \sigma_{ij} = 1$.*

Proof. By definition, for this $\{e_j\}$ setting $U_{ij} \leq 1$ $(\forall i, j : \sigma_{ij} = 1)$. We have to show that $U_{ij} = 1$ holds. Suppose the opposite: $\exists a, b : \sigma_{ab} = 1$, $U_{ab} < 1$. Then $U_{ab} = e_b/(\gamma_{ab} E_a) < 1$, i.e.,

$$e_b/E_a < \gamma_{ab} \ . \tag{4.2}$$

Similarly for the rest of the matrix: $U_{ij} \leq 1$, which means

$$e_j/E_i \leq \gamma_{ij} \quad \forall ij : ij \neq ab, \sigma_{ij} = 1. \tag{4.3}$$

Now let us sum (4.2) and (4.3) over the ath row:

$$\sum_{j=1\ldots k:\sigma_{aj}=1} \frac{e_j}{E_a} < \sum_{j=1\ldots k:\sigma_{aj}=1} \gamma_{aj}$$

meaning that $E_a/E_a < 1$, which is a contradiction. □

4.3.2 Notes on the Types of the Solution

I now show two, somewhat surprising observations about the types of the optimal solution. As the complete proofs are quite lengthy, they are moved to Appendix C, and here I only highlight their most important steps.

For simplicity, in the remainder of Section 4.3 I will use the normalized version of the link number e_j, denoted by f_j to avoid confusion:

$$f_j = \frac{e_j}{\sum_{i=1}^{k} e_i} \quad \Rightarrow \quad f_j \in \mathbb{R}^+, \ \sum_{j=1}^{k} f_j = 1. \tag{4.4}$$

Theorem 16. *There is at least one VRA-1N-mD-Unlimited problem, where matrix G contains integers only but the single optimal solution contains only irrational numbers as f_js.*

Sketch of Proof. Consider the following input matrix:

$$G = \begin{bmatrix} 2 & 1 & 0 \\ 2 & 2 & 1 \end{bmatrix} .$$

I prove in the Appendix C.1 that for this matrix for any optimal solution $U_{12} = U_{21} = U_{23}$ and $U_{11} < U_{21}, U_{22} < U_{12}$. Furthermore, I also show that there is a single optimal solution (f_1, f_2, f_3) in this case, for which expanding and solving the $U_{12} = U_{21} = U_{23}$ system of equations and using that $f_1, f_2, f_3 > 0$ and that $f_1 + f_2 + f_3 = 1$, we get:

$$f_1 = \frac{2}{5}(7 - \sqrt{34}),\ f_2 = \frac{1}{5}(-16 + 3\sqrt{34}),\ f_3 = \frac{1}{5}(7 - \sqrt{34}) \ . \qquad \square$$

This theorem has an important consequence:

Corollary 17. *The optimal settings for the VRA-1N-mD-Unlimited problem cannot always be reached using finite total number of links.*

The next theorem is even less straightforward than Theorem 16.

Theorem 18. *There is at least one VRA-1N-mD-Unlimited problem, whose only optimal solution contains at least one f_j that cannot be written in a finite form using integer constants and the usual $+$, $-$, $\cdot$, $/$ and the nth root ($n \in \mathbb{Z}^+$) operators only.*

Sketch of Proof. Consider the following matrix:

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 6 & 1 & 0 & 0 & 0 & 0 \\ 6 & 6 & 1 & 0 & 0 & 0 \\ 6 & 6 & 6 & 1 & 0 & 0 \\ 6 & 6 & 6 & 6 & 1 & 0 \\ 6 & 6 & 6 & 6 & 6 & 1 \end{bmatrix} .$$

I prove in Appendix C.2 that for this case the maximal error terms for the optimal solution are $U_{61} = U_{22} = U_{33} = U_{44} = U_{55} = U_{66}$. This leads to a fairly simple system of equations on $f_1, \ldots, f_6$. From these equations $f_2, \ldots, f_6$ can be eliminated, and what remains is a polynomial of f_1:

$$923\,521 f_1^5 - 16\,980\,870 f_1^4 + 118\,664\,280 f_1^3 - \\ - 390\,577\,680 f_1^2 + 934\,673\,904 f_1 - 336\,117\,600 = 0 \ . \quad (4.5)$$

I used the mathematical software Maple [55] to show that this polynomial equation has got a single real root only (and four complex ones). According to Galois theory [56], a polynomial equation can be solved by radicals[1] if and only if its Galois group is a solvable group. Using Maple I found that the Galois group of the polynomial given in (4.5) is the symmetric group S_5. This group, consisting of 120 elements, is not solvable, meaning that (4.5) cannot be solved by radicals. □

A solution of the VRA-1N-mD-Unlimited problem is given as $f_1, f_2, \ldots, f_k$, where $f_j \in \mathbb{R}^+$. We expect real constants f_j to be presented by an algorithm that solves the problem in some sort of closed form, but "closed form" can be defined in several

[1] i.e., having a solution that can be written in a finite form using integer constants and the $+$, $-$, $\cdot$, $/$ and the nth root ($n \in \mathbb{Z}^+$) operators only

ways. For now I require f_js to be given by finite expressions that consist of integer constants and the usual $+$, $-$, $\cdot$, / and the nth root ($n \in \mathbb{Z}^+$) operators only.

Corollary 19. *No algorithm can give an optimal solution to VRA-1N-mD-Unlimited in finite number of steps; even if the number of steps may depend on the actual problem.*

The idea behind this corollary is that an optimal solution cannot be computed in finite number of steps if for some inputs it cannot be written in a closed form, since writing the output is part of the solution.

4.3.3 An LP-based Iterative Solution

From the previous subsection we know that finding the exact solution is infeasible. I can, nevertheless, search for an approximation of the optimal solution.

As a first step, I set up an LP that computes $\{f_j\}$ while keeping the per node error under a given constant α. Naturally, for too small αs the LP will not have a solution.

I do this by enforcing each per demand per link error term to be less than or equal to α:

$$\frac{f_j}{\gamma_{ij} \sum_{n=1}^{k} \sigma_{in} f_n} \leq \alpha \ ,$$

which can be rearranged as:

$$0 \leq \sum_{n=1}^{k} \sigma_{in} f_n - \frac{f_j}{\gamma_{ij} \alpha} \ ,$$

which leads to LP 4.2.

Note that in this LP I have provisionally relaxed the constraint $\sum_j f_j = 1$ (see (4.4)), and introduced variables $\hat{f}_j$ to avoid confusion. The reason for this change is that I wanted to enforce $f_j > 0$, meaning that a link cannot be disabled. In an LP, however, only "greater than or equal to" type constraints can be used and no "strictly greater than" types. So I have released $\sum f_j = 1$ and added (4.6) requiring $\hat{f}_j \geq 1$, which does the trick. On the other hand, if an $\{\hat{f}_j\}$ satisfies (4.8) then each $\{c\hat{f}_j\}$ does so as well, where $c > 0$. Therefore the objective function (4.7) has been added; mainly for aesthetic reasons, as for the unlimited links case any $\hat{f}_j$s satisfying (4.8)

LP 4.2 VRA-1N-mD-Unlimited, Given α

$$
\begin{aligned}
&\textit{indices:} && i = 1 \dots D \\
& && j = 1 \dots k \\
&\textit{constants:} && \alpha \quad (\alpha \geq 1,\ \ \alpha \in \mathbb{R}) \\
& && \gamma_{ij} \quad (\gamma_{ij} \in \mathbb{Q},\ \ \gamma_{ij} \geq 0,\ \ \forall i : \sum_{n=1}^{k} \gamma_{in} = 1) \\
& && \sigma_{ij} \quad = \operatorname{sgn}(\gamma_{ij}) \\
&\textit{variables:} && \hat{f}_j \quad (\hat{f}_j \geq 1,\ \ \hat{f}_j \in \mathbb{R}) && (4.6) \\
&\textit{objective:} && \text{minimize} \sum_{j=1}^{k} \hat{f}_j && (4.7) \\
&\textit{constraints:} && 0 \leq \sum_{n=1}^{k} \sigma_{in} \hat{f}_n - \frac{\hat{f}_j}{\gamma_{ij}\alpha}, \quad \forall i, j : \gamma_{ij} > 0 && (4.8)
\end{aligned}
$$

are equally good. Solving LP 4.2 may return large $\hat{f}_j$s, but they can be normalized to one by dividing by their sum, and thereby acquiring f_js that conform to (4.4).

Now we can use binary search to find the smallest α for which LP 4.2 is solvable, as described in Algorithm 4.2. Note that ϵ_U describes how close we want to get to the optimal error: providing it is necessary due to the consequences of Corollary 17. A typical value could be $\epsilon_U = 10^{-8}$. Nevertheless, this way we can approximate the optimal solution arbitrarily close.

Complexity of Algorithm 4.2. This algorithm is based on a linear program that contains no integer variables, hence it can be solved in polynomial time. The question is, how many times is this LP run? Algorithm 4.2 does a binary search on the $[1, 1/\min_{i,j} \gamma_{ij}]$ interval, until it reaches an optimal solution with an error less than ϵ_U. For this $\log_2(1/(\epsilon_U \min_{i,j} \gamma_{ij}))$ steps are enough, meaning that Algorithm 4.2 runs in polynomial time.

Summary. I have given an iterative algorithm for the VRA-1N-mD-Unlimited problem, which quickly converges to an optimal setting. According to Corollary 19 significantly better solution cannot be given.

Algorithm 4.2 VRA-1N-mD-Unlimited

Input: G, ϵ_U
Output: $\{f_j\}$
  $lower \leftarrow 1.0$ {See Lemma 12}
  $upper \leftarrow 1/(\min_{i,j} \gamma_{ij})$ {See Lemma 13}
  while $upper - lower \geq \epsilon_U$ **do**
    $\alpha \leftarrow (upper + lower)/2$
    if SOLVE_LP 4.2(α, G) finds a solution **then**
      $upper \leftarrow \alpha$
    else
      $lower \leftarrow \alpha$
    end if
  end while
  $f \leftarrow$ SOLVE_LP 4.2($upper$, G) {Lower limits are not valid settings, upper limits are valid. We need a valid setting}

Algorithm 4.3 VRA-1N-mD-Unlimited, Positive Matrix G

Input: G ($g_{ij} \neq 0$)
Output: $\{f_j\}$
1: $\gamma_{ij} \leftarrow g_{ij} / \sum_{n=1}^{k} g_{in}$ (repeated for $i = 1 \ldots d$, $j = 1 \ldots k$)
2: $\gamma'_j \leftarrow \min_{i=1 \ldots d} \gamma_{ij}$ (repeated for $j = 1 \ldots k$)
3: $f_j = \frac{\gamma'_j}{\sum_{n=1}^{k} \gamma'_n}$ (repeated for $j = 1 \ldots k$)

4.3.4 A Special Case: Positive Matrix G

In this subsection I examine a special case of the VRA-1N-mD-Unlimited problem, in which matrix G contains only positive elements. For this case Algorithm 4.3 will provide a quicker and simpler solution than Algorithm 4.2. Furthermore, this solution is exact and not just approximative.

Theorem 20. *For a VRA-1N-mD-Unlimited problem in which matrix G is positive, Algorithm 4.3 provides the single optimal solution.*

Proof. The error U to be minimized is the maximal element in

$$(U_{ij}) = \begin{bmatrix} \frac{f_1}{\gamma_{11}} & \frac{f_2}{\gamma_{12}} & \cdots & \frac{f_k}{\gamma_{1k}} \\ \frac{f_1}{\gamma_{21}} & \frac{f_2}{\gamma_{22}} & \cdots & \frac{f_k}{\gamma_{2k}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{f_1}{\gamma_{d1}} & \frac{f_2}{\gamma_{d2}} & \cdots & \frac{f_k}{\gamma_{dk}} \end{bmatrix} .$$

Let us examine this matrix column by column. Clearly, the identity of the maximal element in each column j is independent of f_j and this value is $\max_{i=1\ldots d} f_j/\gamma_{ij} = f_j/\gamma'_j$ (see Step 2 of Alg. 4.3). Thus the solution of the original problem is reduced to minimizing the maximal element in

$$(U'_j) = \begin{bmatrix} \frac{f_1}{\gamma'_1} & \frac{f_2}{\gamma'_2} & \ldots & \frac{f_k}{\gamma'_k} \end{bmatrix} .$$

Now I show that the single optimal solution is $\{f_j\}$ as given in Step 3 of Alg. 4.3. In this case for each $j = 1 \ldots k$: $U'_j = f_j/\gamma'_j = 1/\sum \gamma'_n = U$. For any other $\{\tilde{f}_j\}$ setting $(\sum \tilde{f}_j = 1)$ there trivially exists a j for which $\tilde{f}_j > f_j$ and then $\tilde{U}'_j > U'_j = U$. □

Note that $\sum_{j=1}^{k} \gamma'_j \leq 1$: by definition $\gamma'_j \leq \gamma_{1j}$ $(j = 1 \ldots k)$ and $\sum_{j=1}^{k} \gamma_{1j} = 1$, so $\sum \gamma'_j \leq 1$. Consequently, $U = 1/\sum \gamma'_j \geq 1$, in accordance with my previous results. It also follows that $U = 1$ if and only if G is consistent, again, as expected.

Easily, Algorithm 4.3 runs in $O(kd)$ steps. As reading the input itself requires kd steps, this is very effective.

4.4 Limited Number of Links

Now I give solutions to the original VRA-1N-mD problem, which has limit on the outgoing links used in parallel. We are searching for positive integers $\{e_j\}$ that minimize the error, on the conditions $E_i \leq Q$ $(i = 1 \ldots D)$.

Clearly all link allocations are valid for which $e_j > 0$ $\forall j$ and $\sum_{j=1}^{k} e_j \leq Q$. Also, as shown at the beginning of this chapter, there could be valid allocations for which $E_i \leq Q$ $\forall i$ holds, but $\sum_j e_j \leq Q$ does not hold. This means that according to Lemma 7 there are at least $\binom{Q}{k}$ valid link allocations, which calls for a more efficient solution than the simple exhaustive search.

4.4.1 An ILP-based Iterative Solution

Linear Program 4.2 can be easily modified to LP 4.3 to suit the "limited total number of links" case. Note that the objective function might be omitted here as well, it only forces the LP solver to select a solution with the fewest total number of links. Just as

LP 4.3 VRA-1N-mD, Given α

$$
\begin{aligned}
&\textit{indices:} && i = 1 \dots D \\
& && j = 1 \dots k \\
&\textit{constants:} && \alpha \quad (\alpha \geq 1,\ \ \alpha \in \mathbb{R}) \\
& && \gamma_{ij} \quad (\gamma_{ij} \in \mathbb{Q},\ \ \gamma_{ij} \geq 0,\ \ \forall i : \sum_{j=1}^{k} \gamma_{ij} = 1) \\
& && \sigma_{ij} \quad = \operatorname{sgn}(\gamma_{ij}) \\
& && Q \quad (Q \in \mathbb{Z}^+) \qquad (4.9) \\
&\textit{variables:} && e_j \quad (e_j \in \mathbb{Z}^+) \qquad (4.10) \\
&\textit{objective:} && \text{minimize} \sum_{j=1}^{k} e_j \\
&\textit{constraints:} && 0 \leq \sum_{n=1}^{k} \sigma_{in} f_n - \frac{f_j}{\gamma_{ij}\alpha}, \quad \forall i, j : \gamma_{ij} > 0 \\
& && \sum_{j=1}^{k} \sigma_{ij} e_j \leq Q, \quad \forall i \qquad (4.11)
\end{aligned}
$$

Algorithm 4.4 VRA-1N-mD by ILP

Input: G, Q, ϵ_U

Output: $\{e_j\}$

This is the same as Algorithm 4.2, but with solving LP 4.3 instead of LP 4.2.

before, LP 4.3 works for a fixed α, but using it in Algorithm 4.4 (see above) results in an arbitrarily good approximation for the VRA-1N-mD problem.

Note that LP 4.3 is actually an Integer Linear Program (ILP), and thus although it is a simple and elegant way to find an optimal link setting, polynomial running time is not guaranteed anymore.

Algorithm 4.4 stops if the difference of the errors in the last two iteration steps is less than an input constant, ϵ_U, hence it is an approximation only in general. In the "unlimited number of links" case there is no theoretical lower bound on the difference of the errors of two different $\{e_j\}$ settings, therefore a sufficiently low number was recommended, such as $\epsilon_U = 10^{-8}$. In the "limited total number of links" case, however, due to the finite number of possible allocations, it is possible to give an absolute lower bound on the difference of the errors for two link allocation settings and thus to find an optimal solution. The following lemma gives such a lower bound.

Lemma 21. *For a VRA-1N-mD problem and for any two valid link allocation settings* $\{e_j\}$ *and* $\{e'_j\}$ *the following holds:*

$$\Delta U = |U - U'| \geq \frac{1}{(\max_{i,j} g_{ij})^2 Q^2} \ .$$

Proof. We can suppose $U > U'$. Then

$$\Delta U = U - U' = \max_{i,j:g_{ij}>0} U_{ij} - \max_{x,y:g_{xy}>0} U'_{xy} \geq$$
$$\geq \min_{i,j,x,y:g_{ij}>0,g_{xy}>0,U_{ij}>U'_{xy}} \{U_{ij} - U'_{xy}\} = \min \left\{ \frac{G_i e_j}{g_{ij} E_i} - \frac{G_x e'_y}{g_{xy} E'_x} \right\} =$$
$$= \min \left\{ \frac{G_i E'_x e_j g_{xy} - G_x E_i e'_y g_{ij}}{g_{ij} g_{xy} E_i E'_x} \right\} \geq \frac{1}{(\max_{i,j} g_{ij})^2 Q^2} \ .$$

The last inequality is true because the numerator of the left-hand side fraction is a positive integer, thus it is greater than or equal to one. For the denominator E_i, $E'_x \leq Q$ by definition. □

This means that for the "limited total number of links" case the recommended constant for Algorithm 4.4 is

$$\epsilon_U = \frac{1}{(\max_{i,j} g_{ij})^2 Q^2} \ .$$

4.4.2 A Direct ILP Formulation

Theorems 16 and 18 (page 42) are valid for the "unlimited total number of links" case only. This means that instead of the iterative solution shown above, theoretically a single ILP with e_js and the error (α) as variables might be set up. Actually this really is the case, and LP 4.4 is such an ILP formulation.

A few notes on this ILP. First, variable y_{in} equals one if $E_i = n$, and zero otherwise. This is enforced by the first two constraints. Therefore (4.13) is only effective if $y_{in} = 1$, i.e., $E_i = n$. In this case the constraint is: $e_j/\gamma_{ij} \leq \alpha E_i$, which is the usual $U_{ij} = e_j/(\gamma_{ij} E_i) \leq \alpha$ error condition.

In the objective function (4.12) the main goal is to minimize the error, but if there are several equally good solutions then we try to select one with the smallest total number of links.

LP 4.4 VRA-1N-mD, Direct, Optimal Solution

$$
\begin{aligned}
&\textit{indices:} && i = 1 \dots D \\
& && j = 1 \dots k \\
& && n = 1 \dots Q \\
&\textit{variables:} && e_j \quad (e_j \in \mathbb{Z}^+) \\
& && \alpha \quad (\alpha \geq 1,\ \alpha \in \mathbb{R}) \\
& && y_{in} \quad (y_{in} \in \{0, 1\}) \\
&\textit{constants:} && \gamma_{ij} \quad (\gamma_{ij} \geq 0,\ \gamma_{ij} \in \mathbb{Q},\ \forall i : \sum_{j=1}^{k} \gamma_{ij} = 1) \\
& && \sigma_{ij} \quad = \operatorname{sgn}(\gamma_{ij}) \\
& && Q \quad (Q > 0,\ Q \in \mathbb{Z}) \\
& && r \quad \text{(a small number)} \\
& && M \quad \text{(a large number)} \\
&\textit{objective:} && \text{minimize } \alpha + r \sum_{j=1}^{k} e_j \qquad (4.12) \\
&\textit{constraints:} && \sum_{n=1}^{Q} y_{in} = 1 \\
& && \sum_{j=1}^{k} \sigma_{ij} e_j = \sum_{n=1}^{Q} y_{in} n \\
& && \frac{e_j}{\gamma_{ij}} \leq \alpha n + M(1 - y_{in}), \quad \forall i, j : \gamma_{ij} > 0 \qquad (4.13) \\
& && \sum_{j=1}^{k} \sigma_{ij} e_j \leq Q, \quad \forall i
\end{aligned}
$$

Next, some words on choosing r and M. Constant r should be small enough so that the ILP solver finds the minimal α. In theory r can be arbitrary small, and the smaller the better, but in practice a too small r may cause rounding errors. According to (4.12) we need $\alpha \gg r \sum_{j=1}^{k} e_j$, thus $r \ll \alpha / \sum_{j=1}^{k} e_j$. As in practice $\alpha \approx 1 \dots 10$ and $\sum_{j=1}^{k} e_j \approx Q$, $r \ll 1/Q$ is a good guess. For $Q = 16$ $r = 10^{-5}$ was proven to be a good choice. Using $r = 10^{-10}$, however, produced incorrect results in practice (using GLPK [57] with C++ and Lemon [58]) due to arithmetic underflow.

Similarly, theoretically M can be arbitrary large, in practice, however, a too large M might cause errors because of the finite number representation. Here due to (4.13) the rule of thumb is $M > e_j/\gamma_{ij}$. Using $M > Q/\min \gamma_{ij}$ is satisfactory, but it depends on G. Nevertheless, if $Q = 16$ and $\gamma_{ij} > 0.02$, then $M = 1000$ is suitable. Practically,

Algorithm 4.5 VRA-1N-mD Heuristic
Input: G, Q, ϵ_U
Output: $\{e_j\}$
$\{f_j\} \leftarrow$ Algorithm 4.2(G, ϵ_U)
$\{e_j\} \leftarrow$ Algorithm 3.3*($\{f_j\}$, Q)

for $Q = 16$ and γ_{ij}s inferred from real network capacities, $M = 1000$ was proven to be a good choice, while $M = 10^5$ already resulted in incorrect behavior.

Finally, let us note that although a single ILP may look more appealing than an iteratively used one, in practice the latter (i.e., Alg. 4.4) was proven to be much faster, especially for large Qs. This is most probably due to the large number of auxiliary integer variables (y_{in}) in LP 4.4.

4.4.3 A Heuristic Solution

I provide a suboptimal, but fast heuristic as an alternative to the previously described ILP-based solutions. The idea is to somehow represent matrix G of a VRA-1N-mD problem with a vector of length k, and treat that as vector g of a VRA-1N-1D problem. The latter can be solved quickly, as described in the previous chapter. What remains is to find an efficient method for the matrix G to vector g mapping. Solving the VRA-1N-mD-Unlimited problem does exactly this: the resulting f_js can simply be treated as vector g. This process is summarized in Algorithm 4.5. Certainly some information is lost during the matrix to vector conversion, which can lead to suboptimal results. That is, nevertheless, acceptable as this method is a heuristic only.

A note on Algorithm 3.3* (referred at Alg. 4.5): Algorithms 3.1, 3.2, and 3.3 require integer g_is as input, but internally they only use their normalized form: g_i/G_0, where $G_0 = \sum_{j=1}^{k} g_j$. Sightly modified versions of these algorithms, marked with an asterisk, take g_i/G_0 as input. In Algorithm 4.5 we provide $\{f_j\}$ as this input. This also affects the complexity: instead of $O(Qk \log(G_0^2 E))$ here, by applying Lemma 13, we have $O(Qk \log(1/(\epsilon_U \min_i g_i/G_0)))$.

Observe that in this heuristic we effectively apply the constraint $\sum_{j=1}^{k} e_j \leq Q$ instead of the less restrictive $E_i \leq Q$, which might yield suboptimal results. This can be perceived as the price to pay for the shorter running times.

Complexity of Algorithm 4.5. The first step runs in polynomial time, as it is a (non-integer) linear program embedded in a binary search. The second step runs in $O(Qk \log(1/(\epsilon_U \min f_j)))$. Although this is only pseudo-polynomial in k and Q, and I have not established a lower bound on f_j, for practical problem instances I found Algorithm 4.5 to be much faster than Algorithm 4.4 [59].

