Computer Networks 127 (2017) 243–265

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Optimal resource pooling over legacy equal-split load balancing schemes

Krisztián Németh[a,*], Attila Kőrösi[a,b], Gábor Rétvári[a,b]

[a] Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics, Magyar tudósok krt. 2, Budapest 1117, Hungary

[b] MTA-BME Information Systems Research Group, Magyar tudósok krt. 2, Budapest 1117, Hungary

ARTICLE INFO

Article history:
Received 18 March 2017
Revised 14 July 2017
Accepted 19 August 2017
Available online 31 August 2017

Keywords:
Resource pooling
Load balancing
Traffic splitting
Non-equal splitting
ECMP

ABSTRACT

Splitting traffic flows to different data paths is crucial in current and future networks. Traffic division serves as the basis for load balancing between application servers, optimal Traffic Engineering, using multiple paths in data centers, and several other 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 paper we present a flexible and effective traffic splitting method that is incrementally deployable and fully compatible with practically all existing protocols and data planes. Our proposal, called Virtual Resource Allocation (VRA), is based on setting up virtual resources alongside existing ones, thereby tricking the legacy equal traffic splitting technology into realizing the required non-equal traffic division over the physical media. We propose several VRA schemes, give theoretical bounds on their performance, and also show that the full-fledged VRA problem is NP-complete in general. Accordingly, we provide solution algorithms, including an optimal, but necessarily slow method and several quick heuristics. Our 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. Based on the results, we also give detailed suggestions on which algorithm to apply in different scenarios.

1. Introduction

Unity is strength. Treating separate network resources as one and sharing it among the 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, also often utilizing equal-length 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 service can be offered on the same infrastructure, due to 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*.

The implementation of resource pooling, however, is challenging, as the load balancers can usually split the incoming demands only roughly equally amongst the resources. 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

* Corresponding author.
E-mail addresses: krisztian.nemeth@tmit.bme.hu (K. Németh), korosi@tmit.bme.hu (A. Kőrösi), retvari@tmit.bme.hu (G. Rétvári).

http://dx.doi.org/10.1016/j.comnet.2017.08.017
1389-1286/

have different capacities, which poses a giant barrier when aspiring to an optimal Traffic Engineering [5,9–11].

There are several existing proposals which target specific cases of this issue. Weighted Cost Multipathing (WCMP, [5]), for example, aims unequal traffic splitting at data centers. It assumes SDN-capable switches, and operates by replicating rule table entries. Niagara [10] is another SDN-based proposal, which provides flexible traffic splitting between load balancers by building SDN rules based on the last bits of the source address. Fibbing [12] is lately proposed, interesting architecture, which promises centralized control over distributed routing, without SDN. It works by effectively "lying" to OSPF, advertising fake nodes and links through standard routing protocol messages. A recent application of Fibbing directly targets load balancing [13]. These proposals, however, are more or less coupled to a single field of application, and, in the case of Fibbing, would introduce a new level of abstraction, which it is yet unclear if operators are willing to cope with.

As a solution, we introduce a technique called Virtual Resource Allocation (VRA). 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. We then group the virtual resources and assign them to the physical ones, thereby tricking the legacy equal splitting technology into realizing the required non-equal load division over the existing media.

As an example, we can install two virtual machines on the stronger web-server and present them, along with the unmodified weaker server, to the load balancer. It then sees three servers, and by realizing equal split between them the stronger server 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.

Later on in this paper Traffic Engineering (TE) in IP networks will be used to introduce the VRA proposal. Let us emphasize, however, that TE is just a descriptive example application of the VRA concept, and its possible fields of usage are much broader. For instance, Fibbing can be enhanced by our algorithms proposed for virtual resource mapping. Software Defined Networking is a recent and promising trend in the IP world. While VRA does not depend on SDN, there already exist application possibilities for VRA within an SDN framework and more are likely to come. To name one, in data centers parallel shortest path are very frequent, but their capacities tend to be asymmetric [5], causing ECMP to be a suboptimal tool for splitting. WCMP, using OpenFlow, is designed to cope with this challenge. Yet, it can be enhanced by the algorithms described in this paper to minimize to forwarding table entries while keeping the oversubscription rate under a limit.

Main contributions. We present a precise *VRA problem definition* along with *theoretical error bounds* for different scenarios. We also provide *algorithms* to solve the problem *under different real-life constraints*. Our proposition 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 *highly compatible*, meaning that it does not necessitate installing any new hardware or software component in the network. Finally, VRA is *extremely efficient*, as our 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.

We *show the NP-completeness* of the full-fledged version of the VRA problem, and also show that *no polynomial time algorithm can approximate the optimal solution within* any*constant ratio*. Yet, *we propose an Integer Linear Program* (ILP) as an optimal solution along with *quick heuristics*.

Finally, we present our *simulation evaluation*, which includes our algorithms as well as the existing best-practice solution. Our results underpin that the *VRA approach has a huge practical potential*. This, together with the easy deployability make *VRA an ideal choice for network operators*.

Previous works. This paper is a continuation, and in some sense, a completion of an earlier work carried out by the same authors, which has been published in [14] and [15], and which address the Overlay Optimization an Peer-Local Optimization. The main findings of those conference papers are summarized in this one for the sake of completeness, but everything else in this paper, unless directly cited, are, to our best knowledge, first published here. These include, but not limited to the Peer-Global Optimization ILP, the results about the computational complexity of the problem, and the whole simulation evaluation. While our earlier efforts focused solely on TE, this time we have generalized it into the much more universal VRA context, which have certainly affected the structure of the whole paper.

Organization. The rest of this paper is structured as follows. Section 2 introduces the VRA concept with some simple examples. Sections 3 and 4 carry our main theoretical results about different versions of the original problem. A numerical evaluation is described in Section 5, followed by an overview of the most important related works in Section 6. Finally, Section 7 concludes this paper. Appendix A contains an ILP that solves the full-fledged VRA problem, and Appendix B reveals our theorems and proofs about its computational complexity.

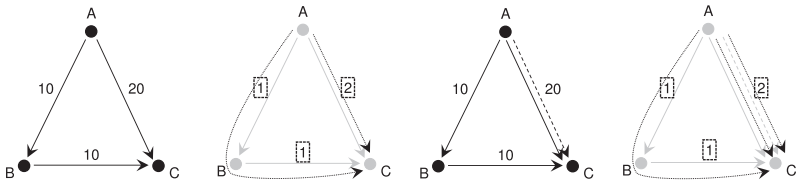
2. Virtual resource allocation overview

In this section we overview 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 that is shown, with the link capacities, in Fig. 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 (also often called link costs, link metrics or SPF (Shortest Path First) metrics), thereby we could easily create two equal cost shortest paths (i.e. paths with minimal total weight): $A-B-C$ and $A-C$, by using for example the weights shown in Fig. 1(b). On the other hand, OSPF ECMP only allows to split the traffic equally between the shortest paths, implying a 150% load on $A-B$ and $B-C$.

If, however, we could set up a virtual link on top of the existing $A-C$ link, and expose it to OSPF (see Fig. 1(c)), it would happily split the traffic in three, sending one third on the $A-B-C$ path and the rest on the $A-C$ physical link (Fig. 1(d)). Naturally, installing a virtual link over the $A-C$ physical link does not change its capacity, it only enables OSPF ECMP to use its full potential in this case. The link weights would also remain unchanged, and the new virtual link would have the same weight as the respective physical one (2 in our example). By this simple administrative intervention we can route the traffic through this network without exceeding the link capacities.

There are several possible ways to set up a virtual link parallel to an existing one. These options include Ethernet VLANs, IP-



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

Fig. 1. A triangular network. Demand: $A \rightarrow C$: 30.

IP tunnels, GRE tunnels, etc. The exact method of setting up this Layer 2 connection is out of the scope of this paper, as we only focus on the effect of the virtual links on the network performance.

2.1. Traffic Engineering basics

Traffic Engineering (TE) is the scientific area of performance management in operational networks [16]. Several methods exist for assigning traffic flows to data paths and thereby approximating optimal TE, the perhaps most well-known being MPLS with RSVP-TE (MPLS-TE). 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 in a way as to ensure that the shortest paths calculated by OSPF will map to exactly the ones chosen by the administrator, which may be a result of solving an adequate linear program or using some related heuristics [17]. OSPF-TE has been proven to be theoretically feasible [18]. In practice, however, in certain networks the quality of OSPF-TE can become arbitrarily poor compared to optimal TE [9], solely because the even-split nature of OSPF ECMP. Finding the best possible link weight configuration is not straightforward, either. It is well known to be NP-hard [9], but as a recent study revealed, even approximating it by a computationally-efficient algorithm within *any* constant ratio is infeasible [19]. Still, there are proposed weight optimization heuristics that perform well in real-life scenarios [20], [21].