Chapter 5

Peer-Global Optimization

I have briefly introduced *Peer-Global Optimization* at Section 2.3.4, and in this chapter I describe it in detail.

The optimization task is the same as for Peer-Local Optimization: given a capacitated network and a set of demands determine for each link a weight and the number of parallel links that together minimize the maximal link utilization (see Fig. 5.1). This time, however, we solve the problem concurrently for all the nodes in the network so that we can reach a theoretically optimal virtual resource allocation.

5.1 VRA-PGO Problem Definition

Let us start with the formal definition of the Virtual Resource Allocation–Peer-Global Optimization problem, using the notations of Table 5.1.

We are given a directed graph (V, F) representing a *network*, with *capacities* c_l for each link l and a *set of demands*, each given by its originating and destination nodes, and the offered traffic volume: $\{O_d, D_d, G_d\}_{d=1}^{D}$. The *maximal number of virtual links* that can be applied at a node (R) is given as well.

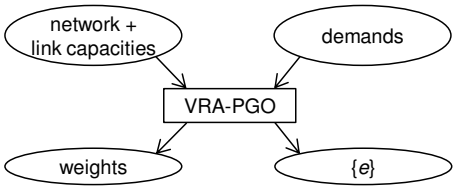


Figure 5.1: Virtual Resource Allocation–Peer-Global Optimization

Notation	Description
V	set of vertices (nodes) in the network
F	set of edges (physical links) in the network
S_n	set of (physical) links originating at node $n \in V$
$\|S_n\|$	number of (physical) links originating at node n
$c_l \in \mathbb{Q}^+$	capacity of link $l \in F$
$w_l \in \mathbb{Q}^+$	weight of link l
$h_l \geq 0$ $(h_l \in \mathbb{Q})$	total actual traffic volume on link l
$e_l > 0$ $(e_l \in \mathbb{Z})$	number of parallel links (both physical and virtual) at the place of link l
$E_n = \sum_{l \in S_n} e_l$	number of (physical and virtual) outgoing links at node n
D	number of demands
$O_d \in F$	originating node of demand d $(1 \leq d \leq D)$
$D_d \in F$	destination node of demand d
$G_d \in \mathbb{Q}^+$	traffic volume for demand d
$R \geq 0$ $(R \in \mathbb{Z})$	maximal number of virtual links per node
$\beta \in \mathbb{Q}^+$	maximal link utilization

Table 5.1: VRA-PGO notations

We are looking for *link weights* w_l and *parallel link number* e_l (including the physical and virtual links) for each link l, which minimizes the *maximal link utilization* $\beta = \max_{l \in F} h_l/c_l$, such that $E_n \leq |S_n| + R$ $(\forall n \in V)$.

In the previous two chapters it was more convenient to use the Bounded Total Resources model (see Sec. 2.2) requiring $E \leq Q$ and $E_i \leq Q$. Nevertheless, for Chapter 3 with a simple $R = Q - k$ substitution the Bounded Virtual Resources model ($E - k \leq R$) could have been used, too. Likewise, most of my findings in Chapter 4 are about the Unlimited Resources model, but the rest are also trivial to transform to the Bounded Virtual Resources scenario. With regard to the numerical evaluation (Chapter 6), however, using the Bounded Virtual Resources model is more practical, as different nodes can have different number of outgoing physical links. For this reason throughout this chapter, unless stated otherwise, I limit the number of virtual resources ($E_n \leq |S_n| + R$).

The problem can now be formulated as follows.

Problem 7, VRA-PGO. *Given* (V, F), $\{c_l\}$, D, $\{O_d, D_d, G_d\}$, *and* R, *find* $\{w_l\}$ *and* $\{e_l\}$ *that minimizes* β, *such that* $E_n \leq |S_n| + R$ $(\forall n \in V)$.

5.2 Optimal Solution

VRA-PGO is computationally hard to solve, as I will show in Sec. 5.3. Yet, I first show a way to find the *optimal* solution in the form of the following ILP.

LP 5.1 VRA Peer-Global ILP

indices:	d	$= 1 \dots D$, demands
	l	$= 1 \dots L$, links
	n	$= 1 \dots N$, nodes
constants:	c_l	> 0, link capacity
	r	a small constant
	r_2	$r_2 < r$, a smaller constant
	r_3	$r_3 < r$, another smaller constant
	M	a large constant
	G_d	the traffic volume for demand d
	O_d	originating node of demand d
	D_d	destination node of demand d
	δ_{dn}	traffic source indicator: $\delta_{dn} = \begin{cases} 1 & \text{if } n = O_d \\ 0 & \text{otherwise} \end{cases}$
	S_n	set of (physical) links originating at node n
	$\lvert S_n \rvert$	number of (physical) links originating at node n
	T_n	set of (physical) links arriving at node n
	R	≥ 0, max. number of usable *virtual* links per node
variables:	u_{dn}	real: node potential for demand d, node n
	w_l	$\geq 0, \leq 1$, real: link weight *minus* r (5.1)
	σ_{dl}	binary: $\sigma_{dl} = 1$ iff l is on a shortest path between O_d and D_d
	β	real, max. error
	e_l	≥ 1, integer: no. of parallel links
	θ_{dl}	≥ 0, real: traffic ratio of demand d that uses link l (5.2)

y_{dnk} $\forall d, n : \{|S_n| > 0, n \neq D_d\}$, $k = 0 \ldots |S_n| + R$, binary:

$y_{dnk} = 1$ iff $\sum_{l \in S_n} \sigma_{dl} e_l = k$

z_{lm} $m = 1 \ldots R + 1$, binary: $z_{lm} = 1$ iff $e_l = m$

objective: minimize $\beta + r_3 \sum_d \sum_l \theta_{dl} + r_3 \sum_l e_l$ (5.3)

constraints:

$$u_{dO_d} = 0 \quad \forall d \tag{5.4}$$

$$w_l + r - u_{dj} + u_{di} \geq (1 - \sigma_{dl}) r_2 \quad \forall l = (i, j), \forall d \tag{5.5}$$

$$w_l + r - u_{dj} + u_{di} \leq (1 - \sigma_{dl}) M \quad \forall l = (i, j), \forall d \tag{5.6}$$

$$\sum_{l \in S_n} \theta_{dl} - \sum_{l \in T_n} \theta_{dl} = \delta_{dn} \quad \forall d, n : n \neq D_d \tag{5.7}$$

$$\sum_{d=1}^{D} \theta_{dl} G_d \leq \beta c_l \quad \forall l \tag{5.8}$$

$$\left(\sum_{x \in T_n} \theta_{dx} + \delta_{dn} \right) m \leq \theta_{dl} k + M(3 - y_{dnk} - z_{lm} - \sigma_{dl}) \tag{5.9}$$

$$\left(\sum_{x \in T_n} \theta_{dx} + \delta_{dn} \right) m \geq \theta_{dl} k - M(3 - y_{dnk} - z_{lm} - \sigma_{dl}) \tag{5.10}$$

$$\sum_{m=1}^{R+1} z_{lm} = 1 \quad \forall l \tag{5.11}$$

$$\sum_{m=1}^{R+1} z_{lm} m = e_l \quad \forall l \tag{5.12}$$

$$\sum_{k=0}^{|S_n|+R} y_{dnk} = 1 \quad \forall d, n : |S_n| > 0,\ n \neq D_d \tag{5.13}$$

$$\sum_{k=0}^{|S_n|+R} y_{dnk} k \leq \sum_{l:\gamma_{dl}=1} e_l + M \left(\sum_{l:\gamma_{dl}=1} (1 - \sigma_{dl}) + \sum_{l:\gamma_{dl}=0} \sigma_{dl} \right) \tag{5.14}$$

$$\sum_{k=0}^{|S_n|+R} y_{dnk} k \geq \sum_{l:\gamma_{dl}=1} e_l - M \left(\sum_{l:\gamma_{dl}=1} (1 - \sigma_{dl}) + \sum_{l:\gamma_{dl}=0} \sigma_{dl} \right) \tag{5.15}$$

$$\theta_{dl} \leq \sigma_{dl} \quad \forall d, l \tag{5.16}$$

$$\theta_{dl} \geq \sigma_{dl} r \quad \forall d, l \tag{5.17}$$

$$\sum_{l \in S_n} e_l \leq |S_n| + R \quad \forall n : |S_n| > 0 \tag{5.18}$$

Notes:
(5.2): the formal definition is: $\theta_{dl} = h_{dl}/G_d$, where h_{dl} is the actual traffic volume for demand d on link l. Certainly, $0 \leq \theta_{dl} \leq 1$, but $\theta_{dl} \leq 1$ follows from the constraints of the ILP.
(5.9)–(5.10): $\forall d, l$, $k = 1 \ldots |S_n| + R$, $m = 1 \ldots R + 1$, $m \leq k$, $n \neq D_d$, where n is the source of l.
(5.14)–(5.15): $\forall d, n : |S_n| > 0$, $n \neq D_d$ and for all combinations of $\gamma_{dl} \in \{0, 1\}$, $l \in S_n$. In other words, these constraints are repeated $2^{|S_n|}$ times for each d, n (where $|S_n| > 0$, $n \neq D_d$), and in each one a different element of $\{0, 1\}^{|S_n|}$ is assigned to $\{\gamma_{dl} : l \in S_n\}$.

Let us see how this ILP works. The first three constraints come from the dual formulation of the multi-commodity flow problem. The node potentials of the demand origins are zero (5.4). If a link is on a shortest path of a demand ($\sigma_{dl} = 1$), then the difference of potentials of its adjacent nodes equals the link weight ($w_l + r$). On the other hand, if l is not part of a shortest path of d ($\sigma_{dl} = 0$), then the weight is larger than the difference (5.5), (5.6). $w_l \leq 1$ (5.1) will guarantee that the weights and therefore the node potentials remain finite. The rest of the constraints assure that for each demand there are a set of links connecting the source and destination for which $\sigma_{dl} = 1$, which will cause some of the node potentials to be nonzero, as expected. This first part provides the required link weights as outputs of the ILP.

The rest of the constraints ((5.7) and below) originate from the primal formulation of a multi-commodity flow problem, augmented by the VRA flow split behavior, i.e., splitting proportionally to the number of parallel links. This second part provides the number of parallel links (e_l) as output. The connection between the two parts is realized exclusively by variables σ_{dl}.

Equation (5.7) is the usual Kirchhoff junction rule. Eq. (5.8) is the link capacity constraint, where β is to be minimized in the objective. Constraints (5.9)–(5.15) together represent the ECMP equal-split rule applied to the virtual link scenario, which could be summarized as:

$$\theta_{dl} = \left(\sum_{x \in T_n} \theta_{dx} + \delta_{dn} \right) \frac{e_l}{\sum_{x \in S_n} e_x \sigma_{dx}} \ , \tag{5.19}$$

where n is the source of l, $\forall d, l : \sigma_{dl} = 1$. Unfortunately (5.19) is not linear, so I have

introduced a set of auxiliary variables, and multiplied and modified the constraints to make it fit the ILP.

Constraints (5.9) and (5.10) basically assert

$$\theta_{dl}k = \left(\sum_{x\in T_n} \theta_{dx} + \delta_{dn}\right) m \ , \tag{5.20}$$

where $m = e_l$ and $k = \sum_{x\in S_n} e_x\sigma_{dx}$. For a brief explanation first observe that eq. (5.9) and (5.10) cover a multitude of inequalities for different values of indexes d, l, k, m, and n as described in the second note under the ILP. Most of these, however, are not "live" constraints as $M(3-y_{dnk}-z_{lm}-\sigma_{dl}) \geq M$ make them automatically true. These are "real" constraints only when $M(3-y_{dnk}-z_{lm}-\sigma_{dl}) = 0$, i.e., $y_{dnk} = z_{lm} = \sigma_{dl} = 1$. For these cases $m = e_l$ and $k = \sum_{x\in S_n} e_x\sigma_{dx}$ are enforced, as described below.

$m = e_l$ is achieved by the simple set of constraints (5.11) and (5.12). Note that the upper bound of the sums is $R+1$, since using R virtual links the maximum of e_l is $R+1$.

Constraints (5.13)–(5.15) are to ensure $k = \sum_{x\in S_n} e_x\sigma_{dx}$. Observe that these constraints allow $k = 0\ldots|S_n| + R$. The upper limit comes from the definition of R, while $k = 0$ is allowed, since if no traffic of demand d goes through node n, then $\sum_{x\in S_n} e_x\sigma_{dx} = 0$. Equation (5.14) is repeated for all the possible $2^{|S_n|}$ combinations of γ_{dl}, but only one of them is a hard constraint (i.e., the right-hand side inside the parenthesis is zero): when $\gamma_{dl} = \sigma_{dl}\ \ \forall l$. The same applies to (5.15), providing together the required $k = \sum_{x\in S_n} e_x\sigma_{dx}$ for (5.20). Note that in these equations the conditions $|S_n| > 0$ and $n \neq D_d$ are theoretically not necessary, they are only included to reduce the number of variables and the related constraints.

Equations (5.16) and (5.17) ensure that there is flow from a demand on a link if and only if the corresponding σ_{dl} equals one. The last constraint, (5.18), limits the number of virtual links per node to R.

The objective function (5.3) minimizes the MLU (β). It also keeps the $\sum\theta_{dl}$ low to avoid loops and minimizes $\sum e_l$ to prevent installing unnecessary virtual links.

Finally, a few words about constants M, r, r_2, and r_3. Theoretically their value can be arbitrary as long as M is "large", r is "small", and r_2 and r_3 are "even smaller". In practice, however, these values should be set fairly accurately. For example, M should be large enough to make each equation, where it is not multiplied by zero, an

ineffective constraint. Theoretically, if we can find such an M, a larger one is always acceptable as well. Yet in practice having numerical values spanning too many orders of magnitudes is not favored by the ILP solvers, and so they may come up with erroneous results. Consequently, M should be kept relatively small, but large enough to fulfill its original purpose. Similar considerations apply for the small constants r, r_2, and r_3. In the simulation, after some theoretical calculations and practical experimenting, I found the following values appropriate: $M = 100$, $r = 10^{-2}$, $r_2 = r_3 = 10^{-4}$.

5.3 Computational Complexity

In the previous subsection I have given a slow, but optimal solution to the VRA-PGO problem. The algorithms given for Peer-Local Optimization can be considered as quicker, but suboptimal heuristics for the same problem. Now I show that finding a fast and even near-optimal solution is impossible, as the problem is computationally hard by its nature.

This subsection consists of two parts: the first one lists the NP-completeness theorems along with their proofs, while the second one carries the inapproximability results. I will use the notations introduced in Sec. 5.1.

5.3.1 NP-Completeness of VRA-PGO

First I slightly reformulate the Virtual Resource Allocation–Peer-Global Optimization problem to be a decision problem. Furthermore, in this first formulation I take the link weights as input parameters.

Problem 8, Virtual Resource Allocation–Peer-Global Optimization with Given Weights (VRA-PGO-GW).

INSTANCE. *A directed graph* (V, F) *representing a* network *with* capacities $c_l \in \mathbb{Q}^+$ *and* link weights $w_l \in \mathbb{Q}^+$ *for each link* $l \in F$. *A* set of demands $\{O_d \in F,\ D_d \in F,\ G_d \in \mathbb{Q}^+\}_{d=1}^{D}$. *The* maximal number of virtual links *that can be applied at a node:* $R \in \mathbb{Z}$ $(R \geq 0)$. *The* maximal link utilization*:* $\beta \in \mathbb{Q}^+$.

QUESTION. *Is there a VRA assigning* $e_l > 0$ $(e_l \in \mathbb{Z})$ *number of links to each physical link* $l \in F$, *such that* $E_n \leq |S_n| + R$ $(\forall n \in V)$ *and* $\max_l h_l/c_l \leq \beta$*?*

Algorithm 5.1 Calculating h_l (sketch)

Input: (V, F), $\{e_l\}$, $\{w_l\}$
Output: $\{h_l\}$

```
H_dn ← 0 ∀d, n {traffic volume of demand d entering node n}
h_dl ← 0 ∀d, l {traffic volume of demand d on link l}
for d ← 1, D do
    Run Dijkstra's alg.: x_nd ← shortest distance between n and D_d (∀n)
    Determine {σ_dl}_∀l from {x_nd} and {w_l} in O(|F|)
    Let (V', F') ⊆ (V, F) for which l ∈ F' iff σ_dl = 1. (V', F') is a DAG.
    Topologically sort n ∈ V' in O(|V'| + |F'|)
    for all n ∈ V' in the topological order do
        if n = O_d then
            H_dn ← G_d
        else
            H_dn ← Σ_{l∈T_n} h_dl {h_dl ≠ 0 due to the topological ordering}
        end if
        for all l ∈ S_n do
            h_dl ← H_dn e_l / Σ_{x∈S_n} e_x σ_dx
        end for
    end for
end for
for l ← 1, L do
    h_l ← Σ_{d=1}^{D} h_dl
end for
```

In the Question above $|S_n|$ can be calculated from (V, F); E_n and h_l can be calculated from (V, F), $\{e_l\}_{l \in F}$, $\{w_l\}_{l \in F}$ and from the set of demands. The only non-trivial point is the calculation of h_l, but it also can be done in polynomial time: see Algorithm 5.1.

This definition can be changed in several ways to obtain different versions of the problem, including the following:

- VRA-PGO: In this case setting the link weights, and this way defining the routing of the demands, is part of the problem, too. This variant is similar to VRA-PGO-GW, only the link weights are moved from the Instance to the Question. This problem has been examined in the previous parts of the dissertation.
- VRA-PGO-GW-SD (Single Demand): In this alternative we have only one origin–destination–traffic volume triplet ($D = 1$).
- VRA-PGO-GW-Q: The definition of VRA-PGO-GW utilizes the Bounded

Virtual Resources constraint ($E_n - |S_n| \leq R$). In this version the Bounded Total Resources ($E_n \leq Q$) limit is used instead.

- VRA-PGO-GW-ABS (Absolute Error): Here instead of the relative error (utilization) β, we have an absolute error, δ, requiring $\max_l(h_l - c_l) \leq \delta$.

By combining the definitions given above, several other equally valid variants of the VRA-PGO problem could be created. Fortunately, the following proofs about computational complexity can be generalized relatively easily to many of these new cases.

I start the list of the results with an important finding:

Theorem 22. *VRA-PGO-GW is NP-complete.*

Proof. This proof is inspired by an NP-completeness proof presented in [9].

First I show that VRA-PGO-GW is in NP, i.e., for a set $\{e_l\}$ it can be checked in polynomial time whether the conditions hold. The first set of conditions ($E_n \leq |S_n| + R$, $\forall n \in V$) is trivial to check in polynomial time. For the second condition ($\max_l h_l/c_l \leq \beta$), h_ls have to be calculated. For this, for each demand the shortest path routes have to be calculated first, which can be done in polynomial time. Then the nodes on the shortest paths (which now together define a DAG) have to be topologically sorted, for which the complexity is $\mathrm{NC}^2 \subseteq \mathrm{P}$. Then computing h_l and ultimately $\max_l h_l/c_l$ can also be done in polynomial time. (See Alg. 5.1 for details.)

Now I prove that VRA-PGO-GW is NP-hard. I will reduce 3SAT to VRA-PGO-GW. 3SAT is a famous NP-complete problem, and is formulated as follows.

Problem 9, Satisfiability of Boolean Expressions in 3CNF (3SAT).

Instance. *A Boolean expression F in conjunctive normal form with no more than three variables per clause (3CNF). F contains n variables $x_1, x_2, \ldots, x_n$, and consists of m clauses $C_1, C_2, \ldots, C_m$, such that each clause is a disjunction of exactly three literals.*[1]

Question. *Is F satisfiable?*

A simple example 3SAT problem is:

$$F = (x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee x_4) \tag{5.21}$$

[1]Some sources use "at most three literals" instead of "exactly three literals". These definitions are practically equivalent.

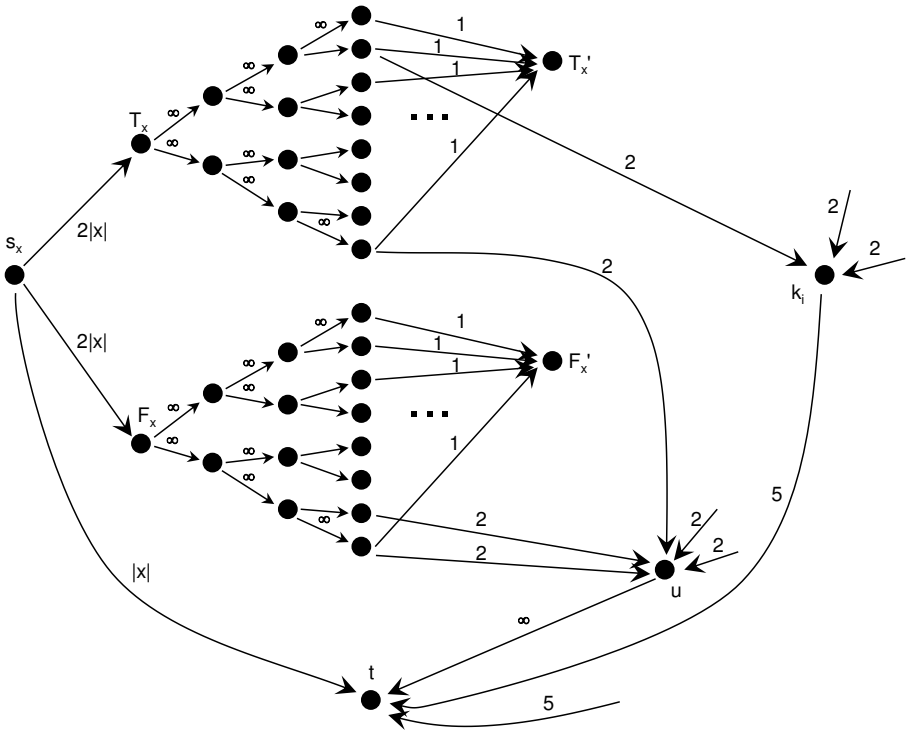


Figure 5.2: Network for the 3SAT reduction

This instance contains two clauses and it is satisfiable with for example $x_1 = x_2 = x_3 = x_4 = \text{TRUE}$.

For any 3SAT instance F, I create a corresponding instance of VRA-PGO-GW. Figure 5.2 sketches the *network*: node k_i corresponds to clause C_i. For each variable x, a set of nodes are defined: s_x, T_x, T'_x and a balanced binary tree between them, and likewise F_x, F'_x and a balanced binary tree between them. There are two global nodes, t and u. The number of the leaves of both trees directly downstream T_x and F_x is $|x|$ for each tree, which denotes the least power of 2 bounding both the number of negative and the number of positive occurrences of x in F ($|x| \geq 1$). For a positive occurrence of x in a clause C_i, there is an arc from a leaf under F_x to node k_i. For a negative occurrence of x in a clause C_i, there is an arc form a leaf under T_x to k_i (just like in the figure). Each leaf can only be used for at most one occurrence of x in the clauses. Each leaf that is not connected to a node representing a clause is connected to the global node u.

The *link capacities* are shown in the figure. The *link weights* are 1 for each link, except for links $s_x t$, for which $w_{s_x t} = \max(\log_2 |x|, 1) + 3$. The *demands* are as follows:

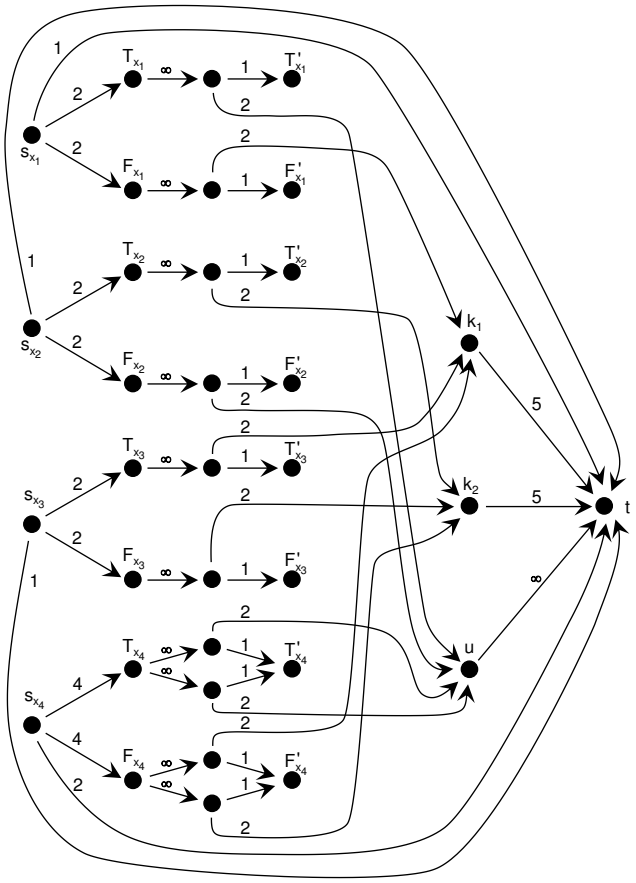


Figure 5.3: Example for $F = (x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee x_4)$

for each variable x: $s_x \to t : 4|x|$, $T_x \to T'_x : |x|$, $F_x \to F'_x : |x|$. Furthermore, $R = 1$, $\beta = 1$. This reduction is clearly polynomial.

As an example I show the VRA-PGO-GW instance for the 3SAT expression presented in (5.21). The graph with link capacities is plot in Fig. 5.3. The link weights are 1 for all the links, except for the four $s_{x_i}t$, for which the weights are 4. The demands are: $s_{x_i} \to t : 4$, $T_{x_i} \to T'_{x_i} : 1$, $F_{x_i} \to F'_{x_i} : 1$ for $i = 1, 2, 3$ and $s_{x_4} \to t : 8$, $T_{x_4} \to T'_{x_4} : 2$, $F_{x_4} \to F'_{x_4} : 2$. $R = 1$, $\beta = 1$.

I now prove that if the F 3SAT instance is satisfiable, then there is a suitable virtual link allocation for the VRA-PGO-GW problem. Let us consider a set of logical constants that satisfy F and use them for x, y, z, etc. Let $e_l = 1$ for all the links, except for one set of links: if x is true, then $e_{s_x T_x} = 2$ and $e_{s_x F_x} = 1$, otherwise, if x if false, then $e_{s_x T_x} = 1$ and $e_{s_x F_x} = 2$. Easily, $E_n \leq |S_n| + R$ holds for all the nodes.

For almost all the links $h_l/c_l \leq 1$ is trivially true, it is nontrivial only for the k_it type links, where the capacity is 5.

I now show that even for these links $h_l/c_l \leq 1$ holds. Each of the three incoming links of k_i has a demand originated at an s_x-like node, where x corresponds to a variable. If the literal in C_i corresponding to x is not satisfied (i.e., x is in positive form and x is false, or x is in negative form and x is true), then 2 units of traffic arrive to node k_i. If, however, the literal is satisfied, then 1 unit of traffic arrives due to the traffic split at node s_x. We know that F is satisfied, i.e., at most two of the literals in C_i are unsatisfied, meaning that at most 5 units of traffic can arrive to node k_i.

Now follows the opposite direction: if there is a suitable virtual link allocation for VRA-PGO-GW, then the 3SAT instance F is satisfiable. First consider the binary tree under T_x. As it has $|x|$ leaves, each connected to T'_x with a link with capacity of 1, and as there is a demand $T_x \to T'_x : |x|$, and as $R = 1$, it follows that $e_l = 1$ for all the links between T_x and T'_x for any suitable virtual link allocation for VRA-PGO-GW. By symmetry, this statement also holds for the links between F_x and F'_x. It is also easy to see that $e_{s_xt} = 1$ and either ($e_{s_xT_x} = 1$ and $e_{s_xF_x} = 2$) or ($e_{s_xT_x} = 2$ and $e_{s_xF_x} = 1$): if it were not so (i.e., $e_{s_xT_x} = e_{s_xF_x} = 1$ and either $e_{s_xt} = 1$ or $e_{s_xt} = 2$), then the capacity limit would be violated on s_xt. There are no other nodes in the network where traffic split occurs.

Observe the following: if the literal in C_i containing x is unsatisfied then 2 units of traffic arrive to k_i from s_x, otherwise 1 unit, as we have equal split throughout both balanced binary trees. Now, let each variable x be true, if $e_{s_xT_x} = 2$, and let it be false if $e_{s_xF_x} = 2$. This variable substitution will satisfy F. The reason is simple: for each clause C_i in F, there is a node k_i, and as $h_{k_it}/c_{k_it} = h_{k_it}/5 \leq 1$, there is at most 5 units of traffic arriving to k_i. This means that for each clause there is at most two unsatisfied literals in F. □

This proof can be easily modified to prove the NP-completeness of different versions of the problem.

Theorem 23. *VRA-PGO-GW-Q is NP-complete.*

Proof. Same as the previous proof, except for using $Q = 4$ instead of $R = 1$. Although in the binary trees it is possible to have $e_l = 2$ simultaneously for a pair of links

originated at a common node, as this still results in equal traffic split, this causes no problem. □

This also proves the following:

Theorem 24. *VRA-PGO-GW-Q is NP-complete if $Q = 4$ is a given constant.*

The strength of this theorem is that there is a constant Q, for which the problem is NP-complete, i.e., in this case Q is not an input parameter. To show an analogy, deciding if there is a k-vertex clique in a graph is NP-complete if k is an input parameter, but for any given k the problem can be solved in polynomial time.

Theorem 25. *VRA-PGO-GW-ABS is NP-complete.*

Proof. Same as the proof of Thm. 22, but by using $\delta = 0$ instead of $\beta = 1$. □

I have developed another proof of Theorem 22, along with some related theorems and proofs. However, to maintain the easy readability of this chapter those have been placed to Appendix C.3. Yet, those proofs are significant not only because they tackle the same problem family differently, but also because they target different variants of VRA-PGO. For example, Theorem 39 claims the NP-completeness of VRA-PGO itself.

The proof of the next theorem will be needed for the inapproximability results.

Theorem 26. *VRA-PGO-GW-SD is NP-complete.*

Proof. This proof is based on the proof of Theorem 22. This time, however, let $|x| = 2$ if there is no more than one negative and no more than one positive occurrence of x in F. VRA-PGO-GW-SD is in NP, for the same reasons as VRA-PGO-GW is in NP.

The *network* is modified, and is shown in Fig. 5.4. The first difference between Figs. 5.2 and 5.4 is the binary tree rooted at node s. It has q leaves, where

$$q = \min_{k \in \mathbb{Z}} 2^k : q > 12 \sum_{i=1}^{n} |x_i| \ .$$

Most of these leaves are connected to the nodes s_x, T_x, F_x, s_y, T_y, F_y, ... (each triplet representing a variable in the related 3SAT instance), as shown in the figure. The number of leaves connected to a single node is shown underlined. The rest of the

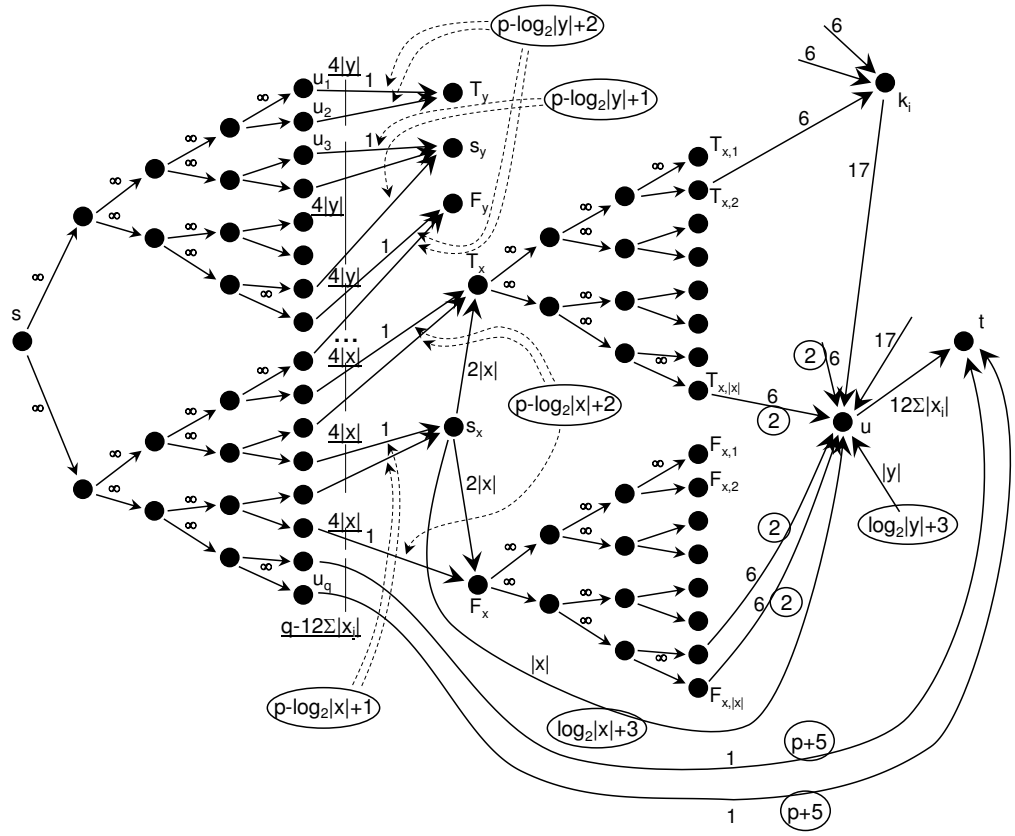


Figure 5.4: Network for VRA-PGO-GW-SD

leaves, not connected to any of the s_x, T_x, F_x type nodes, are connected to t. Their number is $q - 12\sum|x_i|$, which, by the definition of q, is at most $q/2$.

The leaves of the trees rooted at T_x and F_x are named $T_{x,1}$, $T_{x,2}$, ..., $T_{x,|x|}$ and $F_{x,1}$, $F_{x,2}$, ..., $F_{x,|x|}$, respectively. They are connected to some k_i or to u, as described previously.

The link *capacities* are shown next to each link. Let

$$p = \max_{i=1...n} \log_2 |x_i| \ .$$

All the *link weights* are 1 by default, except for the links where it is shown by the numbers in an ellipse. Let the single *demand* be $s \to t : q$. $R = 1$. $\beta = 1$. This reduction is polynomial.

First I prove the 3SAT $\Rightarrow$ VRA-PGO-GW-SD direction. Let x, y, z, etc. be the logical constants that satisfy F. Let $e_l = 1$ for all the links, except the following. If

x is true, then let $e_{s_xT_x} = 2$ and $e_{s_xF_x} = 1$; if x is false, let $e_{s_xF_x} = 2$ and $e_{s_xT_x} = 1$. These kind of link allocations will be called *canonical allocations* below.

First note that all possible directed paths between s and t are also a minimum total weight (i.e., shortest) path: the total length of these shortest paths are $\log_2 q + p + 5$.

Easily, $E_n \leq |S_n| + R$ holds for all the nodes. For almost all the links it is easy to see that $h_l/c_l \leq 1$ is true. For the k_iu type links (with capacity of 17 units), however, some explanation is needed. The reason why these links are not overutilized is the same as described at the proof of Theorem 22: 6 units of traffic arrives at k_i from the variables corresponding to a not satisfied literal and 5 units from the variables representing a satisfied literal. As at least one literal is satisfied in each clause, links k_iu are not overutilized.

Link ut is not overfilled either: the total traffic arriving to T_x, F_x, T_y, F_y, ... is $12\sum|x_i|$, and this is the traffic that will eventually traverse on ut, resulting in $h_{ut}/c_{ut} = 1$.

Now I prove the VRA-PGO-GW-SD $\Rightarrow$ 3SAT direction: if there is a suitable virtual link allocation for VRA-PGO-GW-SD, then the corresponding 3SAT instance is satisfiable.

I will first prove that any suitable virtual link allocation must be canonical.

First consider the tree rooted at s. It has q leaves and each one is part of a minimum total weight path from s to t. As q amount of traffic arrives at s, and as each one of the q leaves has a single outgoing link with one unit capacity and as $\beta = 1$, there must be one unit of traffic on each link leaving the leaves. This means that $e_l = 1$ for all the links within the tree under s.

Each link that is a single outgoing link at any node can have $e_l = 1$ or $e_l = 2$, but as in practice these result in identical behavior I will assume $e_l = 1$ for these links. This is the case for the links originated at the leaves of the tree under s, as well.

$4|x|$ amount of traffic arrives at s_x, and again, all three of its outgoing links are on shortest paths towards t (with total weight from s_x to t of $\log_2|x| + 4$). Easily, $e_{s_xu} = 1$ and either ($e_{s_xT_x} = 1$ and $e_{s_xF_x} = 2$) or ($e_{s_xT_x} = 2$ and $e_{s_xF_x} = 1$), otherwise the capacity limit would be violated on the link s_xu.

Let us now focus on the binary tree under T_x. Note that all the nodes and all the links within the tree are still on a shortest path. Depending on $e_{s_xT_x}$ and $e_{s_xF_x}$ either $5|x|$ or $6|x|$ amount of traffic arrives at T_x. I prove that in both cases $e_l = 1$ for all

the links in the binary tree. First note that if $e_l = 1$ for all the links and $5|x|$ arrives at T_x, then 5 units of traffic will reach each leaf, 10 units each node above the leaves, 20 each node above them, etc. Likewise, if $6|x|$ arrives at T_x, then 6 will arrive at each leaf, 12 at the next level, etc. Next, suppose the opposite of the statement to be proven: for some links within the tree $e_l = 2$. This means that there is at least one leaf, say $T_{x,i}$, where the incoming traffic is at least $10 \cdot 2/3 = 20/3$, if $5|x|$ arrives at T_x. (If there is only one link with $e_l = 2$ in the tree, and that is right above a leaf, then $10 \cdot 2/3 = 20/3$ units of traffic will arrive to exactly one leaf. If this link is higher up in the tree then there will be more than one leaves with 20/3 units of traffic. If there are several links with $e_l = 2$ then the arrival traffic of some leaves could be even higher.) Similarly, if $6|x|$ arrives at T_x then there is at least one leaf, say $T_{x,i}$, where the incoming traffic is at least $12 \cdot 2/3 = 8$. As both of these values (20/3 and 8) are greater than 6, our assumption leads to a contradiction violating $h_l/c_l \leq 1$ on the link leaving $T_{x,i}$, which proves that for all the links in the binary tree $e_l = 1$. The previous reasoning also yields that $h_{T_{x,i}u} = 6$ or $h_{T_{x,i}k_j} = 6$ (whichever exists) if $e_{s_x T_x} = 2$. Likewise, if $e_{s_x T_x} = 1$, then $h_{T_{x,i}u} = 5$ or $h_{T_{x,i}k_j} = 5$.

Certainly, the same proof applies to the links under F_x. As there are no more nodes in this network where traffic split may occur, we can safely suppose $e_l = 1$ for the rest of the links. At this point I have shown that $\beta = 1$ can happen only if the link allocation is canonical.

As links $k_i u$ are not overutilized by assumption, the related variable substitution (i.e., x is true if and only if $e_{s_x T_x} = 2$) will satisfy the 3SAT expression F. □

5.3.2 Inapproximability of VRA-PGO

To examine the approximability of a problem the first step is to formulate it as an NP optimization (NPO) problem [60]. Again, several problem definition versions could have been listed here, but for simplicity I only list those two, which are crucial for the main inapproximability results:

Problem 10, Minimal Error Virtual Resource Allocation–Peer-Global Optimization with Given Weights (MIN-VRA-PGO-GW, shortly MVPG).

INSTANCE. *A directed graph* (V, F) *representing a* network *with* capacities $c_l \in \mathbb{Q}^+$ *and* link weights $w_l \in \mathbb{Q}^+$ *for each link* $l \in F$. *A* set of demands $\{O_d \in F, D_d \in F,$

$G_d \in \mathbb{Q}^+\}_{d=1}^D$. *The* maximal number of virtual links *that can be applied at a node:* $R \in \mathbb{Z}^+$.

SOLUTION. *A virtual resource allocation assigning* $e_l > 0$ *(*$e_l \in \mathbb{Z}$*) number of links to each physical link* $l \in F$*, such that* $E_n \leq |S_n| + R$ *(*$\forall n \in V$*).*

MEASURE. *The* maximal link utilization $\beta = \max_l h_l/c_l$.

GOAL. *Minimize the measure.*

Note that $|S_n|$ can be calculated from (V, F); E_n can be calculated from (V, F) and from $\{e_l\}_{l\in F}$; h_l can be calculated from (V, F), $\{e_l\}_{l\in F}$, $\{w_l\}_{l\in F}$, and from the set of demands, as explained at Alg. 5.1.

Next I show that MVPG is indeed an NPO:

1. The set of the instances of MVPG is recognizable in polynomial time. This means that if Σ is the input alphabet and $\mathcal{I} \subseteq \Sigma^*$ is the set of input instances then $x \in \mathcal{I}$ for an $x \in \Sigma^*$ can be verified within polynomial time of $|x|$. For MVPG it is clearly the case.
2. The size of the solution is indeed a polynomial function of the size of the instance.
3. Deciding if a solution candidate is a solution or not can be done in polynomial time, as computing E_n is fast.
4. The measure can be calculated in polynomial time of the size of the solution. This statement is not trivial, but its proof is essentially the same as the proof of VRA-PGO-GW is in NP, presented in the proof of Theorem 22.

The following version of the previous problem will also be important for the approximability results.

Problem 11, Minimal Error VRA-PGO with Given Weights for a Single Demand (MIN-VRA-PGO-GW-SD, shortly MVPGS).

This is essentially the same as MVPG, but has only exactly one demand.

MVPGS is an NPO problem as well, because the reasons listed for MVPG are all valid for this problem version, too.

The proof of Theorem 22 can be extended to show that generally the optimal solution cannot even be approximated efficiently:

Theorem 27. *No polynomial time algorithm exists that approximates the optimum of MVPG better than a factor of* $6/5$ *(unless* $P = NP$*).*

Proof. This proof uses ideas from a similar reasoning presented in [9]. Also, it heavily relies on the proof of Theorem 22, using the same VRA-PGO-GW instance bound to a 3SAT problem (see Fig. 5.2). Finding an optimal resource allocation, which yields $\beta = 1$, is proven there to be NP-hard. I will now show that for the same instance any virtual resource allocation that results in $\beta > 1$ also results in $\beta \geq 6/5$.

In fact I prove an equivalent statement: for the given VRA-PGO-GW instance if a virtual resource allocation results in $\beta < 6/5$ then it also results in $\beta = 1$. Because of $R = 1$, for each link l, either $e_l = 1$ or $e_l = 2$. Consider first the links within the binary trees. If e_l were 2 for any of them, then there would be an ingress link of T'_x or F'_x, for which $h_l/c_l \geq 4/3$, which is against our assumptions.

Next, consider the outgoing arcs of s_x. If for all the three arcs $e_l = 1$, then $h_{s_x t}/c_{s_x t} = 4/3$, again a contradiction. $e_{s_x t} = 2$ would result in $h_{s_x t}/c_{s_x t} = 2$, which is not possible either. Thus either $e_{s_x T_x} = 2$ or $e_{s_x F_x} = 2$. As no more splitting occurs, we can suppose $e_l = 1$ for the rest of the links.

Now it is easy to see that for almost for all the links $h_l/c_l \leq 1$: the only critical links are the $k_i t$ type ones. Based on the previous observations, for a node k_i each incoming link carries either 1 or 2 units of traffic. Consequently, if $h_{k_i t}/c_{k_i t} < 6/5$ then $h_{k_i t}/c_{k_i t} \leq 1$. □

A corollary of this theorem is that *MVPG is not part of the PTAS (Polynomial Time Approximation Scheme) class*, as it is not possible to efficiently approximate the optimal solution within *every* constant ratio. I show now the same for the single demand version of the problem, which will also be used later, in the proof of a stronger statement:

Theorem 28. *No polynomial time algorithm exists that approximates the optimum of MVPGS better than a factor of* $18/17$ *(unless* $P = NP$*).*

Proof. This proof relies on the proofs of Theorems 26 and 27. I will use the same VRA-PGO-GW-SD instance as the proof of Theorem 26 (see Fig. 5.4). As finding the optimal resource allocation for this instance is already proven to be NP-hard, I will show, just as at the proof of Theorem 27, that if for the given VRA-PGO-GW-SD instance a virtual resource allocation results in $\beta < 18/17$ then it also results in $\beta = 1$.

I first show that a resource allocation with $\beta < 18/17$ must be canonical (for the definition see the proof of Theorem 26). Because of $R = 1$, for each link l, either $e_l = 1$ or $e_l = 2$. Just as before, we may safely suppose $e_l = 1$ for the links that are single outgoing links from a node.

Consider first the links within the binary tree under s. If e_l were 2 for any of them, then there would be an egress link of at least one of the leaf nodes, for which $h_l/c_l \geq 4/3$, which is against our assumptions.

Next, consider the outgoing arcs of s_x. If for all the three arcs e_l were 1, then $h_{s_xu}/c_{s_xu} = 4/3$, again a contradiction. $e_{s_xu} = 2$ would result in $h_{s_xu}/c_{s_xu} = 2$, which is not possible either. Thus either $e_{s_xT_x} = 2$ or $e_{s_xF_x} = 2$.

Let us now focus on the binary trees under T_x. At the proof of Theorem 26 I have shown that if for some links within the tree $e_l = 2$, then there is at least one leaf, say $T_{x,i}$, where the minimum amount of traffic on its outgoing link is at least 20/3. As for this link $h_l/c_l \geq (20/3)/6 = 20/18 > 18/17$, this would violate our $\beta < 18/17$ assumption. Certainly, the same proof applies to the binary tree under F_x.

As no more splitting occurs, we can suppose $e_l = 1$ for the rest of the links. Thus I have shown that a resource allocation with $\beta < 18/17$ is indeed canonical.

Now it is easy to see that if $\beta < 18/17$ then for almost all the links $h_l/c_l \leq 1$: the only critical links are the k_iu type ones. Based on the previous observations, for a node k_i each incoming link carries either 5 or 6 units of traffic. Consequently, either $h_{k_iu} \leq 17$, meaning $h_{k_iu}/c_{k_iu} \leq 1$, or $h_{k_iu} = 18$, yielding $h_{k_iu}/c_{k_iu} = 18/17$, which is against our assumptions.

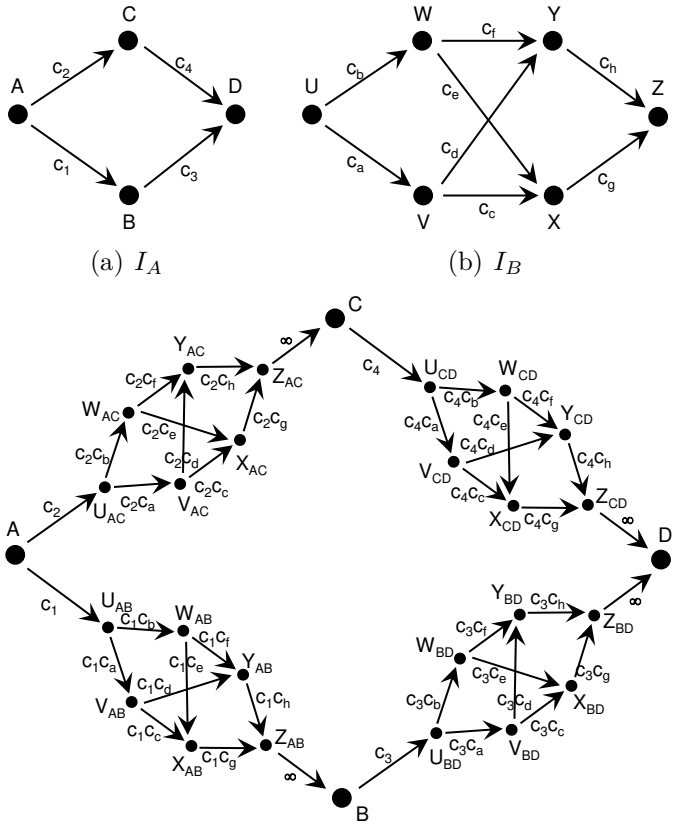
This means that if for this VRA-PGO-GW-SD instance a virtual resource allocation results in $\beta < 18/17$ then it also results in $\beta = 1$. □

From the previous theorem it follows that *MVPGS is not part of the PTAS class*, either. The next statement is the strongest in this section.