2.2. Resource bounds

We shall see that generally a network using VRA performs better as the number of applicable virtual link grows. In practice, however, the number of next hops one can provision for a particular destination entry in the routing table is often limited by the OSPF implementation, in line with the OSPF RFC [7]. For example, in many Cisco, Ericsson and Juniper routers this limit is adjustable, but the maximum allowed setting 16 [22–24]. Similar limits exist for other routing protocol implementations, like EIGRP, IS-IS, RIP [22].

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

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

1. No limit. In this simple case we pose no upper bound on the maximum usable resources. Certainly, this approach is not directly applicable in real life settings, but in some scenarios solving this simpler problem will lead to the solution of 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 Section 3.1 we will require that 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 is the number of used physical links, the smaller is the number of allowed virtual links.
3. Bounded virtual resources. In this scenario the number of virtual resources are limited by a constant R, which is independent of the number of applied physical resources.

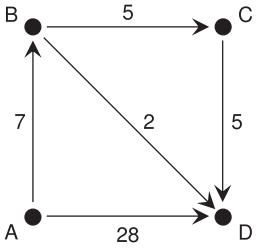


Fig. 2. A capacitated example network. Demand: $A \rightarrow D$: 35.

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 achieve an optimal TE scenario. We examine different virtual resource allocation strategies, each aiming to answer these questions.

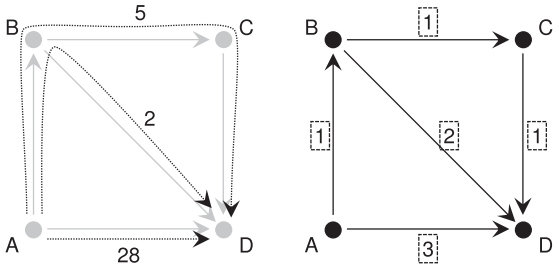
Unless otherwise stated, throughout this paper the applied metric of TE optimality is the widely adopted *Maximum Link Utilization (MLU)*. The link utilization is defined as the link traffic volume divided by the link capacity, and the MLU is the maximum of this measure over all the links of the network.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

The different VRA approaches are introduced via the example shown in Fig. 2. Clearly, the optimal solution would fully utilize 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 subsection); 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 possible TE strategies, which are discussed in depth in the upcoming sections.

Table 1
Performance of the different optimization strategies.

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



(a) Paths for Overlay Optimization (b) Link Weights

Fig. 3. Paths and weights for the rectangular example.

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. We will use this case as a one of the reference points in our 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 1 for the MLUs of this example.)

2.3.2. Overlay optimization

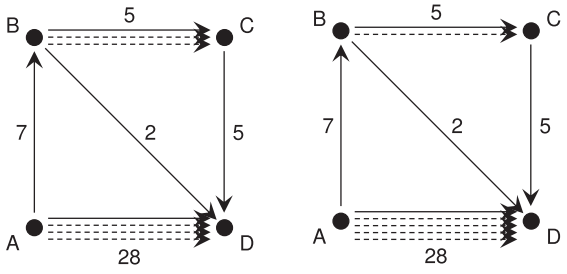
In this important scenario we suppose that a set of source-destination tunnels are already set up; yet, splitting the traffic between these tunnels have to be done somehow. Having MPLS-TE with a Path Computation Element (PCE) or some other kind of advanced means at our disposal, this problem can be tackled relatively easily. As another example, Cisco Express Forwarding (CEF) allows traffic splitting roughly proportionally to the MPLS tunnel bandwidth [25].[1] 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. 3(a). If proportional load sharing was not implemented and these path were exposed to OSPF, it would split the traffic in 3 equal portions, causing a $(35/3)/2 \approx 5.833$ utilization factor 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 $A-D$ path, 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 the $A-B-D$ path, and use the 4 virtual links to $A-D$, splitting the traffic in 5: 1. Now the maximum link utilization is on the $B-C$ link: $(35/6)/5 \approx 1.167$.

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



(a) Peer-Local Optimization (b) Peer-Global Optimization

Fig. 4. Peer Optimization.

2.3.3. Peer Optimization

In this scenario there is no overlay network, traffic optimization takes place directly on the physical infrastructure, using, for example, OSPF-TE. During the optimization the link weights are adjusted, and virtual links are set up. Two different Peer Optimization strategies are described below.

Peer-Local Optimization. In this approach 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. 3(a) would be a shortest path. Fig. 3(b) 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*.

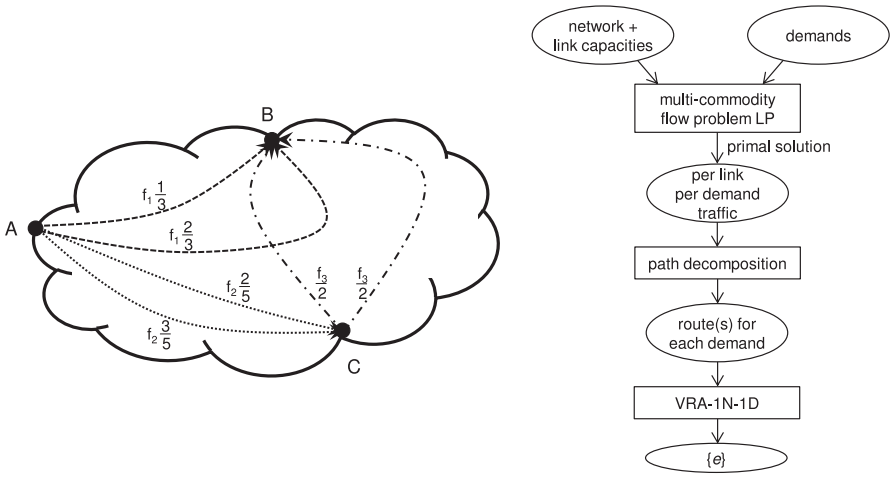
In our case at node A the 7: 28 split can be achieved exactly by adding 3 virtual links to $A-D$ (see Fig. 4(a)). At node B, however, the 5: 2 ratio cannot be perfectly realized by using only 4 virtual links (5 would be enough, though). The best we can do is allocating 2 virtual links to $B-C$. Thus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

Peer-Global Optimization. There is a fundamental problem with the previous approach: as the errors of the local optimization propagate downstream, they can cause further disturbances in the network, whereby local errors can enlarge or weaken each other's effect. This phenomenon can only be taken into account if we minimize the local errors concurrently, in a centralized manner. This is what we 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 we do not detail how we do it, only describe its potential. In our example network the optimal global solution uses the same link weights as shown in Fig. 3(b) and the virtual link provisioning is plot in Fig. 4(b). 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 lower the error downstream, at node B. This trick would not be possible using local considerations only.

[1] Note that even this implementation uses fractions of small integers to approximate the desired split ratio, just as we propose in this paper. In CEF the Bounded total resources model is used with $Q = 16$.



(a) Overlay Optimization example

(b) Overlay Optimization with tunnel/route setup

Fig. 5. Overlay Optimization.

2.3.4. Summary of the optimization strategies

The maximum link utilization of the different optimization strategies for our simple rectangular example network are shown in Table 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 the Peer-Global Optimization, without using any kind of advanced Traffic Engineering technology, approaches the theoretical optimum by only 4%. More realistic numerical studies are presented in Section 5.

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 the SDN rule table size optimization (WCMP), or the minimization of Fibbing virtual link and node numbers, as will be discussed in Section 6.

In the following sections we explain in detail and scrutinize these VRA optimization approaches. For the sake of simplicity in delivery, we will continue to use the TE as the scenario of VRA, keeping in mind that most of our algorithms has a much more general usage potential.

3. Overlay Optimization

For the Overlay Optimization [14] we make the assumption that end-to-end tunnels are already set up, using MPLS-TE for example, and OSPF ECMP is deployed over this overlay. This also means that we allow several paths between a source–destination pair, but *only permit traffic split at the ingress nodes*.

A sample scenario is plot in Fig. 5(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 the proposed technique, however, we can set up 4 virtual links (one between $A-B$ and three between $A-C$) to obtain exactly the required splitting. Note, the phrase “virtual path” could be more appropriate in this case, but for simplicity we continue to call them virtual links.

The beauty of the 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 a virtual link to one of the $A-B$ overlay link does not affect the transmission of the other demands in any way.

The 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 our proposed algorithm are shown in Fig. 5(b).

The LP (Linear Program) for the multi-commodity flow problem, which allows branching [26], 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 more than one route for a demand, each with 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, like *SimPol* proposed in [27], but finding the minimal number of routes is a strongly NP-hard problem [28], which cannot even be approximated arbitrarily well [29].

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. 5(b) we can compare the Overlay Optimization with the other techniques, as described in the Evaluation section.