Theorem 29. *No polynomial time algorithm exists that could approximate the optimal solution of MVPGS to* any *given constant ratio (unless P = NP).*

In other words this means that *MVPGS is not part of the APX class.*

The proof of Theorem 29 applies the *inapproximability gap amplification technique*, which has recently been introduced in [43] to prove a similar inapproximability for the OSPF ECMP link weight configuration problem.



(a) I_A (b) I_B

(c) $I_A \otimes I_B$

Figure 5.5: MVPGS compounding

Just like at the inapproximability proof in [43], I first introduce the $\otimes$ (compound) operator for MVPGS instances. From two instances I_A and I_B a new instance $I = I_A \otimes I_B$ can be crated by compounding if both of the following conditions hold:

1. the traffic volume of the demand to be transmitted in I_B is 1,
2. the allowed maximum number of virtual links (R) is identical in I_A and I_B.

An example of compounding is shown in Fig. 5.5. The capacities are shown next to the links and each link weight is one unit. In I_A the demand is $A \to D : 1$, in I_B it is $U \to Z : 1$. $R = 1$ in all these instances.

The formal definition of $I_A \otimes I_B$ is the following (see Fig. 5.6). Take the I_A network and replace each link in it with the following subnetwork. Let the original link in I_A be ab with capacity c and weight w, and the demand in I_B be $s \to t$. Let the total minimum weight of $s \to t$ (i.e., the length of the shortest path between s and t) be

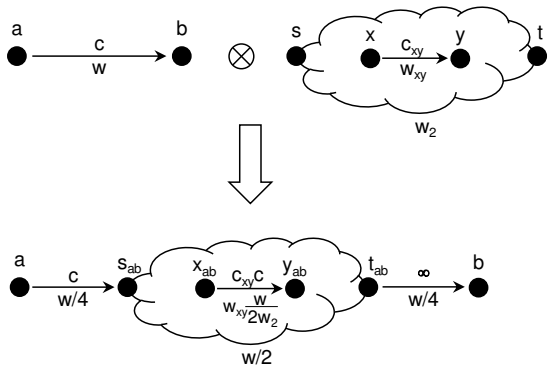


Figure 5.6: $\otimes$ definition

w_2. In $I_A \otimes I_B$ create a link as_{ab} with capacity c and weight $w/4$. Also create a link $t_{ab}b$ with infinite capacity and weight $w/4$. Between s_{ab} and t_{ab} insert the network of I_B: s replaced by s_{ab}, t becoming t_{ab}, x turning to x_{ab}, etc. For each link xy in I_B with capacity c_{xy} and weight w_{xy} create a link $x_{ab}y_{ab}$ of capacity $c_{xy}c$ and weight $w_{xy}w/(2w_2)$. This way the total minimum weight of the ab subnetwork in $I_A \otimes I_B$ will remain w, and also the shortest paths between s_{ab} and t_{ab} will stay as it were between s and t. Let the demand of the new $I_A \otimes I_B$ instance be equal to the demand of I_A, and R in the new instance be as it was in I_A and I_B.

For an MVPGS instance I let $OPT(I)$ denote the measure for the optimal solution, i.e., the minimal β. Furthermore, let us use the notation

$$
\begin{aligned}
I_0 &= \otimes^0 I = I \\
I_1 &= \otimes^1 I = I \otimes I \\
I_2 &= \otimes^2 I = I \otimes (\otimes^1 I) = I \otimes (I \otimes I) \\
&\vdots \\
I_k &= \otimes^k I = I \otimes (\otimes^{k-1} I)
\end{aligned}
$$

Note that in general $(I \otimes I) \otimes I \neq I \otimes (I \otimes I)$.

Now the following lemma can be presented and proven:

Lemma 30. *Let I be an instance of MVPGS with $OPT(I) \geq 1$. Then $OPT(\otimes^k I) = (OPT(I))^{k+1}$ for any $k \in \mathbb{Z}$, $k \geq 0$.*

Proof. The proof is by induction. For $k = 0$ we have $OPT(\otimes^0 I) = OPT(I)^1$, which is clearly true. Now suppose the lemma is true for k, i.e. $OPT(I_k) = OPT(I)^{k+1}$, and I prove it for $k + 1$.

First I prove that $OPT(I_{k+1}) \leq OPT(I)^{k+2}$, by giving a link allocation setting in I_{k+1} with MLU $OPT(I)^{k+2}$. Let e_l^0 denote the number of parallel links at link l in I_0, which results in the optimal allocation. Furthermore, let e_l^k be the number of parallel links at link l for the optimal allocation in I_k (for which $OPT(I_k) = OPT(I)^{k+1}$). Likewise, let e_l^{k+1} denote the number of links at l in I_{k+1}. This new link allocation is:

$$e^{k+1}_{as_{ab}} = e^0_{ab}, \quad e^{k+1}_{t_{ab}b} = 1, \quad e^{k+1}_{x_{ab}y_{ab}} = e^k_{xy} \ .$$

Let $\beta_l^0 = h_l/c_l$ be the utilization of link l in I_0 with the optimal link setting. Similarly, let β_l^k be the link overload in I_k using optimal allocation. The link utilization in I_{k+1} is:

$$\beta^{k+1}_{as_{ab}} = \beta^0_{ab}, \quad \beta^{k+1}_{t_{ab}b} = 0, \quad \beta^{k+1}_{x_{ab}y_{ab}} = \beta^k_{xy}\beta^0_{ab} \ .$$

We used a link allocation for which $\max_l \beta_l^0 = OPT(I)$, which is supposed to be at least one, and we assumed that $\max_l \beta_l^k = OPT(I)^{k+1}$. These yield $\max_l \beta_l^{k+1}$ will not take place at an as_{ab} or $t_{ab}b$ type link. Instead,

$$\max_l \beta_l^{k+1} = \max_{xy,ab} \beta^{k+1}_{x_{ab}y_{ab}} = \max_{xy,ab} \beta^k_{xy}\beta^0_{ab} = OPT(I)^{k+1} \cdot OPT(I) = OPT(I)^{k+2} \ .$$

Next I prove that $OPT(I_{k+1}) \geq OPT(I)^{k+2}$. The proof is by contradiction. Suppose the opposite, i.e. for a suitable link allocation in I_{k+1}: $OPT(I_{k+1}) < OPT(I)^{k+2}$. Thus, using the previous notations, we suppose that for this link setting for each link l: $\beta_l^{k+1} < OPT(I)^{k+2}$.

Let us focus on a subnetwork in I_{k+1}, which corresponds to a link ab in I_0. Let $\beta^{k+1}_{as_{ab}} = \delta_{ab}$. There are two possibilities:

In the first case for all links ab in I_0: $\delta_{ab} < OPT(I)$ in I_{k+1}. This would mean, however, that using $e^0_{ab} = e^{k+1}_{as_{ab}}$ in I_0 would result in $\beta^0 < OPT(I)$, which contradicts to the definition of $OPT(I)$.

In the second case there is at least one link ab in I_0 such that $\delta_{ab} \geq OPT(I)$ in I_{k+1}. Consider now the corresponding $s_{ab} \to t_{ab}$ subnetwork in I_{k+1} with unaltered link allocation. Suppose it had one unit of incoming traffic and let us denote the

utilization for link $x_{ab}y_{ab}$ in this case with $\gamma_{x_{ab}y_{ab}}$. We know that $\beta^{k+1}_{x_{ab}y_{ab}} = \delta_{ab}\gamma_{x_{ab}y_{ab}}$ and we supposed for all the links that $\beta^{k+1}_{l} < OPT(I)^{k+2}$. This means for all xy that

$$OPT(I)^{k+2} > \beta^{k+1}_{x_{ab}y_{ab}} = \delta_{ab}\gamma_{x_{ab}y_{ab}} \geq OPT(I)\gamma_{x_{ab}y_{ab}}$$

that is

$$\gamma_{x_{ab}y_{ab}} < OPT(I)^{k+1} \ ,$$

which means that the I_k instance could be solved with MLU less than $OPT(I)^{k+1}$, which is again a contradiction. □

Now we are ready to prove that MVPGS is not part of the APX class:

Proof of Theorem 29. According to the theorem, for *any* constant factor $\alpha > 1$ there is no polynomial time algorithm that can find a solution to each MVPGS instance I with MLU less than $\alpha \cdot OPT(I)$. To show this, for each α I will create one MVPGS instance and show that it is not possible to quickly approximate the optimum within a factor of α for that instance.

First take the instance described at proof of Theorem 26, with the network plot in Fig. 5.4. Divide each link capacity by q and let the demand be $s \to t : 1$. The other parts of the instance (e.g. the link weights) are unaltered. I will call this instance I_0.

From the proof of Theorem 28 it follows that either $OPT(I_0) = 1$ or $OPT(I_0) \geq 18/17$ (depending on the solubility of the 3SAT problem behind it), and deciding between these two possibilities is NP-hard.

Let k be the smallest positive integer such that $(18/17)^k \geq \alpha$. Now let us create MVPGS instance $I_{k-1} = \otimes^{k-1} I_0$. First note that the size of I_{k-1} can be upper bounded by a polynomial function of the size of I_0; consequently it can also be upper bounded by a polynomial function of the size of the 3SAT problem behind it. Next, according to Lemma 30 $OPT(I_{k-1}) = OPT(I_0)^k$. This means that either $OPT(I_{k-1}) = 1$ or $OPT(I_{k-1}) \geq (18/17)^k \geq \alpha$ and it is NP-hard to decide which case holds. The latter is true as if we could decide in polynomial time between these options then by this we could also solve I_0 quickly.

This means that if $OPT(I_{k-1}) = 1$ then it cannot be approximated in polynomial time better than a factor of α. □

Chapter 6

Numerical Evaluation

The complexity-related results of the previous chapter state the hardness of Peer-Global Optimization in general. They, however, do not necessarily mean that in practical networks no effective solution can exist. To see how the different algorithms perform in realistic environments I have implemented a simulation framework, which is based on my descriptive use case, OSPF ECMP Traffic Engineering.

The simulator takes a capacitated network and a set of demands as inputs and solves the Virtual Resource Allocation problem using several different algorithms. I have implemented the framework and the optimization algorithms in C++ using the powerful LEMON Graph Library [58]. I have solved the embedded linear programs using the IBM ILOG CPLEX Optimizer [61].

Note that unlike my analytical results, this numerical evaluation could hardly be conclusive. Yet, my results show the potential of the proposed techniques on a set of typical inputs. Furthermore, the presented simulation framework provides a quick way to test the performance of different algorithms on any given network and demand set.

6.1 Examined Algorithms

I have included the following seven optimization approaches in my simulator:

1. Overlay Optimization, as described in Chapter 3.
2. Overlay Optimization with Path Exclusion, see Sec. 2.3.2 and Chapter 3.
3. Peer-Local Optimization using ILP, described in Section 4.4.1 (Alg. 4.4).

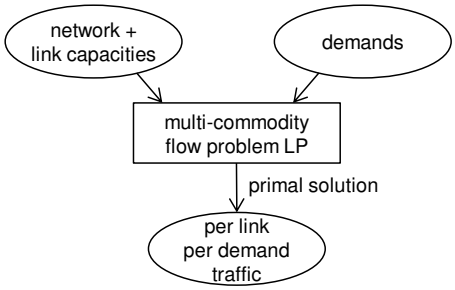


Figure 6.1: Global Optimization

4. Heuristic Peer-Local Optimization, as shown in Section 4.4.3 (Alg. 4.5).
5. Peer-Global Optimization using the ILP presented in Sec. 5.2.
6. Global Optimization, described in Sec. 6.1.1 below.
7. OSPF Weight Optimization, as introduced in Sec. 2.3.1, and detailed in Sec 6.1.2 below.

6.1.1 Global Optimization

Taking the capacitated network and the demands and solving the related multi-commodity flow LP results in the optimal per link per demand traffic (see Fig. 6.1). This serves as the first step of the Overlay Optimization and Peer-Local Optimization mechanisms, as described earlier. If, by using an adequately sophisticated TE mechanism, the demands could be routed perfectly according to the solution of this LP then the theoretical minimal MLU could be reached.

Accordingly, I have included a simple algorithm in my simulation platform that treats the outputs of this multi-commodity LP as actual traffic values. These results will then serve as reference values, since no algorithm (neither Peer-, nor Overlay-based) can perform better than this one. Furthermore, I will actually divide the MLU's of the different algorithms by this optimal MLU to have a normalized value, which is independent of the actual link bandwidths and traffic volumes.

The result of this optimization will be denoted in the charts as *Global Optimum*. Certainly, when displaying MLU values, Global Optimum will be constant 1.0 due to the normalization.

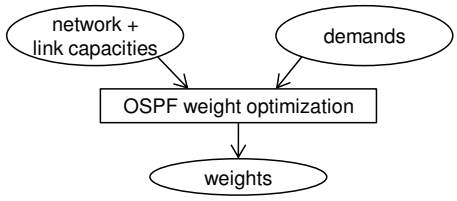


Figure 6.2: OSPF Weight Optimization

6.1.2 OSPF Weight Optimization

The OSPF Weight Optimization problem is simple to define: set the link weights so that running OSPF with ECMP on top of this network will generate the best result, which is the minimal MLU in our case (see Fig. 6.2). This is actually OSPF-TE. Note that in this case we are not applying virtual resources at all.

The OSPF Weight Optimization problem is proven to be NP-hard [9]. In the same paper a link weight local search heuristic is proposed, which have been implemented in an open source toolbox, called TOTEM (TOolbox for Traffic Engineering Methods [62, 63]). TOTEM itself is a Java-based graphical, modular toolkit, and the algorithm described in [9] has been implemented in a C language module called IGPWO (Interior Gateway Protocol Weight Optimizer [64]). Nevertheless, in the rest of this dissertation I will simply refer to this algorithm as the "TOTEM" method.

I have taken the source code of the IGPWO module out of the TOTEM (ver. 3.2.1) framework and (after fixing some bugs) included in my simulations to serve as a best-practice solution of the OSPF Weight Optimization problem. To do so, the error function of TOTEM has been modified. The original implementation contained a convex, piecewise linear cost function of the link load, which were summed over all the links, whereas I simply used the maximal link utilization as the error to be minimized. During the simulations I have used the following settings: iteration number: 50 000, max link weight: 5, random initial weights, minimum sampling rate: 0.001, maximum sampling rate: 0.04, initial sampling rate: 0.02.

6.1.3 Implementation Aspects

Previously, in Chapters 3 and 4 on Overlay and Peer-Local Optimization the Bounded Total Resources model (see Sec. 2.2) has been used (requiring $E \leq Q$ and $E_i \leq Q$) for easier presentation. Yet, as described in Sec. 5.1, these considerations can easily

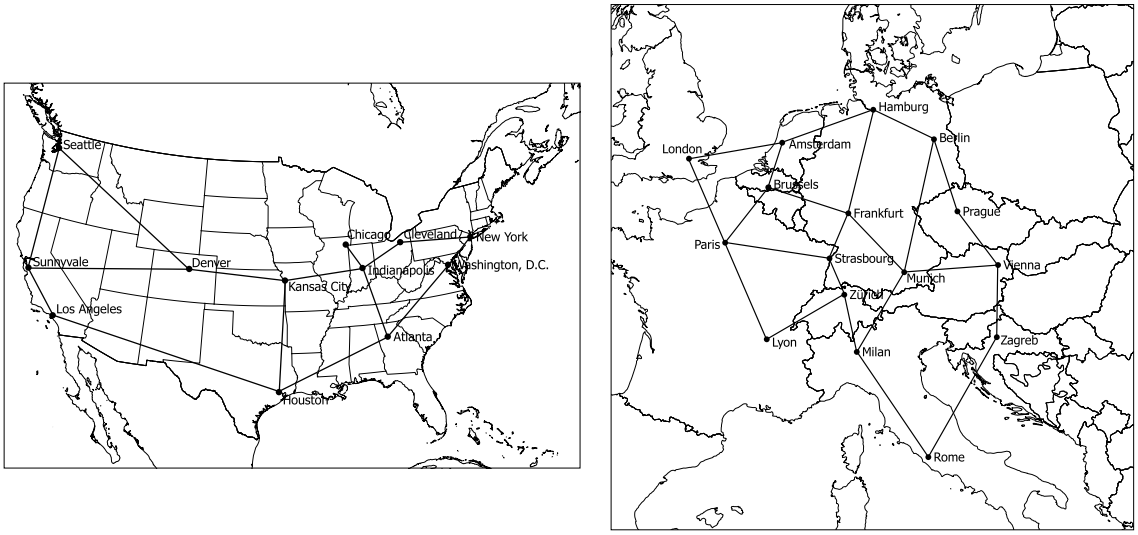
be transformed to the Bounded Virtual Resources scenario. This latter model is also used at the Peer-Global Optimization discussions in Chapter 5. In the simulation the Bounded Virtual Resources constraint has been implemented as well. The actual requirement is $E - |S_n| \leq R$, where E is the total number of links/paths used at a node, $|S_n|$ is the number of physical outgoing links/paths of a node and R is the maximal number of virtual links/paths that can be installed per node. The reason for this choice is that it makes it easier to compare the algorithms running at different nodes with different number of outgoing physical links. Also, in this case for the Peer-Local and Peer-Global Optimization scenarios $R = 0$ reverts to the classical OSPF-TE optimization problem without virtual links, providing a meaningful comparison.

Practical problems arose when the path decomposition module in the Overlay Optimization algorithms (see Fig. 3.1(b)) returned a path with very little traffic on it. In this case the VRA-1N-1D algorithm tried not to overutilize this path, which certainly provided the local optimum for the VRA-1N-1D problem, but it was proven to be highly suboptimal regarding the global MLU. To overcome this issue, if a path was found with traffic less than 5% of the total demand then it was deleted and its traffic was distributed over the rest of the paths, resulting in considerably lower MLU.

Similarly, to avoid the same problem for the Peer-Local Optimization algorithms, the case when a link is on a shortest path of a demand ($\sigma_{ij} = 1$), but it has very little traffic on it ($g_{ij} < 10^{-7}$ or $\gamma_{ij} < 0.05$) is treated exceptionally. In this situation the traffic proportion of the demand on the given link (γ_{ij}) has been risen at the expense of the other outgoing links on the shortest paths.

6.2 Simulation Scenarios

I have used three realistic network topologies for the simulations. The first one is the well known North American Abilene network topology, shown in Fig. 6.3(a) (see also Table 6.1). The second examined network is shown in Fig. 6.3(b): it is a simplified Pan-European optical core network, which have been proposed in [65]. In both networks uniform link bandwidths of 100 units have been used. The third network (AS3967) was taken from the inferred ISP data maps of the Rocketfuel dataset [66]. Approximate POP-level maps has been obtained by collapsing the topologies so that nodes correspond to cities and leaf nodes have been eliminated. This network comes



(a) Abilene network (b) Pan-European network

Figure 6.3: Network topologies

Network	No. of nodes	No. of unidirectional links	Link capacities [units]
Abilene	12	30	100
Pan-European	16	46	100
AS3967	21	72	44–1000

Table 6.1: Network characteristics

with inferred link capacities between 44 and 1000 units, with an average of 345 units. The resulting graph for this network contained 21 nodes over three continents and 72 capacitated unidirectional edges.

I had 5 demands in each simulation session for the Abilene and the Pan-European networks, and 16 for the larger AS3967 topology. In each case the source and destination nodes were selected randomly. The traffic volumes have also been picked at random for each demand with uniform distribution on the $[5, 30]$ units interval. The maximal number of virtual links or paths (R) was varied between 0 and 8, inclusive.

Due to the complexity of the ILP applied in Peer-Global Optimization, it took up to several hours, or even days to run a single instance of simulation (consisting of 9 runs with $R = 0 \ldots 8$) on the smaller Abilene and Pan-European topologies with only 5 demands. Yet, these scenarios have been simulated 300 times (with all the seven algorithms) on a high-performance computer to decrease the variance of the results.

The AS3967 topology was too complex to run Peer-Global Optimization on it, but on the other hand it allowed me to run the rest of the algorithms 3000 times.

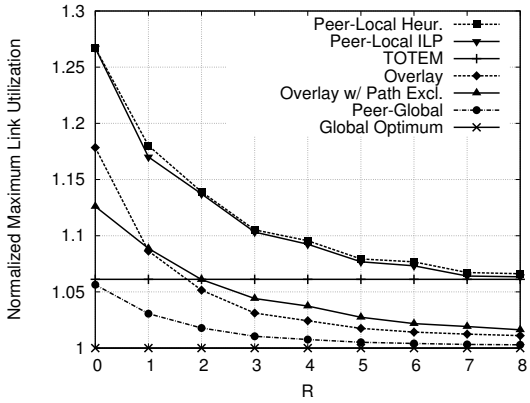
6.3 Simulation Results

The results are shown in Fig. 6.4. The first three charts show the MLU as a function of R for the examined scenarios.

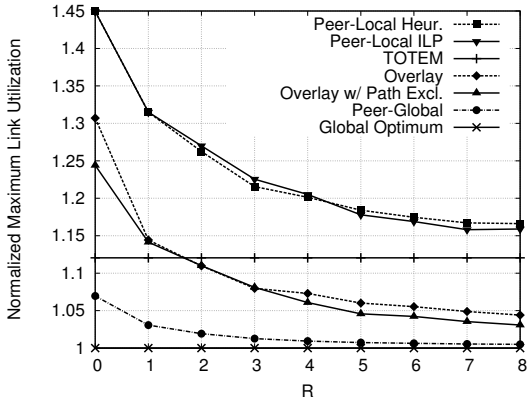
For the Abilene network TOTEM performed almost as good as Peer-Global Optimization for the no virtual link case, which is its theoretical lower bound. For $R = 0$ Peer-Local and Overlay Optimizations performed clearly worse than TOTEM, which is no surprise: running a VRA algorithm without virtual resources does not make much sense. However, allowing only two virtual links per node the performance of Overlay Optimization became as good as TOTEM's, and as R grows, Overlay Optimization clearly overperformed TOTEM, getting as close as a few percents to the Global Optimum. The Peer-Local ILP and Peer-Local Heuristic algorithms performed the worst, only reaching the MLU of TOTEM at $R = 8$. Note, however, that these are quick heuristics only for the VRA-PGO problem. The Peer-Global Optimization's MLU is well below TOTEM's even for $R = 1$ and it keeps decreasing as R increases, almost reaching the Global Optimum for only $R = 4$. This shows that the Peer-Global Optimization approach does have a high potential, but the currently applied heuristics are not taking full advantage of this, leaving space for future research for better ones.

For the Pan-European scenario the results are similar. Note that here TOTEM performed significantly worse comparing to its theoretical limit. This is not surprising though, as TOTEM itself is just a heuristic optimization algorithm for an NP-hard problem. Here Peer-Local Optimizations performed somewhat worse, yet Peer-Global Optimization shows, that the theoretical Global Optimum is very closely approachable using VRA.

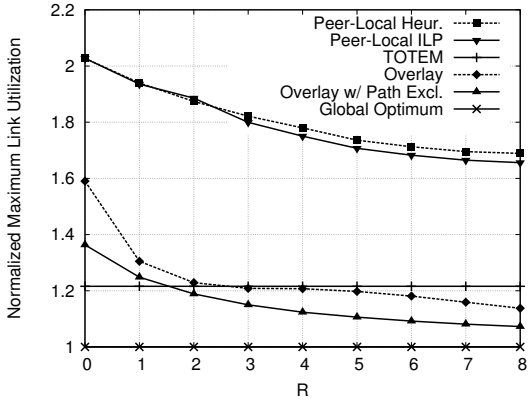
For the AS3967 case the Peer-Local approaches perform even weaker, but Overlay Optimization is still better than TOTEM at $R = 3$ already. Here the "Peer-Global Optimum" curve is missing, as the related ILP was practically unsolvable for this larger network.