3.1. VRA for One Node, One Demand

This subsection is devoted to solving the VRA-1N-1D problem, which is a crucial part of the Overlay Optimization. We start with a precise problem definition.

Table 2
VRA-1N-1D notations.

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

3.1.1. Problem definition

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

Our objective is to share the traffic over the outgoing links using OSPF ECMP, such that the actual $h_1, h_2, \ldots, h_k$ subflow values that emerge are as close as possible to the nominal $g_1, g_2, \ldots, g_k$ subflow volumes. Here, "closeness" between the i-the 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.[2]

Note that this time we 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 of the Overlay Optimization, which only focuses on realizing the traffic split rate given by $\{g_i\}$. This approach also makes the VRA-1N-1D reusable in other resource pooling schemes, like WCMP (SDN) and Fibbing, as described in the Related Works section. Certainly the MLU metric can be used for the Overlay Optimization as a whole, as shown in Section 5.

To reach our objective, we apply virtual links parallel to the physical ones. Let e_i denote the total number of (virtual and physical) links in place of an existing link, and $E = \sum_i e_i$ for the total number of allocated links. To save space in this paper, we 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:

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

and

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

As described in Section 2.2, the total number of outgoing links for a demand is limited in the practical router implementations. Consequently, we use the *Bounded total resources* model here, requiring $E \le Q$.

3.1.2. Bounds on the error

Let us examine the theoretical bounds on the error, starting with a lower limit:

Lemma 1. $U \ge 1$.

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: $G_0 \Sigma_i e_i < E \Sigma_i g_i$, i.e.: $G_0 E < E G_0$. □

[2] Note that we 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.

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

Lemma 2. *If $Q \ge G_0$ then $\exists\{e_i\}$ s.t. $U = 1$.*

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

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

Lemma 3. $U \le G_0$.

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

A stronger upper bound is:

Lemma 4. $U \le \frac{G_0}{\min_i g_i}$.

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

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_i's are given.) In this case the bound is $U \le 1/\min_i \gamma_i$.

The final remark on the error limits is about large G_0's:

Lemma 5. *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 $k = 2$, $g_1 = 1$, $g_2 = x$, s.t. $x > Q (x \in \mathbb{Z})$. 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. □

3.1.3. Optimal solution

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

Lemma 6. *The number of possible VRA-1N-1D allocations are $\binom{Q}{k}$.*

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: $\binom{(Q-k)+(k+1)-1}{(k+1)-1} = \binom{Q}{k}$. □

This means, that for small Q values (like $Q < 30, \ldots, 35$) a brute force search may be feasible, but not for much larger ones. For the given example of OSPF-TE $Q \le 16$ is probably enough in most practical cases, but VRA-1N-1D can be used in other scenarios (like SDN), where Q (representing the maximal rule number) could be in the order of thousands as well. Therefore we show an iterative algorithm that can solve the link allocation problem in pseudo-polynomial time.

Refer to Algorithm 1 that, for a given α, k, $\{g_i\}$ and E, checks whether or not it is possible to assign the links with $U \le \alpha$. If the assignment is feasible, then it also provides a solution and indicates if it is the only solution. We prove that Algorithm 1 provides correct result:

Theorem 7. *VRA-1N-1D-Fixed-E can be solved with $U \le \alpha$ if and only if $\sum_{i=1}^{k} x_i \ge E$, where $x_i = \left\lfloor \frac{\alpha g_i E}{G_0} \right\rfloor$.*

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

Input: α, k, $\{g_i\}$, E
Output: *feasible*, *single_solution*, $\{e_i\}$
  for $i \leftarrow 1 \ldots k$ **do**
    $x_i \leftarrow \left\lfloor \frac{\alpha g_i E}{G_0} \right\rfloor$ (1)
  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 $\{e_i\}$:
    $$\sum_{i=1}^{k} e_i = E; \quad 1 \le e_i \le x_i \quad (e_i \in \mathbb{Z},\ i = 1, \ldots, k) \tag{2}$$
  end if

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

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

thus

$$\frac{\alpha g_i E}{G_0} \ge e_i$$

and since $e_i \in \mathbb{Z}$,

$$\frac{\alpha g_i E}{G_0} \ge \left\lfloor \frac{\alpha g_i E}{G_0} \right\rfloor = x_i \ge e_i. \tag{3}$$

So if $\Sigma x_i \ge E$, then we can find e_i values such that (2) is satisfied, and then due to (3) we will have a valid assignment, where $U \le \alpha$.

On the other hand, if $\Sigma x_i < E$, then we cannot find e_i's such that (2) is satisfied, and $U \le \alpha$. To see this, suppose the opposite. Then (3) still must be true, and then the supposed $\sum_{i=1}^{k} x_i < E$ contradicts $\sum e_i = E$ in (2). □

As calculating x_i's according to (1) is simple (i.e., $O(1)$) and solving (2) can be done in $O(k)$, Algorithm 1 has a complexity of $O(k)$.

Next, we use this VRA-1N-1D-Fixed-E algorithm in a binary search framework for finding a minimal α that is satisfiable, given g_i's and E. To do so, we need a lower and an upper bound on U. For this, we shall use 1 and G_0, respectively, as given by Lemmas 1 and 3. (Lemma 4 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:

Lemma 8. *Consider two different link allocations, $\{e_i\}$ and $\{e'_i\}$, both using a total number of E links. For the associated errors, U and U', if $U \ne U'$ then $|U - U'| \ge 1/(G_0 E)$.*

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

$$U - U' = \max_{i\in 1\ldots k} U_i - \max_{j\in 1\ldots k} U'_j \ge \min_{i,j\in 1\ldots k} U_i - U'_j = \min_{i,j\in 1\ldots k} \frac{G_0 e_i}{E g_i} - \frac{G_0 e'_j}{E g_j} =$$
$$= \min_{i,j\in 1\ldots k} \frac{G_0}{E} \frac{e_i g_j - e'_j g_i}{g_i g_j} \ge \min_{i,j\in 1\ldots k} \frac{G_0}{E} \frac{1}{g_i g_j} = \frac{G_0}{E} \frac{1}{G_0^2} = \frac{1}{G_0 E},$$

since $e_i g_j - e_j g_i$ is a positive integer. □

Algorithm 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* $\ge 1/(G_0 E)$ **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 $\{e_i\}$

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

What remains to be done is to actually find the value of E that yields the smallest error subject to the given Q. This is done by the simple Algorithm 3. Note that this is theoretically not a polynomial time algorithm as the complexity is $O(Qk \log(G_0^2 Q))$, which is not polynomial in Q, as its size is $\log(Q)$. For a fixed Q, however, as it is the case in practice, the algorithm is indeed polynomial. Moreover, this algorithm easily tractable for all practical use cases, even if Q is in the range of thousands or even millions.

Algorithm 3 VRA-1N-1D.

Input: k, $\{g_i\}$, Q
Output: $\{e_i\}$, $U(E)$, U
  best_U $\leftarrow G_0 + 1.0$
  for $E \leftarrow k \ldots Q$ **do**
    $\{current\_e_i\}$, *current_U* $\leftarrow$ VRA-1N-1D-Bin-Search($\{g_i\}$, E)
    if *current_U* $<$ *best_U* **then**
      best_U $\leftarrow$ *current_U*
      $\{best\_e_i\} \leftarrow \{e_i\}$
    end if
    $U(E) \leftarrow$ *best_U* {used in Section 3.1.4}
  end for
  $U \leftarrow$ *best_U*
  $\{e_i\} \leftarrow \{best\_e_i\}$

3.1.4. Other problem formulations

The algorithms described above can be used, with minor modifications, to solve a set of related problems. We list three such problems here along with the proposed solutions.

Minimizing the link number. We have just answered the question "How to allocate the virtual links, if we want to minimize the error with a given link limit (Q)?" Another, similar question might be: "How to allocate the virtual links, if we want to minimize the total number of links (E) with a given error limit (U_{lim})?"

The solution is simple. Consider $U(E)$ generated by Algorithm 3 (VRA-1N-1D). It is a weakly decreasing function, whose domain is a subset of the positive integers. The answer to the question is the E, where $U(E-1) > U_{\text{lim}}$ and $U(E) \le U_{\text{lim}}$.

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 $E = G_0$ is enough in the worst case as well, thus the complexity is $O(G_0 k \log(G_0^3))$.

Simultaneously minimizing the error. The next, somewhat harder question is about solving several VRA-1N-1D problems concurrently: "Given N independent nodes, each with a set of desired traffic volume per outgoing links ($\{g_{ni}\}$), how to allocate virtual links at each node so that the total number of physical and virtual links of all the nodes together is limited (Q) and the goal is to minimize the error over all the links of all the N nodes?"

Although in our OSPF-TE example this problem is not directly addressed, in other use cases, just like the WCMP, where the rule number is constrained, this is indeed a valid an important question, certainly with "rules" instead of "links". A simple greedy algorithm (similar to the one mentioned in Section 5.1 of [10]) is optimal in this case, as described in Algorithm 4. We use Q_1, Q_2, ..., Q_N links for each problem, s.t. $\sum Q_i = Q$, minimizing $\max U_i(Q_i)$.

Algorithm 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

Theorem 9. *Algorithm 4 finds an optimal solution.*

Proof. We prove by contradiction. Suppose Algorithm 4 results in $\{Q_i\}$ and U, and yet there is $\{R_i\}$ with $U' < U$, s.t. $\sum R_i \le \sum Q_i = Q$. Because of the last condition, and because $\exists i$: $R_i \neq Q_i$ there is at least one j, s.t. $R_j < Q_j$. Due to the nature of the algorithm and the nonincreasing property of U_j: $U_j(Q_j - 1) \ge U \ge U_j(Q_j)$. On the other hand $U' \ge U_j(R_j) \ge U_j(Q_j - 1) \ge U$, which contradicts $U' < U$. □

The complexity of Algorithm 4 is $O(NQk \log(G_0^2 Q) + Q)$, where $k = \max k_i$, $G_0 = \max G_{0i}$. Note that the algorithm can 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 $O((Q+N)k \log(G_0^2 Q))$.

Simultaneously minimizing the link number. The last question is another formulation of solving parallel VRA-1N-1D problems: "Given N nodes with demands, how to allocate the virtual links at all of them, if we want to minimize the total sum of the links of all the nodes given a limit on the highest error within them?" Nevertheless, this question is trivial: the number of links should be minimized independently for each problem, as described in the beginning of this subsection.

4. Peer Optimization

The Peer Optimization approaches have been briefly introduced in Section 2.3 and are described in full detail here. In this scenario we are given a capacitated network and a set of demands (see Fig. 6). 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 maximum link utilization. We will refer to this kind of problems as *Virtual Resource Allocation, Peer Optimization (VRA-PO)* problems.

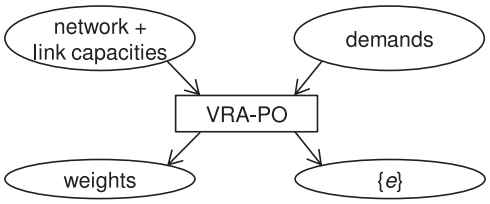
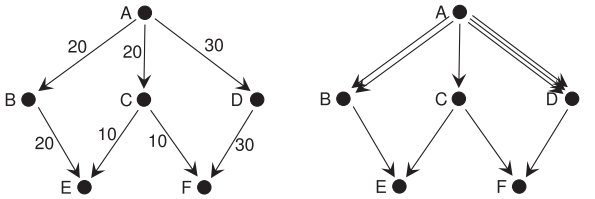
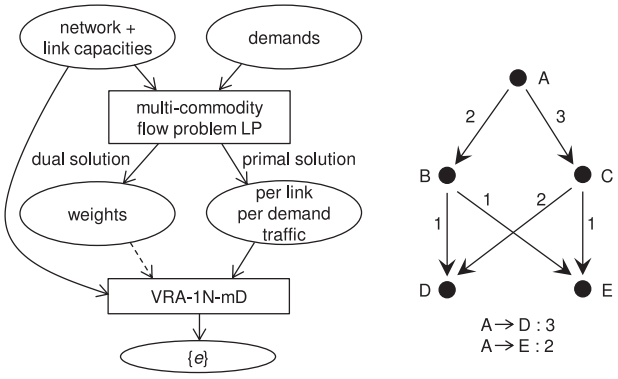


Fig. 6. Peer-Global Optimization with ILP.



(a) A capacitated network. Demands: $A \to E : 30$, $A \to F : 40$ (b) Installed virtual links

Fig. 7. Multi-demand constraint example.



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

Fig. 8. Peer-Local Optimization.

Just as before, here we have a limit on the number of usable links per node as well, using the *Bounded total resources* model. In this scenario, however, the limit exists *per node per demand*, in line with the requirement, that a single traffic flow cannot be split onto too many outgoing links. As an example, consider the capacitated network and the two demands shown in Fig. 7(a). Clearly, for the 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 solve the problems by setting up virtual links as shown in Fig. 7(b). Although this way six links are leaving node A, none of the demands are split onto more than four, obeying the limit.

4.1. Peer-Local Optimization

As introduced in the VRA Overview section, the Peer-Local Optimization works at the node level, consequently its complexity is substantially lower compared to the Peer-Global Optimization. The overview of its operation is shown Fig. 8(a).

The first step is the same as for the Overlay Optimization: solve the multi-commodity LP with splittable flows, which can be done quickly. The primal solution provides the per link per demand traffic volumes, just as before, but in this case we also extract the dual solution. These contain the link weights (minus a constant r) [18],

Table 3
VRA-1N-mD notations.

$G = (g_{ij}) \in \mathbb{Z}^{D\times k}, g_{ij} \geq 0$	Desired traffic volume per outgoing links per demands
$\Gamma = (\gamma_{ij}) \in \mathbb{R}^{D\times k}$	Row-normalized version of G
$\Sigma = (\sigma_{ij}) \in \{0,1\}^{D\times k}$	$\sigma_{ij} = 0$ if $g_{ij} = 0$, $\sigma_{ij} = 1$ otherwise
$G_i = \sum_{n=1}^{k} g_{in}$	Total traffic volume of demand i
h_{ij}	Actual traffic volume per demand per outgoing link
$e_1, e_2, \ldots, e_k$	Number of allocated links (physical and virtual together)
$E_i = \sum_{n=1}^{k} e_n \sigma_{in}$	Total number of parallel links on the shortest paths for demand i
$U_{ij} = e_j/(\gamma_{ij} E_i) \quad g_{ij} > 0$	Per demand per link error
$U = \max_{\gamma_{ij}>0} U_{ij}$	Per node error
$Q \geq E_i \quad (\forall i), \quad Q \in \mathbb{Z}^+$	Upper bound on the number of usable links per demand

necessary to OSPF-TE. The next and final step is to solve the VRA-1N-mD problem for each node independently. VRA-1N-mD stands for the "One Node and Multiple Demand version of the Virtual Resource Allocation", and provides locally optimal virtual link settings, as detailed in Section 4.2 below.

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 has zero traffic on it from that demand. It has been proven in [30] 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 is not able to 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 we followed in our simulation evaluation, to divert a minimal amount of traffic to these links. This is denoted by the dashed arrow in Fig. 8(a).

4.2. VRA for one node, multiple demand

In the VRA-1N-mD version of the problem we deal with a single node and several demands routed through it. Consider the example shown in Fig. 8(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. It is easy to see that 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 this subsection, we examine, how.

The general problem is formulated below.

4.2.1. Problem definition

The formal definition of the problem, using the notations summarized in Table 3, is as follows. For a network node A we are given a

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

matrix, representing how the demands $1, \ldots, D$ arriving at node A 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 and columns. Later on we will also use the row-normalized and the signum versions of G:

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

As an example we show these matrices for the simple instance shown in Fig 8(b):

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

We aim to set up e_j number of parallel links (including the physical and virtual ones) for every 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 setting the number of links to zero on an outgoing link (having $e_j = 0$ for some j), meaning that we are not setting up parallel links, but instead disabling a physical link, in a similar fashion to Overlay Optimization with Path Exclusion. Whether this can be performed or not is a network administration issue, and is outside the scope of this paper. Nevertheless, due to space constraints, in the rest of this paper we suppose that such link disabling is not realizable.

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

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

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

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 we will study the problem variant with limited number of links used for traffic splitting. We are using the *Bounded total resources* model (Section 2.2), but here the upper limit is applied on a per demand basis (by requiring $E_i \le Q$ for all $i = 1, \dots, D$), as described at Fig. 7.

4.2.2. Bounds on the error

We start again with the examination of the theoretical bounds on the error. The first statement is analogous to Lemma 1, giving a lower limit:

Lemma 10. $U \ge 1$.

Proof. We 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 i-th 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 $\sum_{j=1,\dots,k:\sigma_{ij}=1} \frac{e_j}{E_i} < \sum_{j=1,\dots,k:\sigma_{ij}=1} \gamma_{ij} \Rightarrow \frac{E_i}{E_i} < 1$, which is clearly a contradiction. □

Note that for some G matrices $U = 1$ is attainable by properly selecting e_j's, as for instance:

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

$$e = \begin{bmatrix} 15 & 30 & 60 & 10 & 2 \end{bmatrix}.$$

For some G's, however, there is no such e_j setting, like in the previous example:

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

Next, we give an upper bound on the error:

Lemma 11.

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

Proof.