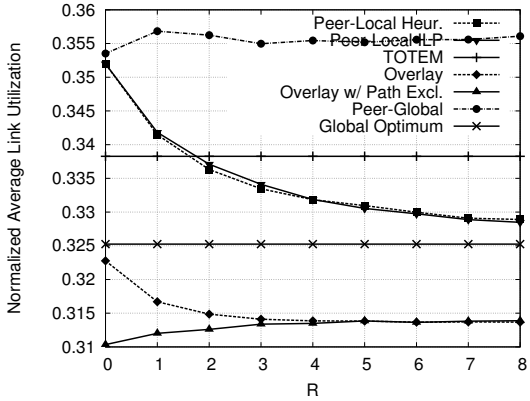
(a) Normalized MLU for the Abilene topology



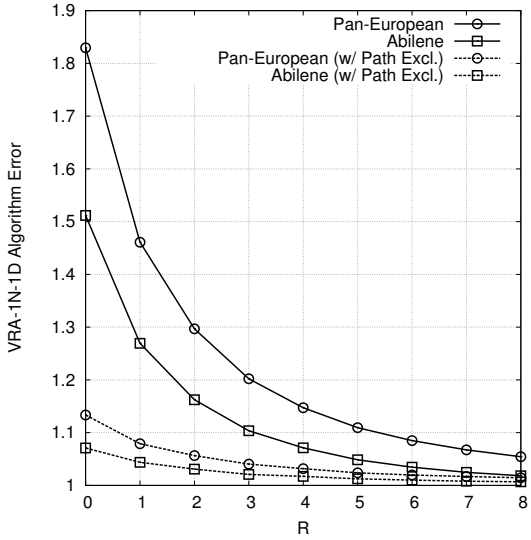
(b) Normalized MLU for the Pan-European topology



(c) Normalized MLU for the AS3967 topology



(d) Normalized Average LU for the Pan-European topology



(e) VRA-1N-1D Algorithm Error (U) for Overlay Optimization

Figure 6.4: Simulation results

Fig. 6.4(d) shows the average link utilizations for the Pan-European network, again normalized by the optimal Maximal Link Utilization, which is why all the results are well below one. As all the presented algorithms aim to minimize a different metric, the MLU, the average link utilization chart is considerably different than the MLU charts. Here, Peer-Global Optimization performs the worst and the best ones are the Overlay methods, but note that all the measured values are within 5% of the optimal MLU.

Regarding the Peer-Local algorithms it might be somewhat surprising that the ILP is not always better than the heuristic approach. The reason is simple: *locally* the ILP version is better, certainly, but a worse local solution can actually result in a better global MLU, as the graphs shows. Nonetheless, in all the cases their performance is very close to each other and mostly the ILP-based version performs better.

Similarly, sometimes Overlay Optimization with Path Exclusion seems to perform worse than plain Overlay Optimization. This is again due to the fact that the (local) optimization objective and the (global) measured metric is different. Note also that Overlay Optimization can theoretically overperform Peer-Global Optimization, although it did not happen in my practical evaluation.

Looking at the solution of the embedded VRA-1N-1D problem (Fig. 6.4(e)), Optimization with Path Exclusion always performs better, which is not surprising as it has a higher degree of freedom. This figure shows also clearly that the convergence to the optimum is quick as R increases. For $R = 6$ the solution is within 10% of the optimum, which indicates the strength of Alg. 3.3 and the whole VRA concept.

6.3.1 Resource Consumption

The simulations has been carried out on the High Performance Computing Cluster of the Budapest University of Technology and Economics (called “superman”). Each optimization session was run on a single core of an Intel Xeon Processor X5660 (but several sessions were run in parallel). The average running times and memory consumption of the different algorithms are summarized in Table 6.2.

The first row (“5 algorithms”) shows the total resource usage of the two Overlay Optimizations, the two Peer-Local Optimizations and Global Optimization. The second row represents the TOTEM optimization alone, while the third row reveals

Algorithms run	Average running time			Average memory usage		
	Abilene	Pan-Eu	AS3967	Abilene	Pan-Eu	AS3967
5 algorithms	0.35 s	0.74 s	1.82 s	7 MB	9 MB	10 MB
TOTEM	5.98 s	18.1 s	74.7 s	4 MB	4 MB	4 MB
7 algorithms	9 m 37 s	9 h 49 m	–	46 MB	1.5 GB	–

Table 6.2: Resource consumption of the different algorithms

the results for all the seven algorithms altogether (i.e., it includes TOTEM and Peer-Global Optimization as well). All the displayed results represent average values gained over 300 simulation runs.

The results show very short ($\approx$ second, sub-second) running times for Overlay, Peer and Global Optimizations. No further performance-profiling has been carried out here, but the results suggest that these algorithms are likely to be suitable when short response times are needed, like real-time TE optimization. For TOTEM, the calculations took several tens of seconds, even exceeding a minute, which is considerably higher than the previous ones, but still can be practical for non-realtime tasks. The memory usage was modest, only 4–10 MB in these cases. Note that the variance of the results discussed so far were very low.

When Peer-Global Optimization was included the average running times increased up to several hours. In this case the variance was much higher as well: the running times for a single session ranged from a couple of seconds to several days. The memory usage also varied from a few MB to almost 30 GB. This means that the proposed Peer-Global Optimization algorithm may not be a viable option in many of the practical cases.

Chapter 7

Conclusions

7.1 Summary

In this dissertation I have studied the possibility of enhancing load balancing schemes by unequal traffic splitting when the underlying technology only offers roughly uniform data distribution among the resources. For this, I have proposed the Virtual Resource Allocation (VRA) framework, which tricks a legacy load balancer into an almost arbitrary traffic split ratio. As an example of this flexibility, if used in an OSPF-TE environment, this simple proposal can significantly enhance the network performance without any hardware or software modification of today's routers. Instead, VRA can be applied right away only by changing a few administrative settings in the management plane.

I have examined the theoretical limits of the formalized problem, and, where it was theoretically possible, have given fast and optimal algorithms to determine where and how much virtual resources to provision. I have shown, however, that finding the optimal allocation for some important scenarios is computationally intractable. For these cases I have proposed quick heuristic algorithms along with a necessarily slow, but optimal one.

I have implemented a simulation evaluation framework for a possible VRA application: IP Traffic Engineering. My simulation results underpin that the VRA approach has a huge practical potential. In the examined networks, the VRA Peer-Global Optimization algorithm achieved much better network performance than the "traditional"

	Computational complexity	Practical running time	Network performance	Requirements beyond OSPF ECMP
VRA Overlay Optimization	pseudo-polynomial	very fast	good, with few virtual links	end-to-end tunnels for each shortest path; virtual paths
VRA Peer-Local Optimization	unknown in general, pseudo-poly. for the heuristic	very fast	slightly worse than the others	virtual links
VRA Peer-Global Optimization	NP-complete	very slow	very good	virtual links
OSPF Weight Optimization (TOTEM as heuristic)	undefined (iterative) for the heuristic, NP-complete in general	moderate	moderate	none
Global Optimization	polynomial	very fast	best (optimal)	explicit path setup with arbitrary split ratios

Table 7.1: High level summary of different techniques

(TOTEM) method. Moreover, according to the results, the theoretical best performance could be approximated up to 1–2% by allowing as few as 3–4 virtual links per node. Another VRA implementation, Overlay Optimization, is also proven to be a very effective tool. Although it requires a more sophisticated network infrastructure, the proposed algorithms performed outstandingly in the simulations. They ran very fast with minimal memory usage, and overperformed TOTEM by using only two or three virtual paths.

For quick comparison Table 7.1 contains a high-level summary of the presented techniques; the numerical details can be found in Chapter 6. A short note on the requirements for Global Optimization (last row, last column in the table): explicit path setup with arbitrary split ratios can be achieved in several ways, including the MPLS-TE implementation of most of today's routers.

Finally, for the list of my theses please refer to Appendix A, and for a comparative list of the problem definitions given in this work see Appendix B.

7.2 Possible Future Work

In this dissertation several findings have been presented concerning different variants of the Virtual Resource Allocation problem. These cover the theoretical basics of VRA, as well as algorithms that can be used in communication networks right away. Yet, certainly there are several possible ways to enhance or extend my findings, which are beyond the limits of a single dissertation. I list a collection of such open questions below.

Global Overlay Optimization. In Chapter 3 I have given a method for Overlay Optimization that first finds the optimal multipath routing of the demands and then allocates virtual paths, as shown in Fig. 3.1(b). Theoretically, better results could be achieved if the routing of the demands and the virtual resource allocation are optimized jointly. Certainly this approach is much harder. Paper [35] discusses a similar problem, but with a different optimization objective: network utility maximization instead of maximum link usage minimization.

Heuristic for VRA-PGO. I have shown that the VRA-PGO problem is NP-complete, and it cannot even be approximated efficiently. I have also provided a slow ILP that finds an optimal solution. Moreover, VRA Peer-Local Optimization itself can be considered as a quick heuristic to VRA-PGO. Despite the fact that in general no good approximation is possible, for realistic networks heuristics better than Peer-Local Optimization could exist. To find one is a task remaining for further research.

NP-completeness of VRA-1N-1D/mD. My complexity-related results on VRA-PGO practically mean that it is generally computationally hard to find optimal (and, actually, near-optimal) solutions to the Virtual Resource Allocation problem for several nodes and several demands simultaneously. I presume that for a single node (VRA-1N-mD) the problem is also NP-complete, and what is more, it is NP-complete for one node and one demand (VRA-1N-1D), too. Proving (or disproving) these is also left for future research.

Certainly, if the NP-completeness of VRA-1N-1D could be proven, it would imply the same property for VRA-1N-mD, being the former special case of the latter. Note,

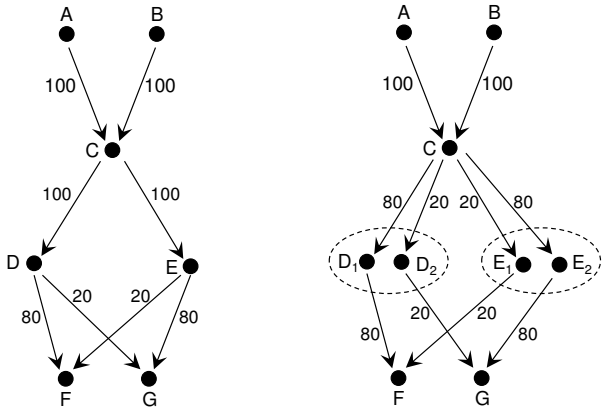
however, that although Algorithm 3.3 (presented in Sec. 3.3) to solve the VRA-1N-1D problem is not polynomial, it runs very fast for Qs in the range of thousands or even millions, which is well above the foreseeable useful domain. Likewise, the Algorithms presented in Sec. 4.4 perform well in practical situations.

NP-completeness of GSA-W. I have proven the NP-completeness of different versions of the VRA-PGO problem. A related open question, as detailed in Appendix C.3.2, is whether the "Good Simultaneous (Diophantine) Approximation in a Weaker Sense" problem is NP-complete. My conjecture is that it is indeed NP-complete, but it seems to be fairly hard to prove this and it is also out of the scope of this dissertation.

Topology virtualization. Several techniques have been proposed in the literature and used in practice for creating a virtual network over a physical one. There are a variety of reasons to do so, one of them is to have a simpler network than the original one, as suggested in [67] recently. My Overlay Optimization method, presented in Chapter 3, is also such a technique. Its main advantage is the decomposition of a large problem into independent, smaller subproblems (one for each demand) what results in a fast and very efficient operation. On the other hand, its path decomposition step may end up with too many end-to-end shortest paths, which can result in suboptimal splitting if the number of applicable virtual paths is not too high. Using end-to-end tunnels (Overlay Optimization) is one extreme, the other is not using overlaying at all. In between them there are several possibilities, including the ones proposed in [67]. Using shorter (non end-to-end) tunnels, may be a beneficial trade-off that results in good decomposability yet smaller number of shortest paths, therefore resulting in better network performance. This topic, however, is left for future work.

Another form of topology virtualization could be introducing virtual nodes alongside the existing ones. As an example, for the capacitated network shown in Fig. 7.1(a) the demands $A \to F : 100$, $B \to G : 100$ cannot be delivered without error using virtual links only. This is because matrix G of node C is

$$G = \begin{bmatrix} 80 & 20 \\ 20 & 80 \end{bmatrix} ,$$



(a) Original topology with link capacities

(b) Virtual topology with link capacities

Figure 7.1: Introducing virtual nodes. Demands: $A \to F : 100$, $B \to G : 100$

which is clearly inconsistent. If, however, we could virtually duplicate nodes D, E and their incoming links (see Fig. 7.1(b)), then this matrix would be

$$G = \begin{bmatrix} 80 & 0 & 20 & 0 \\ 0 & 20 & 0 & 80 \end{bmatrix} ,$$

which is now consistent. This means that with six virtual links[1] ($e = \begin{bmatrix} 4 & 1 & 1 & 4 \end{bmatrix}$) the demands can be routed without any error. As this simple example shows, achieving better resource utilization by node virtualization is another promising future research topic. As a theoretical problem, the best virtualization strategy should be found, and concurrently, practical means for node virtualization in real networks should be experimented with.

Deciding consistency in linear time. The complexity of Algorithm 4.1, which decides the consistency of a matrix G, is $O(d^2k)$. This is polynomial-time and is fast enough in all practical cases, but theoretically it could be interesting to examine whether there is an algorithm solving the same problem in $O(dk)$.

[1]Six virtual links compared to the topology shown in Fig. 7.1(b), which certainly means eight virtual links compared to the original topology depicted in Fig. 7.1(a).

Unique optimal solution for VRA-1N-mD-Unlimited. I suspect the uniqueness of the optimal solution for the VRA-1N-mD-Unlimited problem on some conditions.

Before continuing, let me include here a few simple definitions:

- A *submatrix* B of matrix A is a matrix that is obtained by deleting an arbitrary (non empty) set of rows and/or columns from matrix A.
- An *independent submatrix* B of matrix A is a submatrix, for which the following holds: if we highlight in A the rows and columns corresponding to B, then other than B itself, only 0 valued elements are highlighted.
- An *undividable matrix* is a matrix that contains no independent submatrices.

For example in

$$A = \begin{bmatrix} 1 & 2 & 0 & 1 & 0 & 1 \\ 0 & 0 & \mathbf{3} & 0 & \mathbf{3} & 0 \\ 0 & 0 & \mathbf{3} & 0 & \mathbf{3} & 0 \\ 1 & 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & \mathbf{3} & 0 & \mathbf{3} & 0 \\ 1 & 0 & 0 & 1 & 0 & 2 \end{bmatrix}$$

the boldfaced

$$B = \begin{bmatrix} 3 & 3 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \end{bmatrix}$$

submatrix is an independent submatrix. Here A is not undividable, B is undividable.

My first conjecture is that *there is a unique optimal* $\{f_j\}$ *solution of a VRA-1N-mD-Unlimited problem, if the corresponding matrix* G *is undividable.*

My second conjecture is that *if matrix* G *of a VRA-1N-mD-Unlimited problem contains independent submatrices, then the solution of the problem is unique to its independent, undividable submatrices.*

Let me explain this second conjecture. Let G_1, G_2, ..., G_n be the independent, undividable submatrices of G. (It should also be proven that such a division into submatrices is unique.) I suspect that the solution for each G_i is unique (first conjecture). Let me denote the sum of f_js belonging to the columns of G_i with f_{G_i}. I suspect that

there are infinite number of optimal solutions to the original problem, as any set of $\{f_{G_i}\}$s can be sufficient, as long as $f_{G_i} > 0$ $(\forall i)$ and $\sum_{i=1}^{n} f_{G_i} = 1$.

As an example, suppose that G consists of three independent, undividable submatrices, G_1, G_2 and G_3. Suppose further that f_1, f_2 and f_3 belongs to G_1, f_4 and f_5 to G_2, and f_6 and f_7 to G_3. Then, solving the 3 different subproblems, there will be unique solutions in forms of $\{f_j\}$ such that $f_1+f_2+f_3 = 1$, $f_4+f_5 = 1$ and $f_6+f_7 = 1$. Now, for the original problem af_1, af_2, af_3, bf_4, bf_5, cf_6, cf_7 $(a+b+c = 1, a, b, c > 0)$ is an optimal solution. (Using the previous notations, $F_{G_1} = a$, $F_{G_2} = b$, $F_{G_3} = c$.) Moreover, a, b, c can be arbitrary as long as $a+b+c = 1$ and $a, b, c > 0$, so there are infinite number of solutions to the original problem.

As these considerations are a bit out of the scope of my dissertation, and I have proven my theorems without them, I have left them for possible future work. Nevertheless, proving these conjectures, which are interesting on their own, too, would largely simplify the proofs presented in Appendices C.1 and C.2.

Proof-of-concept deployment. VRA is a general concept for augmenting legacy equal-split load balancing systems. Nevertheless, for a given application, like the presented OSPF-TE, a proof-of-concept deployment would be interesting. This would not require hardware or software modifications in the routers, just some control plane settings when installing the virtual links. The results are expected to be similar to those gained by simulations.

Bibliography

[1] D. Wischik, M. Handley, and M. B. Braun, "The resource pooling principle," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 5, pp. 47–52, 2008.

[2] "IEEE standard for local and metropolitan area networks–link aggregation," *IEEE Std 802.1AX-2008*, pp. 1–163, Nov 2008.

[3] A. Vakali and G. Pallis, "Content delivery networks: status and trends," *IEEE Internet Computing*, vol. 7, no. 6, pp. 68–74, Nov 2003.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[5] J. Zhou, M. Tewari, M. Zhu, A. Kabbani, L. Poutievski, A. Singh, and A. Vahdat, "WCMP: Weighted cost multipathing for improved fairness in data centers," in *Proceedings of the Ninth European Conference on Computer Systems*, ser. EuroSys, Apr. 2014, pp. 5:1–5:14.

[6] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.

[7] J. T. Moy, "OSPF Version 2," RFC 2328, Mar. 2013.

[8] International Organization for Standardization, "Intermediate System to Intermediate System intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (ISO 8473)," ISO/IEC 10589:2002, November 2002.

[9] B. Fortz and M. Thorup, "Increasing internet capacity using local search," *Computational Optimization and Applications*, vol. 29, pp. 13–48, 2004.

[10] N. Kang, M. Ghobadi, J. Reumann, A. Shraer, and J. Rexford, "Efficient traffic splitting on commodity switches," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '15, Dec. 2015, pp. 6:1–6:13.

[11] A. Maghbouleh, "Metric-based traffic engineering: Panacea or snake oil? A real-world study," in *27th North American Network Operators Group Meeting (NANOG27)*, Feb 2003.

[12] A. Sridharan, R. Guérin, and C. Diot, "Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks," in *IEEE INFOCOM 2003*, vol. 2, March 2003, pp. 1167–1177.

[13] D. Xu, M. Chiang, and J. Rexford, "Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering," *IEEE/ACM Trans. Netw.*, vol. 19, no. 6, pp. 1717–1730, 2011.

[14] S. Vissicchio, O. Tilmans, L. Vanbever, and J. Rexford, "Central Control Over Distributed Routing," in *ACM SIGCOMM*, August 2015, pp. 43–56.

[15] O. Tilmans, S. Vissicchio, L. Vanbever, and J. Rexford, "Fibbing in action: On-demand load-balancing for better video delivery," in *ACM SIGCOMM*, 2016, pp. 619–620.

[16] M. Chiesa, G. Rétvári, and M. Schapira, "Lying your way to better traffic engineering," in *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '16, 2016, pp. 391–398.

[17] 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*, ser. SIGCOMM '08, 2008, pp. 63–74.

[18] C. Hopps and D. Thaler, "Multipath issues in unicast and multicast next-hop selection," RFC 2991, Nov. 2000.

[19] S. Kandula, D. Katabi, S. Sinha, and A. Berger, "Dynamic load balancing without packet reordering," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 2, pp. 51–62, Mar. 2007.

[20] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'10, 2010.

[21] K. He, E. Rozner, K. Agarwal, W. Felter, J. Carter, and A. Akella, "Presto: Edge-based load balancing for fast datacenter networks," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM '15, 2015, pp. 465–478.

[22] J. Cao, R. Xia, P. Yang, C. Guo, G. Lu, L. Yuan, Y. Zheng, H. Wu, Y. Xiong, and D. Maltz, "Per-packet load-balanced, low-latency routing for Clos-based data center networks," in *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '13, 2013, pp. 49–60.

[23] A. Kabbani, B. Vamanan, J. Hasan, and F. Duchene, "Flowbender: Flow-level adaptive routing for improved latency and throughput in datacenter networks," in *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '14, 2014, pp. 149–160.

[24] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav, and G. Varghese, "CONGA: Distributed congestion-aware load balancing for datacenters," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, 2014, pp. 503–514.

[25] N. Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford, "HULA: Scalable load balancing using programmable data planes," in *Proceedings of the Symposium on SDN Research*, ser. SOSR '16, 2016, pp. 10:1–10:12.

[26] S. Ghorbani, Z. Yang, P. B. Godfrey, Y. Ganjali, and A. Firoozshahian, "DRILL: Micro load balancing for low-latency data center networks," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17, 2017, pp. 225–238.

[27] E. Vanini, R. Pan, M. Alizadeh, P. Taheri, and T. Edsall, "Let it flow: Resilient asymmetric load balancing with flowlet switching," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI '17)*, 2017, pp. 407–420.

[28] K. Németh and G. Rétvári, "Traffic splitting algorithms in multipath networks: Is the present practice good enough?" in *15th International Telecommunications Network Strategy and Planning Symposium (Networks)*, Oct 2012, pp. 1–6.

[29] I. Pepelnjak, "Improving ECMP Load Balancing with Flowlets," Jan. 2015, retrieved Nov. 2018. [Online]. Available: http://blog.ipspace.net/2015/01/improving-ecmp-load-balancing-with.html

[30] A. Ford, C. Raiciu, M. J. Handley, and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses," RFC 6824, Jan. 2013.

[31] C. H. Benet, A. J. Kassler, T. Benson, and G. Pongracz, "MP-HULA: Multipath Transport Aware Load Balancing Using Programmable Data Planes," in *Proceedings of the 2018 Morning Workshop on In-Network Computing*, ser. NetCompute '18, 2018, pp. 7–13.

[32] Z. Cao, Z. Wang, and E. Zegura, "Performance of hashing-based schemes for Internet load balancing," in *IEEE INFOCOM 2000*, vol. 1, March 2000, pp. 332–341.

[33] M. Wang, C. W. Tan, W. Xu, and A. Tang, "Cost of not splitting in routing: Characterization and estimation," *IEEE/ACM Transactions on Networking*, vol. 19, no. 6, pp. 1849–1859, Dec 2011.

[34] X. Liu, S. Mohanraj, M. Pióro, and D. Medhi, "Multipath routing from a traffic engineering perspective: How beneficial is it?" in *2014 IEEE 22nd International Conference on Network Protocols (ICNP)*, Oct 2014, pp. 143–154.

[35] Y. Bi, C. W. Tan, and A. Tang, "Network utility maximization with path cardinality constraints," in *IEEE INFOCOM 2016*, April 2016, pp. 1–9.

[36] Y. Bi and A. Tang, "Cost of not arbitrarily splitting in routing," in *2017 IEEE 25th International Conference on Network Protocols (ICNP)*, Oct 2017, pp. 1–10.

[37] K. Németh, A. Kőrösi, and G. Rétvári, "Optimal OSPF traffic engineering using legacy Equal Cost Multipath load balancing," in *2013 IFIP Networking Conference*, May 2013, pp. 1–9.

[38] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "Overview and principles of Internet traffic engineering," RFC 3272, May 2002.