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

from which the theorem follows. □

Lemma 12. *There is no universal (G-independent) upper bound on the error.*

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

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

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

4.2.3. Unlimited total number of links

After exploring the theoretical limits, we try to find the optimal link allocation. We start with a simplified version of the problem, where the maximum allowed link constraint is released, i.e., we allow unlimited number of parallel links to be used simultaneously. Hence, for simplicity we will use the normalized version of the link number e_j, denoted by f_j to avoid confusion: $f_j \in \mathbb{R}^+$, $\sum_{j=1}^{k} f_j = 1$.

Notes on the types of the solution. We now show two, somewhat surprising observations about the types of the optimal solution. Although the complete proofs are omitted to save space, we highlight their most important steps below.

Theorem 13. *Even if G contains integers only, the optimal solution may contain irrational numbers.*

Sketch of Proof. Consider the following input matrix:

$$G = \begin{bmatrix} 2 & 1 & 0 \\ 2 & 2 & 1 \end{bmatrix}; \quad \Gamma = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} & 0 \\ \frac{2}{5} & \frac{2}{5} & \frac{1}{5} \end{bmatrix}.$$

It can be proven that for this matrix $U_{12} = U_{21} = U_{23}$ and $U_{11} < U_{21}$, $U_{22} < U_{12}$. For the optimal f_1, f_2, f_3, 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:

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

Clearly, the optimal f_1, f_2, f_3 are irrational in this case. □

This theorem shows that *there are cases of the VRA-1N-mD, where using finite number of parallel links the per node error will never be minimal.*

Theorem 14. *There are VRA-1N-mD problems whose optimal solution contains at least one f_j that cannot be written in a finite form using integer constants and the usual + , - , · , / and the n-th root operators ($n \in \mathbb{Z}^+$) only (i.e., cannot be solved by radicals).*

Sketch of Proof. Consider the following matrix:

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

It can be proven 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 $f_1, \dots, f_6$. From these equations $f_2, \dots, f_6$ can be eliminated, and what remains is a polynomial of f_1:

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

According to the Galois theory, a polynomial equation can be solved by radicals if and only if its Galois group is a solvable group. For the polynomial given in (4), the mathematical software Maple [31] have determined that its Galois group is the symmetric group S_5. This group, consisting of 120 elements, is not solvable, meaning that (4) cannot be solved by radicals. □

This means that in general *finding a closed-form solution (using unlimited number of links) for VRA-1N-mD is not possible*, and instead some kind of approximation algorithm should be applied.

An LP-based solution. We now consider the problem as a linear optimization task. Due to the previous results we cannot write a Linear Program that solves it in one step. Nevertheless, we can easily set up an LP that computes $\{f_j\}$ while keeping the per node error under given α constant. (Certainly, for too small α-s, the LP will not have a solution.)

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

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

which can be rearranged as:

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

which leads to the LP 1.

LP 1 VRA-1N-mD, Unlimited Links, Given α

$$\begin{aligned}
&\textit{indices:} && i = 1 \dots D \\
&&& j = 1 \dots k \\
&\textit{constants:} && \alpha \quad (\alpha \ge 1,\ \alpha \in \mathbb{R}) \\
&&& \gamma_{ij} \quad (\gamma_{ij} \in \mathbb{Q},\ \gamma_{ij} \ge 0,\ \forall i : \sum_{n=1}^{k} \gamma_{in} = 1) \\
&&& \sigma_{ij} \quad = \operatorname{sgn}(\gamma_{ij}) \\
&\textit{variables:} && f_j \quad (f_j \ge 1,\ f_j \in \mathbb{R}) && (5) \\
&\textit{objective:} && \text{minimize} \sum_{j=1}^{k} f_j && (6) \\
&\textit{constraints:} && 0 \le \sum_{n=1}^{k} \sigma_{in} f_n - \frac{f_j}{\gamma_{ij}\alpha}, \quad \forall i,j : \gamma_{ij} > 0 && (7)
\end{aligned}$$

Note that for any solution $\{f_j\}$ that satisfies (7), $\{cf_j\}$ does so as well, where $c > 1$. Therefore the objective function (6) has been added, mainly for aesthetic reasons, as for the unlimited case any $\{f_j\}$'s satisfying (7) are equally good. Note also that (5) contains $f_j \ge 1$, which is responsible for enforcing $f_j > 0$. This may result in large f_j's, but after obtaining them, they can be normalized to one by dividing by their sum.

Now we can use a binary search for the smallest α, for which LP 1 is solvable, as described in Algorithm 5, and this way approximate the optimal solution arbitrarily. Note that ϵ_U shows how close we want to get to the optimal error. Providing it is necessary due to the consequences of Theorem 13. A typical value would be $\epsilon_U = 10^{-8}$.

Algorithm 5 VRA-1N-mD, Unlimited Links.

Input: G, ϵ_U
Output: $\{f_j\}$
  $lower \leftarrow 1.0$ {See Lemma 10}
  $upper \leftarrow 1/(\min_{i,j} \gamma_{ij})$ {See Lemma 11}
  while $upper - lower \ge \epsilon_U$ **do**
    $\alpha \leftarrow (upper + lower)/2$
    if SOLVE_LP 1(α, G) finds a solution **then**
      $upper \leftarrow \alpha$
    else
      $lower \leftarrow \alpha$
    end if
  end while
  $f \leftarrow$ SOLVE_LP 1($upper$, G) {Lower limits are not valid settings, upper limits are valid. We need a valid setting}

About the complexity of Algorithm 5: it 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 5 does a binary search on the $[1,\ 1/\min_{i,j}\gamma_{ij}]$ interval, until it reaches the 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 5 runs in polynomial time.

To sum up, *we have an iterative solution for the unlimited link version of the VRA-1N-mD problem, which quickly converges to the optimal setting*. According to Theorem 14, a significantly better solution cannot be given.

4.2.4. Limited total number of links

Now we 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 $E_i \le Q (i = 1, \dots, D)$.

ILP-based solution. Linear Program 1 can be easily modified to LP 2 to suit the limited total number of links case. Note that the objective function might be omitted here as well, it only forces the ILP solver to select a solution with the fewest total number of links.

LP 2 VRA-1N-mD, Limited Links, Given α

$$\begin{aligned}
&\textit{indices:} && i = 1 \dots D \\
&&& j = 1 \dots k \\
&\textit{constants:} && \alpha \quad (\alpha \ge 1,\ \alpha \in \mathbb{R}) \\
&&& \gamma_{ij} \quad (\gamma_{ij} \in \mathbb{Q},\ \gamma_{ij} \ge 0,\ \forall i : \sum_{n=1}^{k} \gamma_{in} = 1) \\
&&& \sigma_{ij} \quad = \operatorname{sgn}(\gamma_{ij}) \\
&&& Q \quad (Q \in \mathbb{Z}^+) && (8) \\
&\textit{variables:} && e_j \quad (e_j \in \mathbb{Z}^+) && (9) \\
&\textit{objective:} && \text{minimize} \sum_{j=1}^{k} e_j \\
&\textit{constraints:} && 0 \le \sum_{n=1}^{k} \sigma_{in} f_n - \frac{f_j}{\gamma_{ij}\alpha}, \quad \forall i,j : \gamma_{ij} > 0 \\
&&& \sum_{j=1}^{k} \sigma_{ij} e_j \le Q \quad \forall i && (10)
\end{aligned}$$

Also note that LP 2 is actually an Integer Linear Program (ILP), and thus although it is a simple and elegant way to find the optimal link setting, the polynomial running time is not guaranteed anymore. Just as before, LP 2 works for a fixed α, but using it in Algorithm 6 solves the VRA-1N-mD problem.

Algorithm 6 VRA-1N-mD by ILP.

Input: G, Q, ϵ_U
Output: $\{e_j\}$
  This is the same as Algorithm 5, but with solving LP 2 instead of LP 1.

A heuristic solution. We provide a suboptimal, but faster heuristic alternative to the ILP-based solution in Algorithm 7. The idea is very simple: reduce VRA-1N-mD to VRA-1N-1D. For the reduction we essentially solve the VRA-1N-mD problem with unlimited number of links.

Algorithm 7 VRA-1N-mD Heuristic.

Input: G, Q, ϵ_U
Output: $\{e_j\}$
  $\{f_j\} \leftarrow$ ALGORITHM 5(G, ϵ_U)
  $\{e_j\} \leftarrow$ ALGORITHM 3*($\{f_j\}$, Q)

A note on Algorithm 3* (referred at Algorithm 7): Algorithms 1–3 require $\{g_i\}$ as input, but internally they only use it in their normalized form: g_i/G_0 ($G_0 = \sum_{j=1}^{k} g_j$). A modified versions of these algorithm, marked by an asterisk, take g_i/G_0 as input. This also affects the complexity: instead of $O(Qk\log(G_0^2 E))$ here, by applying Lemma 11, we have $O(Qk\log(1/(\epsilon_U \min_i g_i/G_0)))$. In Algorithm 7 we provide $\{f_j\}$ as this input.

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

Considering the complexity of Algorithm 7: 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 we have not established a lower bound on f_j, on practical problem instances we found Algorithm 7 to be much faster than Algorithm 6 [15].

4.3. Peer-Global Optimization

In Section 2 the Peer-Global Optimization has been briefly introduced as a solution to the VRA Peer Optimization (VRA-PO) problem. Unlike the Peer-Local Optimization, this approach has the potential to find the *optimal* solution to the problem. To do so, we propose a single Integer Linear Program, called VRA Peer-Global ILP. The ILP itself is fairly long, therefore we present it separately in Appendix A. Although it does provide the optimal solution, it uses plenty of auxiliary variables, mainly integers, making it hardly usable in most of the practical cases. Nonetheless, Section 5 contains examples where this ILP had been successfully solved hundreds of times.

Having a slow ILP and the Peer-Local algorithms, which can be viewed as heuristic solutions to the Peer-Global Optimization problem, the question naturally arises: is there a computationally efficient, optimal solution of the VRA-PO problem? We show that the answer is strongly negative, which justifies the importance of the work and the results described previously.

Our most important results about the computational hardness on the VRA-PO problem are summarized below. These findings imply that *there are no solutions that are both computationally efficient and optimal*. As the detailed explanations of these results are quite lengthy, the formal definitions, theorem statements and their proofs has been moved to Appendix B.

Our first result is that *VRA-PO is NP-complete*. Moreover, we have found that *the optimal solutions cannot even be efficiently approximated*. First we state that no polynomial time algorithm exists that approximates the optimum of VRA-PO better than a factor of 6/5. This means that the problem is not part of the PTAS (Polynomial Time Approximation Scheme, [32]) class, as it is not possible to efficiently approximate the optimal solution within *every* constant ratio. Having a single demand is a version of the problem, which is similar in this respect, as we can prove that no polynomial time algorithm exists that approximates the optimum better than a factor of 18/17.

The next and final result is the strongest one. The APX (meaning "Approximable") class is a collection of optimization problems that can be approximated to *some* constant ratio. We have shown that the VRA-PO problem is not part of the APX class, meaning that *no polynomial time algorithm can approximate the optimal solution within* any*constant ratio*.

5. Evaluation

The negative results of the previous section state the hardness of the VRA-PO in general. They, however, do not necessarily mean that in practical networks no effective solution can exist. To see how well our algorithms perform in realistic environments, we have implemented a simulation framework, which is based on our descriptive use case, the OSPF ECMP Traffic Engineering. The simulator takes a capacitated network and a set of demands as inputs, and solves the MLU minimization problem using several different algorithms.

The framework and the optimization algorithms have been implemented in C++ using the powerful LEMON Graph Library [33]. The embedded linear programs have been solved using the IBM ILOG CPLEX Optimizer.

5.1. Examined algorithms

The following seven optimization approaches have been included in the simulations:

1. Overlay Optimization, as described in Section 3.
2. Overlay Optimization with Path Exclusion, see Sections 3 and 2.3.2.
3. Peer-Local Optimization using ILP, described in Sections 4.1 and 4.2.4 (Algorithm 6).
4. Heuristic Peer-Local Optimization, as shown in the same sections (Algorithm 7).
5. Peer-Global Optimization, outlined in Section 4.3.
6. Global Optimization, described in Section 5.1.1 below.
7. OSPF Weight Optimization, as introduced in Section 2.3.1, and detailed in Section 5.1.2 below.

At the Overlay and Peer Optimizations the bounds on number of the virtual resources have been implemented slightly differently than introduced in the previous sections. There the *Bounded total resources* model has been used (see Section 2.2), by requiring $E \leq Q$ and $E_i \leq Q$, respectively. In the simulation the *Bounded virtual resources* constraint has been implemented as $E \leq |S_n| + R$, 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 of this choice is that it makes the comparison of algorithms running at nodes with different outgoing physical links easier. Also, in this case for the Peer 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. 5(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 whose traffic is 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 value of the demand on the given link (G_{ij}) has been risen at the expense of the other links.

5.1.1. 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*. 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, we have included a simple algorithm in our simulation platform that treats the output of the multi-commodity LP as the 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, we will actually divide the MLU's of the 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, the Global Optimum will be constant 1.0 due to the normalization.

5.1.2. OSPF Weight Optimization

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,

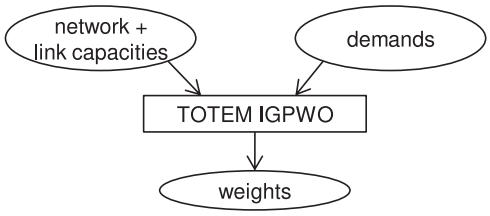


Fig. 9. Global Optimization with TOTEM.

called TOTEM (TOolbox for Traffic Engineering Methods [34], [35]). 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 [36]). Nevertheless, in the rest of this paper we will simply refer to this algorithm as the "TOTEM" method (see Fig. 9).

We have taken the source code of IGPWO module out of the TOTEM framework (version 3.2.1) and included in our 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 we simply used the maximum link utilization as the error to be minimized. During the simulations we 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.

5.2. Simulation scenarios

Two realistic network topologies had been simulated. The first one is the well known North American Abilene network topology, shown in Fig. 10(a). The second examined network is shown in Fig. 10(b): it is a simplified Pan-European optical core network, which have been proposed in [37]. In both networks uniform link bandwidths of 100 units have been used.

We had 5 simultaneous demands in each run. In each case the source and destination nodes were selected randomly, requesting random demand bandwidths with uniform distribution on the [5, 30] units interval. The maximal number of virtual links or paths (R) was varied for each setting between 0 and 8, inclusive.

Due to the complexity of the ILP applied in the Peer-Global Optimization, it took up to several hours, or even days to run a single instance of simulation (consisting of 9 runs with $R = 0, \ldots, 8$) with only 5 demands. Yet, these scenarios have been simulated 300 times (with all the 7 algorithms) on a high-performance computer to decrease the variance of the results.

5.3. Simulation results

The results are shown in Fig. 11. The first two charts show the MLU as a function of R for the examined scenarios.

For the Abilene network, TOTEM performed almost as good as the Peer-Global Optimization for the no virtual link case, which is its theoretical lower bound. For $R = 0$ the Peer-Local and the 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 nodes the performance of the Overlay Optimization becomes as good as TOTEM's, and as R grows, the Overlay Optimization clearly overperforms 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 both of these optimizations are quick heuristics only. The Peer-Global Optimal MLU is well below TOTEM's even for $R = 1$, and it keeps decreasing as R increases, and almost reaches the Global Optimum for only $R = 4$. It means, that the Peer Optimization approach does have a high potential, but the 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 the Peer-Local Optimizations performed somewhat worse, yet the Peer-Global Optimization shows, that the theoretical Global Optimum is very closely approachable using VRA.

Fig. 11(c) shows the *average* link utilizations for the Pan-European network, again normalized by the optimal *Maximum* Link Utilization, which is why all the results are well below one. As all the presented algorithms aims to minimize a different metric, the MLU, the average link utilization chart is considerably different than the MLU charts. Here, the 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. Nevertheless, in all the cases their performance is very close to each other.

Similarly, sometimes the Overlay Optimization with Path Exclusion seems to perform worse than the 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 the Overlay Optimization can theoretically overperform the Peer-Global Optimization, although it does not happen in our practical evaluation.

Looking at the solution of the embedded VRA-1N-1D problem (Fig. 11(d)), the 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 Algorithm 3 and the whole VRA concept.

Table 4
Resource consumption of the different algorithms.

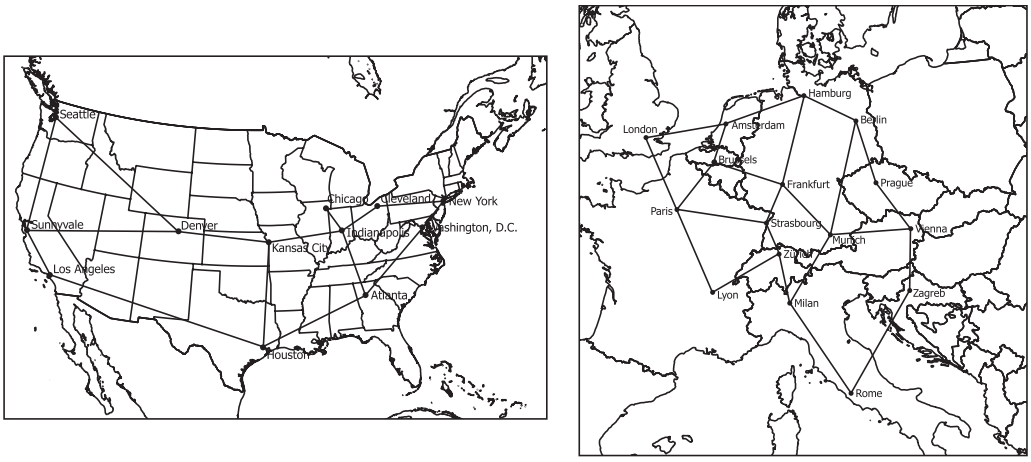
	Average running times		Average memory usages	
	Abilene	Pan-Eu	Abilene	Pan-Eu
5 Algorithm	0.35 s	0.74 s	7 MB	9 MB
TOTEM	5.98 s	18.1 s	4 MB	4 MB
7 Algorithm	9m 37 s	9 h 49 m	46 MB	1.5 GB

5.4. Resource consumption

Each optimization session was run on a single core of an Intel Xeon Processor X5660 (but several sessions were run in parallel on a high performance computer). The average running times and memory consumption of the different algorithms are summarized in Table 4.