[39] E. C. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," RFC 3031, Jan. 2001.

[40] D. Awduche, L. Berger, D.-H. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels," RFC 3209, Dec. 2001.

[41] Y. Wang and Z. Wang, "Explicit routing algorithms for internet traffic engineering," in *Computer Communications and Networks, 1999. Proceedings. Eight International Conference on*, 1999, pp. 582–588.

[42] Z. Wang, Y. Wang, and L. Zhang, "Internet traffic engineering without full-mesh overlaying," in *IEEE INFOCOM 2001*, vol. 1, April 2001, pp. 565–571.

[43] M. Chiesa, G. Kindler, and M. Schapira, "Traffic Engineering with Equal-Cost-MultiPath: An Algorithmic Perspective," in *IEEE INFOCOM 2014*, April 2014, pp. 1590–1598.

[44] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *IEEE INFOCOM 2000*, vol. 2, 2000, pp. 519–528 vol.2.

[45] M. Pióro, Á. Szentesi, J. Harmatos, A. Jüttner, P. Gajowniczek, and S. Kozdrowski, "On open shortest path first related network optimisation problems," *Performance Evaluation*, vol. 48, no. 1–4, pp. 201 – 223, 2002, Performance Modelling and Evaluation of ATM & IP Networks.

[46] Cisco Systems, Inc, "Cisco Nexus 7000 Series NX-OS Unicast Routing Command Reference," June 2016, Chapter: M Commands, Section: maximum-paths (EIGRP, IS-IS, RIP, OSPF, OSPFv3). Retrieved Nov. 2018. [Online]. Available: http://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus7000/sw/unicast/command/reference/n7k_unicast_cmds/l3_cmds_m.html

[47] Current Analysis, Inc., "Product Assessment: Ericsson - SmartEdge Series," July 2009, retrieved Nov. 2018. [Online]. Available: http://archive.ericsson.net/service/internet/picov/get?DocNo=13/28701-FGB101647&Lang=EN&HighestFree=Y

[48] Juniper Networks, Inc., "JunosE™Software for E Series™Broadband Services Routers, Command Reference A to M," July 2013, Release 14.3.x, Section: commands/maximum-paths. Retrieved Nov. 2018. [Online]. Available: https://www.juniper.net/techpubs/en_US/junose14.3/information-products/topic-collections/command-reference-a-m/sw-cmd-ref-a-m.pdf

[49] I. Pepelnjak, "Unequal cost load-sharing," Feb. 2007, retrieved Nov. 2018. [Online]. Available: http://blog.ipspace.net/2007/02/unequal-cost-load-sharing.html

[50] M. Pióro and D. Medhi, *Routing, Flow, and Capacity Design in Communication and Computer Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

[51] V. S. Mirrokni, M. Thottan, H. Uzunalioglu, and S. Paul, "A simple polynomial time framework for reduced-path decomposition in multipath routing," in *IEEE INFOCOM 2004*, vol. 1, March 2004, pp. 739–749.

[52] B. Vatinlen, F. Chauvet, P. Chrétienne, and P. Mahey, "Simple bounds and greedy algorithms for decomposing a flow into a minimal set of paths," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1390–1401, 2008.

[53] T. Hartman, A. Hassidim, H. Kaplan, D. Raz, and M. Segalov, "How to split a flow?" in *IEEE INFOCOM 2012*, March 2012, pp. 828–836.

[54] G. Rétvári, J. J. Bíró, and T. Cinkler, "On Shortest Path Representation," *IEEE/ACM Transactions on Networking*, vol. 15, no. 6, pp. 1293–1306, Dec. 2007.

[55] "Maple (mathematical software)," 2018, retrieved Nov. 2018. [Online]. Available: http://www.maplesoft.com/products/Maple/

[56] R. Joseph, *Galois Theory*, ser. Universitext. Springer New York, 1998, second edition.

[57] "GNU Linear Programming Kit," Jun. 2012, Retrieved Nov. 2018. [Online]. Available: https://www.gnu.org/software/glpk/

[58] "LEMON Graph Library – Library for Efficient Modeling and Optimization in Networks," July 2014, Version 1.3.1. Retrieved Nov. 2018. [Online]. Available: http://lemon.cs.elte.hu/

[59] K. Németh, A. Kőrösi, and G. Rétvári, "Enriching the poor man's traffic engineering: Virtual link provisioning for optimal OSPF TE," in *Telecommunications Network Strategy and Planning Symposium (Networks), 16th International*, Sept 2014, pp. 1–7.

[60] V. Kann, "On the Approximability of NP-complete Optimization Problems," Ph.D. dissertation, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden, May 1992.

[61] "IBM ILOG CPLEX Optimizer," Retrieved Nov. 2018. [Online]. Available: https://www.ibm.com/analytics/cplex-optimizer

[62] G. Leduc, H. Abrahamsson, S. Balon, S. Bessler, M. D'Arienzo, O. Delcourt, J. Domingo-Pascual, S. Cerav-Erbas, I. Gojmerac, X. Masip, A. Pescapè, B. Quoitin, S. P. Romano, E. Salvadori, F. Skivée, H. T. Tran, S. Uhlig, and H. Ümit, "An open source traffic engineering toolbox," *Comput. Commun.*, vol. 29, no. 5, pp. 593–610, Mar. 2006.

[63] "TOTEM Project: TOolbox for Traffic Engineering Methods," Latest version (3.2.1) released at Nov. 2008. Retrieved Nov. 2018. [Online]. Available: http://totem.run.montefiore.ulg.ac.be/

[64] H. Ümit, "Interior Gateway Protocol Weight Optimization Tool," Feb. 2007, retrieved Nov. 2018. [Online]. Available: http://www.poms.ucl.ac.be/totem/

[65] S. D. Maesschalck, D. Colle, I. Lievens, M. Pickavet, P. Demeester, C. Mauz, M. Jaeger, R. Inkret, B. Mikac, and J. Derkacz, "Pan-european optical transport networks: An availability-based comparison," *Photonic Network Communications*, vol. 5, no. 3, pp. 203–225, May 2003.

[66] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "Inferring link weights using end-to-end measurements," in *ACM Internet Measurement Workshop*, 2002, pp. 231–236.

[67] S. I. Nikolenko, K. Kogan, and A. Fernández Anta, "Network simplification preserving bandwidth and routing capabilities," in *IEEE INFOCOM 2017*, May 2017.

[68] M. Pióro, A. Szentesi, J. Harmatos, A. Jüttner, P. Gajowniczek, and S. Kozdrowski, "On open shortest path first related network optimisation problems," *Perform. Eval.*, vol. 48, no. 1-4, pp. 201–223, May 2002.

[69] J. C. Lagarias, "The computational complexity of simultaneous diophantine approximation problems," *SIAM J. Comput.*, vol. 14, no. 1, pp. 196–209, Feb. 1985.

Index

Page numbers in *italics* represent the definition or another important appearance of the indexed term.

3SAT, *61*, 61–68, 70, 75

Abel–Ruffini theorem, 119

Cisco, 12, 14
CONGA, 6
COYOTE, 6

data center, 1, 4, *6*
DRB, 6
DRILL, 7

ECMP, *2*, 6, 10, 12–14, 18, 20, 21, 31, 34, 57, 71, 76, 78, 86
EIGRP, 12
Ericsson, 12
Exact Cover by 3-Sets, *see* X3C

Fibbing, *5*, 6, 17, 21
Flare, 6
FlowBender, 6
flowlet, 6, 7, *7*

Galois theory, 43, 120
Global Optimization, 77, *77*, 81, 83, 86
Good Simultaneous (Diophantine) Approximation, *see* GSA
Good Simultaneous (Diophantine) Approximation in a Weaker Sense, *see* GSA-W
GSA, *123*
GSA-W, 88, *123*, 123–125

Hedera, 6
HULA, 6, 7

IS-IS, 2, 4, 12

Juniper, 12

LetFlow, 7

Maple, 43, 119, 120
MP-HULA, 7
MPLS-TE, *11*, 14, 18, 86
Multipath TCP, 7

Niagara, *5*, 12, 28
NP-completeness, 59–68, 86–88, 120–125

NP-hardness, 12, 19, 78, 81
NPO approximation, 12, 19, *68*, 87
  APX, 71, 75
  inapproximability gap amplification, 71
  PTAS, 70, 71

OSPF, 2, 4, 10, 12–15, 18, 20, 21, 30, 31, 71, 86
  -TE, *11*, 14, 17, 23, 28, 31, 76, 78, 79, 91
  weight optimization, 12, 14, 77, 78, 86

PEFT, 4
Presto, 6

resource bounds, *12*
  Bounded Total Resources, 13, *13*, 14, 21, 30, 34, 54, 61, 78
  Bounded Virtual Resources, 13, *13*, 54, 61, 79
  Unlimited Resources, *12*, 37, 54
RIP, 12
RSVP-TE, 11

Satisfiability of Boolean Expressions in 3CNF, *see* 3SAT
SDN, 1, 3–5, 12, 17, 21, 23

TOTEM, *78*, 81, 83, 84, 86
Traffic Engineering (TE), 1–4, 6, 10, *11*, 13, 17, 77, 78, 84, 85

VRA, 2, 3, *10*
  Overlay Optimization, 14, *18*, 76–78, 81, 83, 86, 88
  Peer-Global Optimization, 16, 17, *53*, 77, 79–81, 83–88, 120–125
  Peer-Local Optimization, 15, *30*, 59, 76–78, 81, 83, 86, 87
VRA-1N-1D, 4, 5, 19, *20–21*, 21–29, 51, 79, 83, 87
VRA-1N-mD, 4, 6, 31, 32, *32–35*, 35–37, 47–52, 87
  -Unlimited, *35*, 35–47, 51, 90–91, 108–120

WCMP, *4–5*, 6, 12, 17, 21, 28, *29*

X3C, *120*, 120–123

Appendix A

List of Theses

In this chapter I list my theses with references to the related parts of this dissertation. The citations refer to my publications listed in Appendix D.

Thesis Group 1. [C3, J1] *I have studied the possibility of enhancing load balancing schemes by unequal traffic splitting when the underlying technology only offers uniform data distribution among the resources. For this I have proposed the Virtual Resource Allocation technique that augments the load balancer to realize an almost arbitrary traffic split ratio. For the OSPF Traffic Engineering application scenario I have introduced and analyzed the Overlay Optimization method, which is a specialization of the Virtual Resource Allocation, utilizing an overlay network.*

Discussed in Chapters 2 and 3.

Thesis 1.1. *I have proposed a solution, named Overlay Optimization, to the Traffic Engineering problem in communication networks that uses end-to-end tunnels with parallel virtual paths and OSPF routing on top of this overlay network. As a part of this solution, I have formalized the Virtual Resource Allocation problem for only one network node and one demand as an optimization problem. I have shown via an example that the performance of OSPF Traffic Engineering can be enhanced by Overlay Optimization.*

See the introduction of Chapter 3 and Section 3.1. For the example see Fig. 2.1.

Thesis 1.2. *I have given bounds on the error of the VRA-1N-1D problem under different constraints.*

Discussed in Section 3.2.

Thesis 1.3. *I have given an optimal solution with pseudo-polynomial running time to the VRA-1N-1D problem. Furthermore, I have given an optimal, pseudo-polynomial time algorithm for the problem variant of minimizing the link number under a constraint on the maximal error. I have also given optimal, pseudo-polynomial time algorithms for variants of the previous two problems in which the error or link number minimization have to be done simultaneously at several nodes, while having a common constraint on the total link number or on the error, respectively.*

See Sections 3.3 and 3.4.

Thesis Group 2. [C1, J1] *I have proposed and examined in detail another Virtual Resource Allocation scheme in the OSPF Traffic Engineering scenario that eliminates the overlay network from the architecture, thereby facilitating the deployment. This approach operates on the original network topology and makes decisions locally at the network nodes, so I named it Peer-Local Optimization.*

Presented in Chapter 4.

Thesis 2.1. *I have proposed a new solution to the OSPF Traffic Engineering problem, named Peer-Local Optimization, which is not using overlays and relies solely on decisions made locally at the network nodes. As part of this solution I have formalized the Virtual Resource Allocation problem for one node and several demands as an optimization problem.*

Discussed at the introduction of Chapter 4 and in Section 4.1.

Thesis 2.2. *I have shown that matrix G, which forms the core of the VRA-1N-mD and the VRA-1N-mD-Unlimited problems, can be almost arbitrary: any nonnegative matrix with at least one non-zero element in each row and in each column can be matrix G for a given node in a suitable network.*

This corresponds to Theorem 11 in Section 4.2.1.

Thesis 2.3. *I have given bounds on the error of the VRA-1N-mD and the VRA-1N-mD-Unlimited problems along with a polynomial time algorithm that decides whether the general lower bound can be reached for a particular problem instance with unlimited number of links.*

See Sections 4.2.2 and 4.3.1.

Thesis 2.4. *I have proven that an optimal virtual link settings for VRA-1N-mD-Unlimited cannot always be reached using finite number of links.*

See Theorem 16 and Corollary 17 in Section 4.3.2.

Thesis 2.5. *I have shown that no algorithm can give an optimal solution to the VRA-1N-mD-Unlimited problem in finite number of steps; even if the number of steps may depend on the actual problem.*

See Theorem 18 and Corollary 19 in Section 4.3.2.

Thesis 2.6. *I have given an approximation algorithm that can find, in polynomial time, a solution that is arbitrarily close to the optimal solution of the VRA-1N-mD-Unlimited problem.*

See Alg. 4.2 and the related discussion in Section 4.3.3.

Thesis 2.7. *I have given two different, Integer Linear Program-based optimal solutions to the VRA-1N-mD problem. I have also given a pseudo-polynomial running time heuristic to the same problem.*

See the whole Section 4.4.

Thesis Group 3. [J1] *I have proposed and comprehensively studied another solution for OSPF Traffic Engineering, which I call Virtual Resource Allocation–Peer-Global Optimization. I have proven that it is NP-complete and cannot be approximated efficiently. I have also given an Integer Linear Program that finds an optimal solution and observed that Peer-Local Optimization can be used here as a faster heuristic.*

Presented in Chapter 5.

Thesis 3.1. *I have identified and formally described the Peer-Global Optimization problem, which can provide a near-optimal solution for OSPF Traffic Engineering.*

See Section 5.1.

Thesis 3.2. *I have given an Integer Linear Program that finds an optimal solution to the Peer-Global Optimization problem.*

See LP 5.1 presented in Section 5.2.

Thesis 3.3. *I have proven the NP-completeness of the Peer-Global Optimization problem and several of its variants.*

Discussed in Section 5.3.1.

Thesis 3.4. *I have formulated two variants of Peer-Global Optimization as NP optimization problems and have proven that it is impossible to computationally efficiently approximate their optimal solution within* every *constant ratio (unless $P = NP$).*

See the first part of Section 5.3.2, including Theorems 27 and 28.

Thesis 3.5. *I have shown that the optimal solution of the MVPGS problem (which is a variant of Peer-Global Optimization) cannot be approximated with a polynomial time algorithm within* any *constant ratio (unless $P = NP$).*

See Theorem 29 in Section 5.3.2.

Appendix B

List of Problem Definitions

Table B.1 shorty summarizes the different problem definitions given throughout this work for easier reference and comparison.

Problem	Inputs	Constraints	Outputs	Objective / Question	Page of definition
1. VRA-1N-1D	k, $\{g_i\}$, Q	$\sum_i e_i \leq Q$	$\{e_i\}$	min. U	21
2. VRA-1N-1D-Link-Min	k, $\{g_i\}$, U_{lim}	$U \leq U_{\text{lim}}$	$\{e_i\}$	min. E	27
3. Parallel-VRA-1N-1D	k_n, $\{g_{ni}\}$, Q	$\sum_n E_n \leq Q$	$\{e_{ni}\}$	min. $U_{\max}$	28
4. Parallel-VRA-1N-1D-Link-Min	k_n, $\{g_{ni}\}$, U_{lim}	$U_{\max} \leq U_{\text{lim}}$	$\{e_{ni}\}$	min. $\sum_n E_n$	29
5. VRA-1N-mD	k, D, G, Q	$E_i \leq Q$ $(i = 1 \dots D)$	$\{e_j\}$	min. U	35
6. VRA-1N-mD-Unlimited	k, D, G	–	$\{e_j\}$	min. U	35
7. VRA-PGO	(V, F), $\{c_l\}$, D, $\{O_d, D_d, G_d\}$, R	$E_n \leq \lvert S_n\rvert + R$ $(\forall n \in V)$	$\{w_l\}$, $\{e_l\}$	min. β	54
8. VRA-PGO-GW	(V, F), $\{c_l\}$, $\{w_l\}$, D, $\{O_d, D_d, G_d\}$, R, β	$E_n \leq \lvert S_n\rvert + R$ $(\forall n \in V)$, $\max_l h_l/c_l \leq \beta$	yes/no	$\exists\{e_l\}$?	59
VRA-PGO-GW-SD	(V, F), $\{c_l\}$, $\{w_l\}$, O_1, D_1, G_1, R, β	$E_n \leq \lvert S_n\rvert + R$ $(\forall n \in V)$, $\max_l h_l/c_l \leq \beta$	yes/no	$\exists\{e_l\}$?	60
VRA-PGO-GW-Q	(V, F), $\{c_l\}$, $\{w_l\}$, D, $\{O_d, D_d, G_d\}$, R, β	$E_n \leq Q$ $(\forall n \in V)$, $\max_l h_l/c_l \leq \beta$	yes/no	$\exists\{e_l\}$?	60
VRA-PGO-GW-ABS	(V, F), $\{c_l\}$, $\{w_l\}$, D, $\{O_d, D_d, G_d\}$, R, δ	$E_n \leq \lvert S_n\rvert + R$ $(\forall n \in V)$, $\max_l(h_l - c_l) \leq \delta$	yes/no	$\exists\{e_l\}$?	61
9. 3SAT	F	(F is in 3CNF)	yes/no	Is F satisfiable?	61
10. MVPG	(V, F), $\{c_l\}$, $\{w_l\}$, D, $\{O_d, D_d, G_d\}$, R	$E_n \leq \lvert S_n\rvert + R$ $(\forall n \in V)$	$\{e_l\}$	min. β	68
11. MVPGS	(V, F), $\{c_l\}$, $\{w_l\}$, O_1, D_1, G_1, R	$E_n \leq \lvert S_n\rvert + R$ $(\forall n \in V)$	$\{e_l\}$	min. β	69
12. X3C	$X = \{x_1, \dots, x_p\}$ $(p = 3q)$, $\mathcal{C} = \{C_1, \dots, C_n\}$ $(C_i \subseteq X$, $\lvert C_i\rvert = 3$, $n \geq q)$	$\mathcal{C}'$ consists of q pair-wise disjoint subsets of X	yes/no	$\exists\mathcal{C}' \subseteq \mathcal{C}$?	120
13. GSA	$g_1, \dots, g_d$, N, s_1, s_2	$1 \leq W \leq N$, $\min_{n\in\mathbb{Z}} \lvert Wg_i - n\rvert \leq s_1/s_2$ $\forall i$	yes/no	$\exists W$?	123
14. GSA-W	$g_1, \dots, g_d$, N, s_1, s_2	$1 \leq W \leq N$, $\min_{n\in\mathbb{Z}} \lvert g_i - n/W\rvert \leq s_1/s_2$ $\forall i$	yes/no	$\exists W$?	123

Table B.1: Summary of problem definitions

Appendix C

Auxiliary Proofs

C.1 Proof of Theorem 16

This section is devoted to prove

Theorem 16 (revisited). *There is at least one VRA-1N-mD-Unlimited problem, where matrix G contains integers only but the single optimal solution contains only irrational numbers as f_js.*

Before the theorem itself I prove some lemmas first. For the notations and definitions used below please refer to Sections 4.1 and 4.3.

Lemma 31. *For a matrix G in the form*

$$G = \begin{bmatrix} g_{11} & g_{12} & 0 \\ g_{21} & g_{22} & g_{23} \end{bmatrix} \quad (g_{11}g_{12}g_{21}g_{22}g_{23} \neq 0) \ , \tag{C.1}$$

and for any f_1, f_2, f_3 ($f_j > 0$, $\sum f_j = 1$), which minimizes U, both of the following statements hold:

$$U = U_{11} \vee U = U_{21} \tag{C.2}$$

$$U = U_{12} \vee U = U_{22} \tag{C.3}$$

Proof. First I prove (C.2). The proof is by contradiction: suppose that for some G (in the form of (C.1)) and $\{f_j\}$, which minimizes U, (C.2) does not hold, i.e., $U_{11} < U$

and $U_{21} < U$. I will give an f_1', f_2', f_3' triplet ($f_j' > 0$, $\sum f_j' = 1$), such that the corresponding error $U' < U$.

Let

$$\begin{aligned} f_1' &= \alpha f_1 + 1 - \alpha, \\ f_2' &= \alpha f_2, \\ f_3' &= \alpha f_3, \\ 0 &< \alpha \leq 1 \ . \end{aligned}$$

I will prove that there is a suitable α for which $U' < U$.

Recall

$$\begin{bmatrix} U_{11} & U_{12} & 0 \\ U_{21} & U_{22} & U_{23} \end{bmatrix} = \begin{bmatrix} \frac{f_1}{(f_1+f_2)\gamma_{11}} & \frac{f_2}{(f_1+f_2)\gamma_{12}} & 0 \\ \frac{f_1}{(f_1+f_2+f_3)\gamma_{21}} & \frac{f_2}{(f_1+f_2+f_3)\gamma_{22}} & \frac{f_3}{(f_1+f_2+f_3)\gamma_{23}} \end{bmatrix} =$$
$$= \begin{bmatrix} \frac{f_1}{(f_1+f_2)\gamma_{11}} & \frac{f_2}{(f_1+f_2)\gamma_{12}} & 0 \\ \frac{f_1}{\gamma_{21}} & \frac{f_2}{\gamma_{22}} & \frac{f_3}{\gamma_{23}} \end{bmatrix} ,$$

and that $U = \min U_{ij}$ ($ij \in \{11, 12, 21, 22, 23\}$). Observe

$$\begin{aligned} U_{12}' &= \frac{f_2'}{(f_1' + f_2')\gamma_{12}} = \frac{\alpha f_2}{(\alpha f_1 + 1 - \alpha + \alpha f_2)\gamma_{12}} = \frac{f_2}{(f_1 + f_2 - 1 + \frac{1}{\alpha})\gamma_{12}}, \\ U_{22}' &= \frac{f_2'}{\gamma_{22}} = \frac{\alpha f_2}{\gamma_{22}}, \\ U_{23}' &= \frac{f_3'}{\gamma_{23}} = \frac{\alpha f_3}{\gamma_{23}} \ . \end{aligned}$$

These are all strictly increasing continuous functions of $\alpha \in (0, 1]$. Also note that

$$\begin{aligned} U_{11}' &= \frac{f_1'}{(f_1' + f_2')\gamma_{11}} = \frac{\alpha f_1 + 1 - \alpha}{(\alpha f_1 + 1 - \alpha + \alpha f_2)\gamma_{11}}, \\ U_{21}' &= \frac{f_1'}{\gamma_{21}} = \frac{\alpha f_1 + 1 - \alpha}{\gamma_{21}} \end{aligned}$$

are both continuous functions of $\alpha \in (0, 1]$.

Because of the continuity and monotonicity and the fact that for $\alpha = 1$: $U_{11}' = U_{11} < U$ and $U_{21}' = U_{21} < U$, for a sufficiently small $\epsilon > 0$ having $\alpha = 1 - \epsilon$ we

will have $U'_{11} < U$ and $< U'_{21} < U$. For this α also $U'_{12} < U_{12}$, $U'_{22} < U_{22}$, $U'_{23} < U_{23}$. Therefore for the f'_1, f'_2, f'_3 triplet with the given α we have $U' < U$, which is a contradiction.