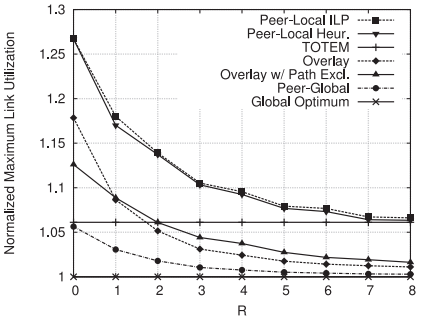
The first row ("5 algorithms") shows the total resource usage of the two Overlay Optimizations, two Peer-Local Optimizations and the Global Optimization. The second row represents the TOTEM optimization alone, while the third row reveals the results for all the seven algorithms altogether. The first two rows were acquired by re-running the algorithms 300 times and averaging the results, while the last row ("7 algorithms") are from the actual runs that have been described earlier, which means that they also represent the average of 300 runs.

The results show very short ($\approx$ second, sub-second) running times for the Overlay, Peer and Global Optimizations. No further

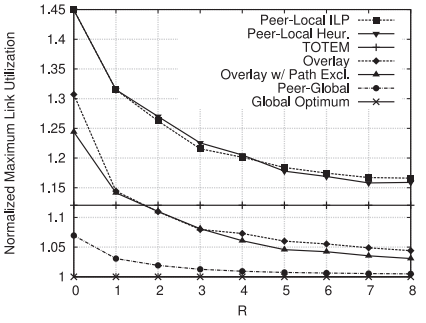


(a) Abilene network (b) Pan-European network

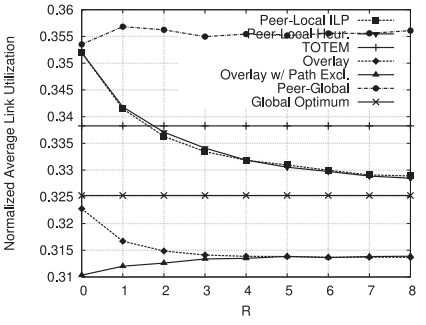
Fig. 10. Network topologies.



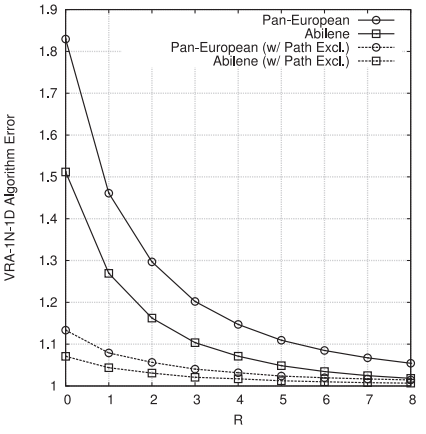
(a) Normalized MLU for the Abilene topology



(b) Normalized MLU for the Pan-European topology



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



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

Fig. 11. Simulation results.

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, up to a minute, which is considerably higher than the previous ones, but still can be practical in non-realtime cases. 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 the 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 is not recommended to be used practice, but, as emphasized earlier, it was only included as a reference value for the Peer Optimizations.

6. Related works

In this section we briefly overview the most relevant publications in chronological order.

Achieving near-optimal Traffic Engineering solutions for current OSPF/IS-IS networks [38] targets the same problem as we do in our TE example: overcoming the equal split limitation of the OSPF/IS-

Table 5
High level summary of different techniques.

	Computational complexity	Practical running time	Network performance	Requirements
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	OSPF ECMP only
Global Optimization	Polynomial	Very fast	Best (optimal)	MPLS-TE or similar techniques

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 [[39]) 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 a quick and optimal Traffic Engineering, but *at the expense of modifying the routing basics and using unlimited number of next hops*.

The already mentioned *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 (LPM), 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 our VRA-1N-mD problem.

WCMP is therefore similar to our work both in terms of the goal and the applied technology (LPM-based forwarding). The similarities continue if we look at the WCMP problem formulation. Their fundamental mathematical problem is perfectly identical to our VRA-1N-1D case: 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, our algorithms, which can directly be applied to the WCMP problems, always provide the *optimal solutions*, not just approximations. Furthermore, our proposals can achieve this with comparable or even smaller computational complexity.

Fibbing [12] 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 the other participants of the routing protocol. The applicability of Fibbing for per destination load balancing with uneven splitting ratios has recently been demonstrated [13]. If a routing has a single shortest path for a destination, and two parallel paths are to be used with equal traffic share, a fake node and a fake link has to be injected to the network using Fibbing. If, for example, 33%–67% traffic ratio is to be achieved in an unequal load sharing case, two fake nodes and links has to be applied. Easily, *our VRA-1N-1D algorithm can be used with Fibbing* to find the best approximation of an arbitrary split ratio using minimal 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 traffic splitting scheme targeting an SDN environment. Its goal is inherently the same as ours: 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 ours: try to approximate a given ratio by the sum of fractions of small integers. In VRA we mostly use fractions with common denominator, Niagara uses sums of $1/2^i$. We 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 our VRA-1N-mD 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 Section 5.1 of [10], NIAGARA utilizes a greedy algorithm to minimize the total number of rules for multiple aggregates. *This idea has been* adopted in our Algorithm 4, and *proven to be optimal in Section 3.1.4*. Niagara seems to be a promising and powerful tool. On the other hand, VRA poses much lower requirements to the network, and is therefore more easily deployable.

COYOTE [40] 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 rations are realized in the network by combining the Fibbing way of injecting fake protocol messages and some of our algorithms to optimize the number and location of the fake entities.

Several techniques have been proposed for creating a virtual network over a physical one. There are a variety of reasons to do so, one of them being simpler than the original one, as suggested in [41] recently. Our Overlay Optimization is also such a technique.

Its main advantage is the decomposition of a large problem into independent, smaller subproblems, one for each demand. This results in fast and very efficient operation. On the other hand, the path decomposition (see Fig. 5(b)) may end up with too much end-to-end shortest paths, which can result in suboptimal splitting if the number of applicable virtual paths are 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 [41]. Using shorter (non end-to-end) virtual overlays, may be a good 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.

Finally, let us note that the realization of a predefined (equal or non-equal) splitting rate is not straightforward in the data plane. A packet-based solution can be accurate, but it may cause reordering of datagrams of a single flow. Using network flows as the unit of splitting, on the other hand, may be less exact regarding the splitting ratio [42]. Consequently, an intermediate approach, the so-called flowlet-based splitting has been proposed [43], which are now actively used in the network nodes [44]. A recent proposal, called FlowBender [45], aims to enhance the flowlet-based splitting by taking advantage of congestion notifications.

7. Conclusions

In this paper we 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, we have proposed the Virtual Resource Allocation (VRA) technique, which tricks the load balancer into an almost arbitrary traffic split ratio. This simple proposal *does not require 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.

We have *examined the theoretical limits* of the formalized problem, and have *given fast and optimal algorithms*, where possible. We have shown, however, that finding the optimal solution for some important scenarios are computationally intractable. For these cases we *have proposed heuristic algorithms*, although finding better ones could result in further performance improvements.

We have implemented a simulation evaluation framework for a possible VRA application: IP Traffic Engineering. The simulation results underpin that *the VRA approach has a huge practical potential*. In the examined networks, the Peer-Global Optimization algorithm, which applies VRA, achieved much better network performance than the "traditional" (TOTEM) method. Moreover, by allowing as few as 3–4 virtual links per node, the theoretical best performance could be approximated up to 1–2%. Another VRA implementation, the 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 links. For quick comparison Table 5 contains a high-level summary of the presented techniques. The numerical details can be found in the Evaluation section.

Finally, we have discussed how VRA can be applied to a variety of present and future scenarios, including, but not limited to Fibbing, SDN environments like WCMP, classical OSPF-TE and COYOTE.