Due to symmetry, the same reasoning can be applied to prove (C.3). □

Lemma 32. *For a matrix G in the form given in* (C.1) *and for any f_1, f_2, f_3 ($f_j > 0$, $\sum f_j = 1$) minimizing U, if $U_{11} = U_{12}$ then matrix G is consistent.*

Proof. $U_{11} = U_{12}$ means

$$\frac{f_1}{(f_1 + f_2)\gamma_{11}} = \frac{f_2}{(f_1 + f_2)\gamma_{12}} \ . \tag{C.4}$$

Substituting $\gamma_{12} = 1 - \gamma_{11}$ into (C.4) we got $\gamma_{11} = f_1/(f_1 + f_2)$. Substituting this back to the definition of U_{11} we got $U_{11} = 1$, meaning also that $U_{11} = U_{12} = 1$.

According to Lemma 31 either $U = U_{11} = U_{12} = 1$ (meaning that matrix G is consistent) or $U = U_{21} = U_{22} > 1$. Let us take a closer look on this latter case. $U_{21} = U_{22}$ means $f_1/\gamma_{21} = f_2/\gamma_{22}$, but as $U_{11} = U_{12}$, by the definition of U_{11} and U_{12}, we also have $f_1/\gamma_{11} = f_2/\gamma_{12}$, i.e. $\gamma_{11}/\gamma_{12} = \gamma_{21}/\gamma_{22}$. Now because $\gamma_{11} + \gamma_{12} = 1$ we have

$$\gamma_{11} = \frac{\gamma_{21}}{\gamma_{21} + \gamma_{22}}, \ \gamma_{12} = \frac{\gamma_{22}}{\gamma_{21} + \gamma_{22}} \ ,$$

which means again that G is consistent with $f_1 = \gamma_{21}$, $f_2 = \gamma_{22}$, $f_3 = \gamma_{23}$. □

Lemma 33. *For a matrix G in the form given in* (C.1) *and for any f_1, f_2, f_3 ($f_j > 0$, $\sum f_j = 1$) minimizing U, if $U = U_{21} = U_{22} = U_{23}$ then matrix G is consistent.*

Proof. $U_{21} = U_{22} = U_{23} = U$ means

$$\frac{f_1}{\gamma_{21}} = \frac{f_2}{\gamma_{22}} = \frac{f_3}{\gamma_{23}} = U \ ,$$

i.e.,

$$\begin{aligned} f_1 &= \gamma_{21} U, \\ f_2 &= \gamma_{22} U, \\ f_3 &= \gamma_{23} U \ . \end{aligned}$$

Summing these equations and using $f_1 + f_2 + f_3 = \gamma_{21} + \gamma_{22} + \gamma_{23} = 1$ yields $U = 1$. □

Lemma 34. *For a matrix G in the form given in* (C.1) *and for any f_1, f_2, f_3 ($f_j > 0$, $\sum f_j = 1$) minimizing U: $U = U_{23}$.*

Proof. This proof is by contradiction, too. Suppose we have a G and an optimal $\{f_j\}$ for which $U_{23} < U$. Nevertheless, according to the Lemma 31 (C.2) and (C.3) must still hold. Let us divide this proof into four cases, according to how (C.2) and (C.3) can be true:

Case 1. $U = U_{21} > U_{11}$ and $U = U_{22} > U_{12}$.

The proof for this case is similar to the proof of Lemma 31. Let

$$\begin{aligned} f'_1 &= \alpha f_1, \\ f'_2 &= \alpha f_2, \\ f'_3 &= \alpha f_3 + 1 - \alpha, \\ 0 &< \alpha \leq 1 \ . \end{aligned} \tag{C.5}$$

Now $U'_{21} = f'_1/\gamma_{21} = \alpha f_1/\gamma_{21}$ is strictly increasing and continuous function of α and so is $U'_{22} = f'_2/\gamma_{22} = \alpha f_2/\gamma_{22}$. Also note that for $\alpha = 1$: $U'_{21} = U_{21} = U$ and $U'_{22} = U_{22} = U$. On the other hand $U'_{23} = f'_3/\gamma_{23} = (\alpha f_3 + 1 - \alpha)/\gamma_{23}$ is a continuous function of $\alpha \in (0, 1]$ with $U'_{23} = U_{23} < U$ for $\alpha = 1$. Also notice that $U'_{11} = U_{11} < U$ and $U'_{12} = U_{12} < U$.

This means, there is a suitable α (close to 1), for which $U' > U'_{11}$, $U' > U'_{12}$, $U' = U'_{21} < U$, $U' = U'_{22} < U$ and $U' > U'_{23}$. This means that f'_1, f'_2, f'_3 with this α result in an error $U' < U$, which is a contradiction.

Case 2. $U = U_{11} \geq U_{21}$ and $U = U_{22} > U_{12}$.

Let us use again $\{f'_i\}$ as defined in (C.5). As shown in the previous case, U'_{11} and U'_{12} is independent of α; U'_{21}, U'_{22} are continuous, increasing and U'_{23} is continuous function of $\alpha \in (0, 1]$. Furthermore, for $\alpha = 1$: $U'_{21} = U_{21} \leq U$, $U'_{22} = U_{22} = U$ and $U'_{23} = U_{23} < U$.

This means that there is a suitable α, for which $U'_{21} < U$, $U'_{22} < U$, $U'_{23} < U$, $U'_{12} = U_{12} < U$ and $U'_{11} = U_{11} = U = U'$. So $\{f'_i\}$ is minimizing the error $U' = U$ just as well as $\{f_i\}$ does, but this G, $\{f'_i\}$ setting contradicts (C.3) of Lemma 31.

Case 3. $U = U_{21} > U_{11}$ and $U = U_{12} \geq U_{22}$.

This case, by symmetry, is essentially identical to Case 2.

Case 4. $U = U_{11} \geq U_{21}$ and $U = U_{12} \geq U_{22}$.

$U_{11} = U_{12}$, so applying Lemma 32 shows that matrix G is consistent. Consequently, according to Lemma 15, $U_{23} = 1 = U$, which contradicts our assumption. □

Lemma 35. *For a matrix G in the form given in* (C.1) *and for any f_1, f_2, f_3 ($f_j > 0$, $\sum f_j = 1$) minimizing U both of the following statements hold:*

1. *$U_{11} = U_{21}$ if and only if G is consistent,*
2. *$U_{12} = U_{22}$ if and only if G is consistent.*

Proof. If G is consistent then, according to Lemma 15, $U_{11} = U_{12} = U_{21} = U_{22} = U_{23} = U = 1$, which proves the first direction of the statements.

For the other direction I start with the first statement. According to Lemma 31 $U = U_{11} = U_{21}$. Moreover, due to the same lemma $U = U_{12}$ and/or $U = U_{22}$. Let us split the proof into two cases accordingly:

If $U = U_{12}$ holds then $U = U_{11} = U_{12}$, so, according to Lemma 32, G is consistent.

If $U = U_{22}$ is true, then due to Lemma 34 $U_{23} = U$, i.e., $U = U_{21} = U_{22} = U_{23}$. Consequently, Lemma 33 can be applied, which proves the consistency of G.

Considering the symmetry, the same reasoning can be applied to prove the second direction of the second statement. □

The last lemma in this section is the following:

Lemma 36. *For an* inconsistent *matrix G in the form given in* (C.1) *and for any f_1, f_2, f_3 ($f_j > 0$, $\sum f_j = 1$) minimizing U: either $U = U_{11} = U_{22} = U_{23}$ or $U = U_{21} = U_{12} = U_{23}$ is true. Both cases are possible.*

Proof. G is inconsistent, so due to Lemma 35 $U_{11} \neq U_{21}$ and $U_{12} \neq U_{22}$. According to this and Lemmas 31 and 34, for a given G exactly one of the following statements is

true:

$$U = U_{11} = U_{12} = U_{23} \tag{C.6}$$

$$U = U_{11} = U_{22} = U_{23} \tag{C.7}$$

$$U = U_{21} = U_{12} = U_{23} \tag{C.8}$$

$$U = U_{21} = U_{22} = U_{23} \tag{C.9}$$

Statement (C.6), however, cannot be true for an inconsistent G, as it contradicts Lemma 32. Similarly, equation (C.9) would contradict Lemma 33.

What remains is to show that both (C.7) and (C.8) is possible for different Gs. Consider any inconsistent G in form given in (C.1). For an optimal $\{f_j\}$ setting either (C.7) or (C.8) will be true. Now swap the first two columns of G to get G'. Due to the symmetry $f'_1 = f_2$, $f'_2 = f_1$, $f'_3 = f_3$ is an optimal solution of G'. Consequently, if (C.7) is true for G then (C.8) is true for G' and vice versa. □

Now we are ready to put the pieces together:

Proof of Theorem 16. Consider the following matrices:

$$G = \begin{bmatrix} 2 & 1 & 0 \\ 2 & 2 & 1 \end{bmatrix}, \quad \Gamma = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} & 0 \\ \frac{2}{5} & \frac{2}{5} & \frac{1}{5} \end{bmatrix} \quad . \tag{C.10}$$

G is clearly inconsistent, so according Lemmas 35 and 36 we know that for any optimal f_1, f_2, f_3, either

$$U_{11} = U_{22} = U_{23} \text{ and } U_{21} < U_{11},\, U_{12} < U_{22}, \text{ or}$$
$$U_{21} = U_{12} = U_{23} \text{ and } U_{11} < U_{21},\, U_{22} < U_{12}.$$

I show now that the latter case holds.

Supposing the opposite, we pay attention only to $U_{12} < U_{22}$. This means that

$$\frac{f_2}{(f_1 + f_2)\frac{1}{3}} < \frac{f_2}{\frac{2}{5}} \quad .$$

Rearranging this (using $f_2 > 0$) we got $6/5 < f_1 + f_2$, which contradicts $f_1 + f_2 < 1$.

At this point we know that for any optimal f_1, f_2, f_3: $U_{21} = U_{12} = U_{23}$. This results in the following system of equations:

$$\begin{aligned}\frac{f_2}{(f_1+f_2)^{\frac{1}{3}}} &= \frac{f_1}{\frac{2}{5}},\\ \frac{f_2}{(f_1+f_2)^{\frac{1}{3}}} &= \frac{f_3}{\frac{1}{5}},\\ f_1+f_2+f_3 &= 1\ ,\end{aligned}$$

which can be reduced to:

$$\begin{aligned}5f_1^2-28f_1+12 &= 0,\\ f_2 &= 1-\frac{3}{2}f_1,\\ f_3 &= \frac{f_1}{2}\ .\end{aligned}$$

Solving this using $f_1 < 1$ we got:

$$\begin{aligned}f_1 &= \frac{2}{5}(7-\sqrt{34}),\\ f_2 &= \frac{1}{5}(-16+3\sqrt{34}),\\ f_3 &= \frac{1}{5}(7-\sqrt{34})\ ,\end{aligned}$$

which, due to the reasoning above, is the only optimal solution to the problem given with (C.10). Consequently, for the single optimal solution f_1, f_2, f_3 are irrational. □

C.2 Proof of Theorem 18

In this section I prove

Theorem 18 (revisited). *There is at least one VRA-1N-mD-Unlimited problem, whose only optimal solution contains at least one f_j that cannot be written in a finite form using integer constants and the usual $+$, $-$, $\cdot$, $/$ and the nth root ($n \in \mathbb{Z}^+$) operators only.*

A suitable example is enough to prove this theorem. Throughout the proof and the related lemmas I will be using the VRA-1N-mD-Unlimited problem defined by the following matrices:

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 6 & 1 & 0 & 0 & 0 & 0 \\ 6 & 6 & 1 & 0 & 0 & 0 \\ 6 & 6 & 6 & 1 & 0 & 0 \\ 6 & 6 & 6 & 6 & 1 & 0 \\ 6 & 6 & 6 & 6 & 6 & 1 \end{bmatrix}, \quad \gamma = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{6}{7} & \frac{1}{7} & 0 & 0 & 0 & 0 \\ \frac{6}{13} & \frac{6}{13} & \frac{1}{13} & 0 & 0 & 0 \\ \frac{6}{19} & \frac{6}{19} & \frac{6}{19} & \frac{1}{19} & 0 & 0 \\ \frac{6}{25} & \frac{6}{25} & \frac{6}{25} & \frac{6}{25} & \frac{1}{25} & 0 \\ \frac{6}{31} & \frac{6}{31} & \frac{6}{31} & \frac{6}{31} & \frac{6}{31} & \frac{1}{31} \end{bmatrix}. \tag{C.11}$$

For an optimal solution $f_1, f_2, \ldots, f_6$ the error matrix is:

$$U_{ij} = \begin{bmatrix} U_{11} & 0 & 0 & 0 & 0 & 0 \\ U_{21} & U_{22} & 0 & 0 & 0 & 0 \\ U_{31} & U_{32} & U_{33} & 0 & 0 & 0 \\ U_{41} & U_{42} & U_{43} & U_{44} & 0 & 0 \\ U_{51} & U_{52} & U_{53} & U_{54} & U_{55} & 0 \\ U_{61} & U_{62} & U_{63} & U_{64} & U_{65} & U_{66} \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{f_1}{(f_1+f_2)\frac{6}{7}} & \frac{f_2}{(f_1+f_2)\frac{1}{7}} & 0 & 0 & 0 & 0 \\ \frac{f_1}{(f_1+f_2+f_3)\frac{6}{13}} & \frac{f_2}{(f_1+f_2+f_3)\frac{6}{13}} & \frac{f_3}{(f_1+f_2+f_3)\frac{1}{13}} & 0 & 0 & 0 \\ \frac{f_1}{(f_1+\cdots+f_4)\frac{6}{19}} & \frac{f_2}{(f_1+\cdots+f_4)\frac{6}{19}} & \frac{f_3}{(f_1+\cdots+f_4)\frac{6}{19}} & \frac{f_4}{(f_1+\cdots+f_4)\frac{1}{19}} & 0 & 0 \\ \frac{f_1}{(f_1+\cdots+f_5)\frac{6}{25}} & \frac{f_2}{(f_1+\cdots+f_5)\frac{6}{25}} & \frac{f_3}{(f_1+\cdots+f_5)\frac{6}{25}} & \frac{f_4}{(f_1+\cdots+f_5)\frac{6}{25}} & \frac{f_5}{(f_1+\cdots+f_5)\frac{1}{25}} & 0 \\ \frac{f_1}{(f_1+\cdots+f_6)\frac{6}{31}} & \frac{f_2}{(f_1+\cdots+f_6)\frac{6}{31}} & \frac{f_3}{(f_1+\cdots+f_6)\frac{6}{31}} & \frac{f_4}{(f_1+\cdots+f_6)\frac{6}{31}} & \frac{f_5}{(f_1+\cdots+f_6)\frac{6}{31}} & \frac{f_6}{(f_1+\cdots+f_6)\frac{1}{31}} \end{bmatrix}.$$

Furthermore, I will denote with $U_{\max}$ the set consisting of the maximal elements of U_{ij} for the given optimal solution. Certainly the value of the maximal elements is the per node error, U, as defined at Sec. 4.1.

Before the actual proof I present two lemmas.

Lemma 37. *For any optimal solution of the VRA-1N-mD-Unlimited problem given by the matrix* (C.11)*, at least one element of the set* $\{U_{21}, U_{31}, U_{41}, U_{51}, U_{61}\}$ *is an element of* U_{max}*.*

Proof. The proof is by contradiction. We suppose the opposite, yet give $f'_1, f'_2, \ldots, f'_6$, for which $U' < U$. Let:

$$\begin{aligned} f'_1 &= \alpha f_1 + 1 - \alpha, \\ f'_2 &= \alpha f_2, \\ f'_3 &= \alpha f_3, \\ f'_4 &= \alpha f_4, \\ f'_5 &= \alpha f_5, \\ f'_6 &= \alpha f_6, \\ 0 &< \alpha \leq 1 \ . \end{aligned}$$

Let us first examine U'_{i1} $(i = 2, \ldots, 6)$:

$$U'_{i1} = \frac{f'_1}{(f'_1 + \cdots + f'_i)\gamma_{i1}} = \frac{\alpha f_1 + 1 - \alpha}{(\alpha(f_1 + \cdots + f_i) + 1 - \alpha)\gamma_{i1}} \ .$$

U'_{i1} is clearly a continuous function of $\alpha \in (0, 1]$, with $U'_{i1} = U_{i1}$ for $\alpha = 1$.

Now let us see U'_{ij} $(j > 1,\ U'_{ij} > 0)$:

$$U'_{ij} = \frac{f'_j}{(f'_1 + \cdots + f'_i)\gamma_{ij}} = \frac{\alpha f_j}{(\alpha(f_1 + \cdots + f_i) + 1 - \alpha)\gamma_{ij}} = \frac{f_j}{(f_1 + \cdots + f_i + \frac{1}{\alpha} - 1)\gamma_{ij}} .$$

U'_{ij} is again a continuous function of $\alpha \in (0, 1]$, with $U'_{ij} = U_{ij}$ for $\alpha = 1$. It can also be seen that U'_{ij} is a strictly increasing function of $\alpha \in (0, 1]$.

Because of the monotonicity and continuity described above, for a sufficiently small $\epsilon > 0$ having $\alpha = 1 - \epsilon$ we will have $U'_{i1} < U$ $(i = 2, \ldots, 6)$ (as we indirectly supposed $U_{i1} < U$ and $U'_{i1} = U_{i1}$ for $\alpha = 1$) and $U'_{ij} < U$ $(j > 1,\ U'_{ij} > 0)$ (due to the monotonicity and the limit value at $\alpha = 1$). This, on the other hand, means that $U' < U$, which contradicts the optimality of the initial solution. □

Lemma 38. *For any optimal solution of the VRA-1N-mD-Unlimited problem given by the matrix* (C.11)*, the largest elements of the 1st, 2nd, . . . , 6th columns of matrix* U_{ij} *are* U_{61}*,* U_{22}*,* U_{33}*,* U_{44}*,* U_{55}*,* U_{66}*, respectively.*

Proof. First I prove that $U < 6$ for any optimal solution of the given problem. To do so I show that $U < 6$ for $f_1 = f_2 = \cdots = f_6 = 1/6$. If $f_1 = f_2 = \cdots = f_6$, then

clearly in each row of U_{ij} the biggest element is the rightmost one. The value of these elements are: $U_{22} = \frac{7}{2}$, $U_{33} = \frac{13}{3}$, $U_{44} = \frac{19}{4}$, $U_{55} = \frac{25}{5}$, $U_{66} = \frac{31}{6}$, and these are all less than 6. This means that for any *optimal* solution, which certainly cannot be worse than this particular one, $U < 6$.

Let us now consider the first row of matrix U_{ij}. I first show that $U_{21} < U_{31}$. Suppose the opposite:

$$\frac{f_1}{(f_1+f_2)\frac{6}{7}} \geq \frac{f_1}{(f_1+f_2+f_3)\frac{6}{13}} \ ,$$

which can be rearranged to:

$$6 \leq \frac{f_3}{(f_1+f_2+f_3)\frac{1}{13}} = U_{33} \ .$$

This means that $U \geq U_{33} \geq 6$, which contradicts the optimality according to our first statement ($U < 6$).

Using the very same steps, $U_{31} < U_{41}$, $U_{41} < U_{51}$, $U_{51} < U_{61}$ can be shown: supposing the opposite would yield $U_{44} \geq 6$, $U_{55} \geq 6$, $U_{66} \geq 6$, respectively.

With this we have shown that U_{61} is the largest element in the first column. Let us now move on to the second one.

First I show that $U_{22} > U_{32}$:

$$\frac{f_2}{(f_1+f_2)\frac{1}{7}} > \frac{f_2}{(f_1+f_2+f_3)\frac{6}{13}} \ ,$$

which can be rewritten as

$$13(f_1+f_2) < 42(f_1+f_2+f_3) \ ,$$

which is true as by our assumption $f_1, \ldots, f_6 > 0$.

In a similar way it can be shown that $U_{22} > U_{42}$, $U_{22} > U_{52}$, $U_{22} > U_{62}$ (concluding the second column), $U_{33} > U_{43}$, $U_{33} > U_{53}$, $U_{33} > U_{63}$ (third column), $U_{44} > U_{54}$, $U_{44} > U_{64}$ (fourth column), $U_{55} > U_{65}$ (sixth column). □

Using these results we can return to the main problem of this section:

Proof of Theorem 18. First I prove that $U_{\max} = \{U_{61}, U_{22}, U_{33}, U_{44}, U_{55}, U_{66}\}$ for any optimal solution of the VRA-1N-mD-Unlimited problem given by the matrix (C.11).

From Lemma 37 and Lemma 38 follows that $U_{61} \in U_{\max}$. From Lemma 38 follows also that the only possible other members of $U_{\max}$ are U_{22}, U_{33}, U_{44}, U_{55}, U_{66}.

Now I show that $U_{22} \in U_{\max}$. Suppose the opposite, that we have an optimal set of f_js, where $U_{22} < U$. Let as define a new set of f_js as:

$$\begin{aligned} f'_2 &= \alpha f_2 + 1 - \alpha, \\ f'_j &= \alpha f_j, \ \ (j = 1, 3, 4, 5, 6) \\ 0 &< \alpha \leq 1 \ . \end{aligned}$$

In the same fashion as shown at the proof of Lemma 37, it can be shown that U'_{22} is a continuous function of $\alpha \in (0, 1]$, with $U'_{22} = U_{22}$ for $\alpha = 1$. It can also be shown similarly that U_{61}, U_{33}, U_{44}, U_{55}, U_{66}, which are the possible elements of $U_{\max}$, are continuous and strictly increasing function of $\alpha \in (0, 1]$. Because of this monotonicity and continuity, for a sufficiently small $\epsilon > 0$ having $\alpha = 1 - \epsilon$ we will have $U'_{22} < U$ and $U'_{ij} < U_{ij} \leq U$ $(ij = 61, 33, 44, 55, 66)$, which contradicts the optimality of the initial solution.

Next, I show in two steps that $U_{33} \in U_{\max}$. Similarly to the previous paragraph, let us suppose the opposite, namely for an optimal f_j setting $U_{33} < U$.
Step 1. Let us have

$$\begin{aligned} f'_3 &= \alpha f_3 + 1 - \alpha, \\ f'_j &= \alpha f_j, \ \ (j = 1, 2, 4, 5, 6) \\ 0 &< \alpha \leq 1 \ . \end{aligned}$$

For a sufficiently small $\epsilon > 0$ having $\alpha = 1 - \epsilon$: $U'_{33} < U$ and $U'_{61} < U_{61}$, $U'_{44} < U_{44}$, $U'_{55} < U_{55}$, $U'_{66} < U_{66}$, but $U'_{22} = U_{22}$.
Step 2. Let

$$\begin{aligned} f''_1 &= \beta f'_1 + 1 - \beta, \\ f''_j &= \beta f'_j, \ \ (j = 2, 3, 4, 5, 6) \\ 0 &< \beta \leq 1 \ . \end{aligned}$$

According to similar considerations as above, for a β that is sufficiently close to 1: $U''_{61} < U_{61}$, $U''_{22} < U'_{22} = U_{22}$, $U''_{33} < U'_{33} < U$, $U''_{44} < U'_{44} < U_{44}$, $U''_{55} < U'_{55} < U_{55}$, $U''_{66} < U'_{66} < U_{66}$. According to Lemma 38 these are the possible candidate elements of $U_{\max}$, so $U'' < U$, which is a contradiction again.

The idea of the proof in the last paragraph can be easily reused to prove that $U_{44} \in U_{\max}$, $U_{55} \in U_{\max}$ and $U_{66} \in U_{\max}$.

I have just proven that for any optimal solution of the problem given by (C.11): $U = U_{61} = U_{22} = U_{33} = U_{44} = U_{55} = U_{66}$. Using this we can set up the following system of equations:

$$\begin{aligned}
\frac{f_2}{(f_1+f_2)^{\frac{1}{7}}} &= \frac{f_3}{(f_1+f_2+f_3)^{\frac{1}{13}}} \\
\frac{f_2}{(f_1+f_2)^{\frac{1}{7}}} &= \frac{f_4}{(f_1+f_2+f_3+f_4)^{\frac{1}{19}}} \\
\frac{f_2}{(f_1+f_2)^{\frac{1}{7}}} &= \frac{f_5}{(f_1+f_2+f_3+f_4+f_5)^{\frac{1}{25}}} \\
\frac{f_2}{(f_1+f_2)^{\frac{1}{7}}} &= \frac{f_6}{(f_1+f_2+f_3+f_4+f_5+f_6)^{\frac{1}{31}}} \\
\frac{f_2}{(f_1+f_2)^{\frac{1}{7}}} &= \frac{f_1}{(f_1+f_2+f_3+f_4+f_5+f_6)^{\frac{6}{31}}} \\
1 &= f_1+f_2+f_3+f_4+f_5+f_6 \ .
\end{aligned}$$

From these equations f_2, f_3, f_4, f_5, f_6, can be eliminated, and what remains is a polynomial of f_1:

$$\begin{aligned}
923\,521 f_1^5 - 16\,980\,870 f_1^4 + 118\,664\,280 f_1^3 - \\
- 390\,577\,680 f_1^2 + 934\,673\,904 f_1 - 336\,117\,600 = 0 \quad \text{(C.12)}
\end{aligned}$$

I used the mathematical software Maple [55] to show that this polynomial equation has got a single real root only (and four complex ones).

Now a little algebra follows [56]. According to the Abel–Ruffini theorem, there is no general algebraic solution to polynomial equations of degree five or higher. This does not mean, however, that no polynomial equation of degree five or more can be

solved by radicals[1]: $x^5 = 1$ for example is pretty easy to solve. On the other hand, there are polynomials that cannot be solved, such as $x^5 - x + 1 = 0$. According to Galois theory, a polynomial equation can be solved by radicals if and only if its Galois group is a solvable group.

Using Maple I found that the Galois group of the polynomial given in (C.12) is the symmetric group S_5. This group, consisting of 120 elements, is not solvable ([56], p. 125), meaning that (C.12) cannot be solved by radicals, proving Theorem 18. □

C.3 Addendum on Computation Complexity

In this section I present alternative proofs and their corollaries about the computational complexity of the VRA-PGO related problems. They not only represent different approaches, but also tackle different variants of the same problem family. As an example, Theorem 39 states the NP-completeness of the original VRA-PGO problem itself.

C.3.1 X3C Reduction

Theorem 22 (revisited). *VRA-PGO-GW is NP-complete.*

Proof 2 of Theorem 22. This proof is partially based on the idea presented in [68].

I will not prove again that VRA-PGO-GW is in NP, only the more interesting part, namely that VRA-PGO-GW is NP-hard. I will reduce the X3C (Exact Cover by 3-Sets) problem to VRA-PGO-GW. X3C is NP-complete, and is defined as follows:

Problem 12, Exact Cover by 3-Sets (X3C).

INSTANCE. *Set $X = \{x_1, \ldots, x_p\}$ of $p = 3q$ elements and a family $\mathcal{C}$ of n 3-subsets of X ($\mathcal{C} = \{C_1, \ldots, C_n\}, C_i \subseteq X, |C_i| = 3, i = 1, \ldots, n$), $n \geq q$.*

QUESTION. *Does $\mathcal{C}$ contain a subfamily $\mathcal{C}' \subseteq \mathcal{C}$, consisting of q pair-wise disjoint subsets of X ($|\mathcal{C}'| = q$)?*

For an arbitrary instance of X3C I create an instance of VRA-PGO-GW corresponding to it. The *network* is shown in Figure C.1: the nodes in the second and

[1] i.e., having a solution that can be written in a finite form using integer constants and the $+$, $-$, $\cdot$, / and the nth root ($n \in \mathbb{Z}^+$) operators only

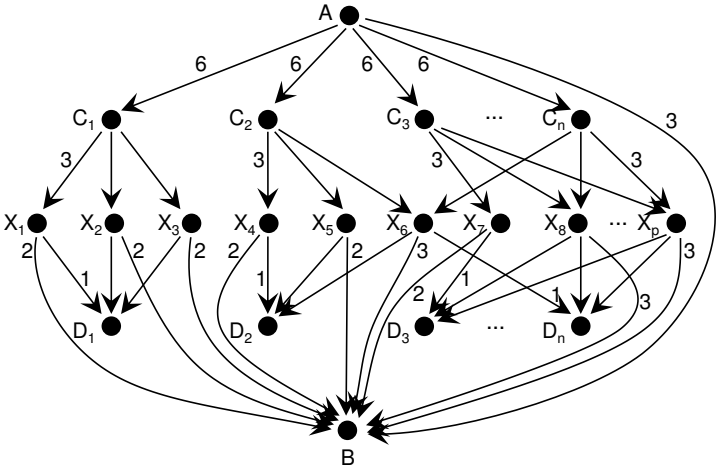


Figure C.1: Network for the X3C reduction

fourth row correspond to the 3-subsets of X in the X3C problem and the nodes in the third row correspond to the elements of X. The *link capacities* are shown in the figure, which are 3 for all links C_iX_j and are 1 for all links X_jD_i. For links X_jB the link capacity equals to the number of edges arriving to node X_j plus one. The *link weights* are one unit for all of the links but link AB, for which $w_{AB} = 3$. The *demands* are the following: $A \to B : 3(n+q+1)$, $C_i \to D_i : 3$, $i = 1 \ldots n$. The *maximal number of virtual links* is $R = q$. The MLU is $\beta = 1$. This reduction is polynomial.

I now prove that if the X3C instance contains a suitable $\mathcal{C}'$ subfamily, then there is a suitable virtual link allocation for the VRA-PGO-GW problem. Let $C'_1, \ldots, C'_q$ be the members of $\mathcal{C}'$. Let us assign $e_l = 1$ for all the links, except links AC'_i, for which $e_{AC'_i} = 2$. Constraints $E_n \leq |S_n| + R$ are trivially not violated.

The $\max_l h_l/c_l \leq \beta = 1$ is not violated either. For node A the total traffic is $3(n+q+1)$, which is split onto exactly $n+q+1$ links, resulting in 3 units of traffic per link. As $e_{AC_i} \leq 2$, $h_{AC_i}/c_{AC_i} \leq 1$. For $C_i \to X_j \to D_i$ there is an extra one unit of traffic from the $C_i \to D_i$ demand. This means that for $C_i \in \mathcal{C}'$ $h_{C_iX_j}/c_{C_iX_j} = 1$, and for $C_i \notin \mathcal{C}'$ $h_{C_iX_j}/c_{C_iX_j} = 2/3$. It can easily be seen that for the rest of the links $h_l/c_l = 1$.

Let us now see the opposite direction: if there is a suitable virtual link allocation for the VRA-PGO-GW problem, then the X3C instance contains a suitable $\mathcal{C}'$ subfamily. First observe the demands $C_i \to D_i : 3$. Suppose C_i is connected to X_k, X_l and X_m. Note that the demand, in order to be routed without violating the constraint

$\max_l h_l/c_l \leq \beta = 1$ at links X_kD_i, X_lD_i, and X_mD_i, must be split equally at nodes C_i, i.e., $e_{C_iX_k} = e_{C_iX_l} = e_{C_iX_m}$. Next, note that no traffic split occurs at nodes X_j due to the different destinations of the demands.

The most interesting traffic split occurs at node A. Observe that the volume of demand $A \to B : 3(n+q+1)$ equals the total capacity of the incoming links to B. This means that all the links arriving to node B must be fully filled. This is true for link AB as well, meaning that the traffic volume arriving to node A (i.e. $3(n+q+1)$) must be split onto $n+q+1$ links to have 3 units of traffic on AB. Consequently, a link AC_i carries $3e_{AC_i}$ traffic. Thus $e_{AC_i} > 2$ would result a $h_l/c_l > 1$ on the given link, so it is not possible. Hence there must be q virtual links distributed over (real) links AC_i (i.e., $e_{AC_i} = 2$ for q links, and $e_{AC_i} = 1$ for $n-q$). To avoid the overloading of links X_jB, the AC_is, for which $e_{AC_i} = 2$, are selected in a way that actually solves the corresponding X3C problem: $C_i \in \mathcal{C}'$ in the X3C problem if and only if $e_{AC_i} = 2$ in VRA-PGO-GW. □

This proof can easily be extended to different variants of the problem as follows. The first one is especially interesting as it states the NP-completeness of the original version of the problem.

Theorem 39. *VRA-PGO is NP-complete.*

Note that for $R = 0$ the problem is reduced to a weight-searching problem, which has been proven to be NP-complete in [9]. For $R > 0$, however, the statement is yet to be proven.

Proof. VRA-PGO is in NP, for the same reason as VRA-PGO-GW is. I use the same reduction as in Proof 2 of Theorem 22. If the X3C instance contains a suitable $\mathcal{C}'$ subfamily, then VRA-PGO is solvable: the solution is the same as above, with link weights being one unit for all of the links but link AB, for which $w_{AB} = 3$.

The opposite direction is pretty easy, too: notice that in order to fully transfer all the demands without exceeding the link capacities on the links incoming to nodes D_i and B, all the links of the network must be utilized. This can be achieved with essentially the same weight setting as above: $\forall l\ w_l = c$, except for $w_{AB} = 3c$ for any $c > 0$. After this the proof is the same as Proof 2 of Theorem 22. □

Theorem 25 (VRA-PGO-GW-ABS is NP-complete) can be proven here, too. The proof is essentially the same as the above proof of Thm. 22, but using $\delta = 0$ in place of $\beta = 1$.

Likewise, Theorem 23 (VRA-PGO-GW-Q is NP-complete) can be proven here, too: using $Q = n + q + 1$ instead of $R = q$, the proof is almost the same as above. Only in the first part of the proof "Constraints $E_n \leq |S_n| + R$ are trivially not violated" should be replaced by the following: "Constraints $E_{dn} \leq Q$ are not violated: it is trivial for node A, and also for nodes C_i (as $n, q \geq 1$). For nodes X_i: $E_{dX_i} = |S_{X_i}| = |T_{X_i}| + 1 \leq n + 1 < Q$". Note, however, that the Q used here can be fairly large.

C.3.2 Good Simultaneous Approximation Reduction

In this section VRA-PGO-GW-ABS is examined again, but with $\delta > 0$. My conjecture is that this case is NP-complete, too, but I will only prove a weaker statement. For this I first show the definition of the Good Simultaneous Diophantine Approximation problem (from [69], but with slightly modified notations):

Problem 13, Good Simultaneous (Diophantine) Approximation (GSA).

INSTANCE. *A finite vector of rationals $g_1, \ldots, g_d$ and positive integers N, s_1, s_2.*

QUESTION. *Is there an integer W with $1 \leq W \leq N$ such that*

$$\min_{n \in \mathbb{Z}} |W g_i - n| \leq \frac{s_1}{s_2} \quad \forall i \ ?$$

The NP-completeness of the GSA is proven by J. C. Lagarias in [69]. I will use another problem for the proof, which is a variation of GSA:

Problem 14, Good Simultaneous (Diophantine) Approximation in a Weaker Sense (GSA-W).

INSTANCE. *A finite vector of rationals $g_1, \ldots, g_d$ and positive integers N, s_1, s_2.*

QUESTION. *Is there an integer W with $1 \leq W \leq N$ such that*

$$\min_{n \in \mathbb{Z}} |g_i - \frac{n}{W}| \leq \frac{s_1}{s_2} \quad \forall i \ ?$$

As far as I know, the NP-completeness of GSA-W is not proven, but my conjecture is that it is true. Consequently, the theorem I prove is:

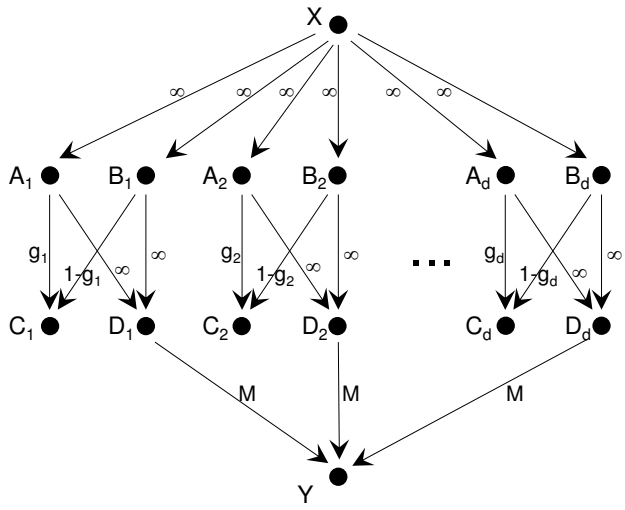


Figure C.2: Network for the GSA-W Reduction

Theorem 40. *VRA-PGO-GW-ABS with $\delta > 0$ is NP-complete if GSA-W is NP-complete.*

Proof. VRA-PGO-GW-ABS is in NP, and the proof is essentially the same as at Theorem 22. Next I prove by Karp reduction that VRA-PGO-GW-ABS is NP-hard if GSA-W is NP-complete. Suppose we have a GSA-W instance. We can safely suppose that for all i $g_i < 1$.

Consider the *network* in Fig. C.2 with the *link capacities* shown, where $M = dN$. Let $w_l = 1$ be the *weights* for all the links and the *demands* be:

$$X \to Y : dM$$
$$X \to C_i : 1 \quad (i = 1, \ldots, d) \ .$$

Let the *max. absolute error* be $\delta = s_1/s_2$ and the *resource bound*: $R = (N-2)d$ (or $Q = Nd$). (Note that R or Q can be pretty large here.) The reduction is polynomial.

I first show that if GSA-W is solvable then the given VRA-PGO-GW-ABS instance can be solved as well. Let the solution of the GSA-W be W. For $i = 1, \ldots, d$ let $f_i = \arg\min_{n \in \mathbb{Z}} |g_i - n/W|$. Let $e_{XA_i} = f_i$ and $e_{XB_i} = W - f_i$ for $i = 1 \ldots d$ and for the rest of the links let $e_l = 1$.

The condition $E_n \leq |S_n| + R$ (or $E_{dn} \leq Q$) is only interesting at node X. There $E_X = Wd$, and as $|S_n| + R = 2d + (N-2)d$ and as $W \leq N$, the constraint is not

violated. (Similarly, for $d : X \to Y$, $n = X$: $E_{dn} = Wd \leq Nd = Q$. For the other demands and/or nodes the inequality is trivial.)

The next question is whether $\max_l(h_l - c_l) \leq \delta$. Certainly we only have to consider edges with $c_l < \infty$. For edges $l = D_iY$: $h_l - c_l = M - M = 0 < \delta$. For edges $l = A_iC_i$: $h_l - c_l \leq \delta$ is to be proven, where $h_l = f_i/W$, $c_l = g_i$. If $h_l - c_l = f_i/W - g_i$ is negative, then it is clearly less than δ. If not, we can use $f_i/W - g_i = |f_i/W - g_i|$, which is less than or equal to δ by the initial conditions. The reasoning for the B_iC_i edges is very similar.

I show now that a solution of VRA-PGO-GW-ABS solves GSA-W, too. For $\delta \geq 1$ the solution is trivial, so let us consider $\delta < 1$. First observe that for all $i, j \in \{1, \ldots, d\}$ $e_{XA_i} + e_{XB_i} = e_{XA_j} + e_{XB_j}$, which is required for $h_{D_iY} - c_{D_iY} = 0$. I show this by contradiction, supposing the opposite, i.e., the difference is positive for at least one i:

$$h_{D_iY} - c_{D_iY} > 0 \Rightarrow \frac{dM(e_{XA_i} + e_{XB_i})}{E_X} - M > 0 \Rightarrow$$
$$\Rightarrow d(e_{XA_i} + e_{XB_i}) > E_X \Rightarrow d(e_{XA_i} + e_{XB_i}) \geq E_X + 1 \ .$$

Using this we got:

$$h_{D_iY} - c_{D_iY} = \frac{dM(e_{XA_i} + e_{XB_i})}{E_X} - M \geq M\left(\frac{E_X + 1}{E_X} - 1\right) =$$
$$= \frac{M}{E_X} \geq \frac{M}{|S_X| + R} = \frac{M}{Nd} = 1 \ .$$

This would then mean a higher link error than δ, a contradiction, so $e_{XA_i} + e_{XB_i} = e_{XA_j} + e_{XB_j}$ is proven.

Let us now consider links A_iC_i and B_iC_i. Supposing that VRA-PGO-GW-ABS is solvable, the error on the links is at most δ. Let $W = E_x/d = e_{XA_i} + e_{XB_i}$, which exists, according to the previous paragraph. This means:

$$h_{A_iC_i} - c_{A_iC_i} = \frac{e_{XA_i}}{W} - g_i,$$
$$h_{B_iC_i} - c_{B_iC_i} = \frac{W - e_{XA_i}}{W} - (1 - g_i) = -(h_{A_iC_i} - c_{A_iC_i}) \ .$$

Consequently, $|h_{A_iC_i} - c_{A_iC_i}| = |h_{B_iC_i} - c_{B_iC_i}| \leq \delta$, therefore W is a solution of GSA-W. □

Appendix D

Publications

International journals

[J1] **Krisztián Németh**, Attila Kőrösi, Gábor Rétvári. "Optimal Resource Pooling over Legacy Equal-Split Load Balancing Schemes", *Computer Networks*, 127, Nov. 2017, pp. 243-265.

[J2] Nikolett Bereczky, Amalia Duch, **Krisztián Németh**, Salvador Roura. "Quad-kd trees: A general framework for kd trees and quad trees", *Theoretical Computer Science*, 616, Feb. 2016, pp. 126-140.

[J3] Péter Füzesi, **Krisztián Németh**, Niklas Borg, Rikard Holmberg, István Cselényi. "Provisioning of QoS enabled inter-domain services", *Computer Communications*, 26 (10), Jun. 2003, pp. 1070-1082.

[J4] Gábor Fehér, **Krisztián Németh**, István Cselényi. "Performance Evaluation Framework for IP Resource Reservation Signaling", *Performance Evaluation*, 48 (1-4), May 2002, pp. 131-156.

Hungarian journals

[J5] **Németh Krisztián**. "Hívásengedélyezés garantált minőségű hálózatokban (áttekintés)", *Híradástechnika*, ISSN: 0018-2028, LVII, 2002/9, pp. 2-4.

[J6] Cselényi István, Füzesi Péter, **Németh Krisztián**. "Az internet szolgálatminőség fejlődése", *Magyar Távközlés*, ISSN: 0865-9648, 2000/2, pp. 26-31.

[J7] **Krisztián Németh**. "IP Multicasting over ATM", *Magyar Távközlés, Selected Papers*, ISSN 0865-9648, 1999, pp. 11-15.

[J8] **Németh Krisztián**. "IP multicast ATM felett", *Magyar Távközlés*, ISSN: 0865-9648, 1998/7, pp. 3-6.

International conferences

[C1] **Krisztián Németh**, Attila Kőrösi, Gábor Rétvári. "Enriching the poor man's traffic engineering: Virtual link provisioning for optimal OSPF TE", *Networks 2014*, Funchal, Madeira Island, Portugal, September 2014.

[C2] Nikolett Bereczky, Amalia Duch, **Krisztián Németh**, Salvador Roura. "Quad-K-d Trees" *Latin American Theoretical INformatics* (LATIN 2014), Montevideo, Uruguay, March 31 - April 4, 2014, Proc. LNCS 8392, pp. 743-754.

[C3] **Krisztián Németh**, Attila Kőrösi, Gábor Rétvári. "Optimal OSPF Traffic Engineering using Legacy Equal Cost Multipath Load Balancing", *IFIP Networking 2013*, Brooklyn, New York, USA, May 2013

[C4] Tibor Cinkler, Réka Kosznai, Péter Balázs Soproni, **Krisztián Németh**. "GSP, the Generalised Shared Protection" *9th International Conference on Design of Reliable Communication Networks* (DRCN 2013), Budapest, Hungary, March 2013.

[C5] **Krisztián Németh**, Gábor Rétvári. "Traffic Splitting Algorithms in Multipath Networks: Is the Present Practice Good Enough?", *Networks 2012*, Rome, Italy, October 2012.

[C6] Zsolt Kovácsházi, Gábor Papp, **Krisztián Németh**. "A Hybrid Multicast Video Distribution Method: a Technology for E-Entertainment", *2005 Networking and Electronic Commerce Research Conference* (NAEC 2005), Riva del Garda, Italy, October 2005, pp. 1-11.

[C7] Niklas Borg, Rikard Holmberg, Péter Füzesi, **Krisztián Németh**. "NAIS – Network Architecture for Inter-Domain Services", *Networks 2002*, Munich, Germany, June 2002. Proceedings ISBN: 3-8007-2711-0

[C8] **Krisztián Németh**, Péter Füzesi. "An Analysis of IP Resource Reservation Protocols", *Networks 2002*, Munich, Germany, June 2002. Proceedings ISBN: 3-8007-2711-0, pp. 213-220

[C9] István Moldován, **Krisztián Németh**. "Quality of Service Architectures Using MPLS Networks", *9th IFIP Working Conference on Performance Modelling and Evaluation of ATM and IP Networks*, Budapest, Hungary, June 2001. Proceedings ISBN 963 420 694 8

[C10] István Moldován, **Krisztián Németh**, Tibor Cinkler. "Merging in MPLS Networks", *IEEE ICT 2001*, Bucharest, Romania, June 2001

[C11] Csaba Simon, **Krisztián Németh**, Sándor Székely. "Point-to-Multipoint ATM Signalling Performance Measurements", *8th IFIP Workshop on Performance Modelling and Evaluation of ATM and IP Networks*, Ilkley, UK, July 2000. Proceedings: Networks UK, D. D. Kouvatsos (Ed.) ISBN 0-9540151-1-8

[C12] **Krisztián Németh**, Gábor Fehér, István Cselényi. "Simulation Study for IP Resource Reservation", *8th IFIP Workshop on Performance Modelling and Evaluation of ATM and IP Networks*, Ilkley, UK, July 2000. Proceedings: Networks UK, D. D. Kouvatsos (Ed.) ISBN 0-9540151-1-8

[C13] Gábor Fehér, **Krisztián Németh**, István Cselényi. "Router Benchmarking Framework for QoS Signaling", *8th IFIP Workshop on Performance Modelling and Evaluation of ATM and IP Networks*, Ilkley, UK, July 2000. Proceedings: Networks UK, D. D. Kouvatsos (Ed.) ISBN 0-9540151-1-8

[C14] István Cselényi, Gábor Fehér, **Krisztián Németh**. "Benchmarking of Signaling Based Resource Reservation in the Internet", *Networking 2000*, IFIP-TC6, Paris, France, May 2000. Proceedings: LNCS 1815; Networking 2000: Broadband Communications, High Performance Networking and Performance of Communication Networks; ed. Guy Pujolle et al., ISSN 0302-9743, ISBN 3-540-67506-X

[C15] **Krisztián Németh**, Krzysztof Szarkowicz. "IP Multicasting over ATM", *7th IFIP Workshop on Performance Modelling and Evaluation of ATM and IP Networks*, IFIP WG 6.3, 6.4, Antwerp, Belgium, June 28-30, 1999

[C16] Gábor Fehér, **Krisztián Németh**, Markosz Maliosz, István Cselényi, Joakim Bergkvist, David Ahlard, Tomas Engborg. "Boomerang – A Simple Protocol for Resource Reservation in IP Networks", *IEEE Workshop on QoS Support for Real Time Internet Applications in conjunction with Real-Time Technology and Applications Symposium* (RTAS), Vancouver, Canada, June 2-4, 1999

Hungarian conferences

[C17] **Németh Krisztián**. "Hívásengedélyezés garantált minőségű csomagkapcsolt hálózatokban", *PKI Tudományos Napok 111*, Budapest, November 2002.

Other publications

[O1] Attila Kőrösi, **Krisztián Németh**. "Methods and packet network devices for forwarding packet data traffic", *International Patent Application*, Int. Application No.: PCT/EP2013/060404, International Filing Date: 21 May, 2013, Publication No.: WO/2014/187475, Publication Date: 27 Nov., 2014

[O2] Gábor Fehér, **Krisztián Németh**, András Korn, István Cselényi. "Benchmarking Terminology for Resource Reservation Capable Routers", *IETF RFC* 4883, July 2007