Acknowledgments

This research was partially supported by the Hungarian Scientific Research Fund (grant no. OTKA 108947). The simulation infrastructure used in this work was supported by TÁMOP (grant no. TÁMOP-4.2.2.B-10/1–2010-0009). The authors would like to thank Edit Halász, Tamás Henk and Attila Vidács for their support during this work.

Appendix A. ILP Formulation of VRA Peer Optimization

In this section we present the ILP solution of the VRA-PO, which has been introduced in Section 4.3. (For the formal problem definition, see B.1.)

LP 3 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
$|S_n|$ 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 (A.1)
σ_{dl} binary: $\sigma_{dl} = 1$ iff l is on a shortest path between O_d and D_d
α real, max. error
e_l ≥ 1, integer: no. of parallel links
θ_{dl} ≥ 0, real: traffic ratio of demand d that uses link l (A.2)
y_{dnk} $\forall d, n : \{|S_n| > 0, n \neq D_d\}, k = 0 \dots |S_n| + R,$ binary:
$y_{dnk} = 1$ iff $\sum_{l \in S_n} \sigma_{dl} e_l = k$
z_{lm} $m = 1 \dots R + 1,$ binary: $z_{lm} = 1$ iff $e_l = m$

objective: minimize $\alpha + r_3 \sum_d \sum_l \theta_{dl} + r_3 \sum_l e_l$ (A.3)

constraints: $u_{dO_d} = 0 \quad \forall d$ (A.4)

$$w_l + r - u_{dj} + u_{di} \geq (1 - \sigma_{dl}) r_2 \quad \forall l = (i, j), \forall d \tag{A.5}$$

$$w_l + r - u_{dj} + u_{di} \leq (1 - \sigma_{dl}) M \quad \forall l = (i, j), \forall d \tag{A.6}$$

$$\sum_{l \in S_n} \theta_{dl} - \sum_{l \in T_n} \theta_{dl} = \delta_{dn} \quad \forall d, n : n \neq D_d \tag{A.7}$$

$$\sum_{d=1}^{D} \theta_{dl} G_d \leq \alpha c_l \quad \forall l \tag{A.8}$$

$$\left(\sum_{x \in T_n} \theta_{dx} + \delta_{dn} \right) m \leq \theta_{dl} k + M(3 - y_{dnk} - z_{lm} - \sigma_{dl}) \tag{A.9}$$

$$\left(\sum_{x \in T_n} \theta_{dx} + \delta_{dn} \right) m \geq \theta_{dl} k - M(3 - y_{dnk} - z_{lm} - \sigma_{dl}) \tag{A.10}$$

$$\sum_{m=1}^{R+1} z_{lm} = 1 \quad \forall l \tag{A.11}$$

$$\sum_{m=1}^{R+1} z_{lm} m = e_l \quad \forall l \tag{A.12}$$

$$\sum_{k=0}^{|S_n|+R} y_{dnk} = 1 \quad \forall d, n : |S_n| > 0, n \neq D_d \tag{A.13}$$

$$\sum_{k=0}^{|S_n|+R} y_{dnk} k \leq \sum_{l : \gamma_{dl} = 1} e_l + M \left(\sum_{l : \gamma_{dl} = 1} (1 - \sigma_{dl}) + \sum_{l : \gamma_{dl} = 0} \sigma_{dl} \right) \tag{A.14}$$

$$\sum_{k=0}^{|S_n|+R} y_{dnk} k \geq \sum_{l : \gamma_{dl} = 1} e_l - M \left(\sum_{l : \gamma_{dl} = 1} (1 - \sigma_{dl}) + \sum_{l : \gamma_{dl} = 0} \sigma_{dl} \right) \tag{A.15}$$

$$\theta_{dl} \leq \sigma_{dl} \quad \forall d, l \tag{A.16}$$

$$\theta_{dl} \geq \sigma_{dl} r \quad \forall d, l \tag{A.17}$$

$$\sum_{l \in S_n} e_l \leq |S_n| + R \quad \forall n : |S_n| > 0 \tag{A.18}$$

Notes: (A.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, $0 \le \theta_{dl} \le 1$, but $\theta_{dl} \le 1$ follows from the constraints of the ILP.

(A.9)–(A.10): $\forall d,\ l,\ k = 1, \ldots, |S_n| + R,\ m = 1, \ldots, R+1,\ m \le k,\ n \ne D_d$, where n is the source of l.

(A.14)–(A.15): $\forall d,\ n$: $|S_n| > 0,\ n \ne D_d$ 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 $|S_n| > 0, n \ne D_d$), and in each one a different element of $\{0, 1\}^{|S_n|}$ is assigned to $\{\gamma_{dl}\colon\ l \in S_n\}$.

Let us see briefly explain this ILP. The first three constraints come from the dual formulation of a multi-commodity flow problem. The node potentials of the demand origins are zero (A.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 (A.5), (A.6). $w_l \le 1$ (A.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 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 are realized exclusively by the σ_{dl} variables.

Equation (A.7) is the usual Kirchhoff junction rule. (A.8) is the link capacity constraint, where α is to be minimized in the objective. Constraints (A.9)–(A.15) are together represent the ECMP equal-split rule applied to the virtual link scenario, which could be summarized as:

$$\theta_{dl} = \left(\sum_{x \in T_n} \theta_{dx} + \delta_{dn} \right) \frac{e_l}{\sum_{x \in S_n} e_x \sigma_{dx}}, \tag{A.19}$$

where n is the source of l, $\forall d, l : \sigma_{dl} = 1$. Unfortunately (A.19) is not linear, so we have introduced a set of auxiliary variables and largely multiplied the number of constraints to make it fit the ILP. Constraints (A.9) and (A.10) basically assert

$$\theta_{dl} k = \left(\sum_{x \in T_n} \theta_{dx} + \delta_{dn} \right) m, \tag{A.20}$$

where $m = e_l$ and $k = \sum_{x \in S_n} e_x \sigma_{dx}$.

$m = e_l$ is achieved by the simple set of constraints (A.11) and (A.12). Note the upper bound of the sums is $R+1$, since using R virtual links the maximum of e_l is $R+1$.

Constraints (A.13)–(A.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 (A.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 (A.15), providing together the required $k = \sum_{x \in S_n} e_x \sigma_{dx}$ for (A.20). Note that in these equations the conditions $|S_n| > 0$ and $n \ne D_d$ are theoretically not necessary, they are only included to reduce the number of variables and the related constraints.

Eqs. (A.16) and (A.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, (A.18), limits the number of virtual links per node to R.

The objective function (A.3) minimizes the MLU (α). It also keeps the $\Sigma\theta_{dl}$ low to avoid loops and minimizes Σe_l to prevent installing unnecessary virtual links.

Finally, a few words about the M, r, r_2 and r_3 constants. Theoretically, their value can be arbitrary, as long as M is "large", r is "small" and r_2, r_3 are even smaller. In practice, however, the actual computation largely depends on these values. 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 therefore they may come up with an erroneous solution. Consequently, M should be kept relatively small, but large enough to fulfill its original purpose. Similar considerations apply for the small r, r_2, r_3 constants. In the simulation, after some theoretical calculations and practical experimenting, we have used the following values: $M = 100$, $r = 10^{-2}$, $r_2 = r_3 = 10^{-4}$.

Appendix B. Computational complexity of VRA Peer Optimization

This section reveals the details of our computational complexity related results. It consists of two subsections: the first one lists the NP-completeness theorems along with their proofs, while the second one carries the inapproximability results. We will use the notations introduced in the ILP at Appendix A, extended with a few new ones listed in Table B6.

B1. NP-completeness of VRA-PO

First we formally define the Virtual Resource Allocation–Peer Optimization as a simple a decision problem. In this first formulation we take the link weights as input parameters.

Virtual Resource Allocation–Peer Optimization with Given Weights (VRA-PO-GW)

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

Question. Is there 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 $E_n \le |S_n| + R$ $(\forall n \in V)$ and $\max_l h_l/c_l \le \beta$?

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

This definition can be changed in several ways to gain different versions of the basic VRA-PO problem, including the following possibilities:

- VRA-PO: In this case setting the link weights, and this way defining the routing of the demands, is part of the problem, too. This problem has been examined in the previous parts of the paper.
- VRA-PO-GW-SD (Single Demand): In this variant we have only one origin–destination–traffic volume triplet ($D = 1$).
- VRA-PO-GW-Q: The definition of VRA-PO-GW utilizes the Bounded virtual resources constraint (R) (described in Section 2.2). In this version the Bounded total resources (Q) limit is used instead.

Table B6
VRA-PO notations.

V	The set of vertices (nodes) in the network
F	The set of edges (physical links) in the network
$E_n = \sum_{l \in S_n} e_l$	The number of outgoing links at node n
$h_l \in \mathbb{Q}, h_l \ge 0$	The total actual traffic volume on link l

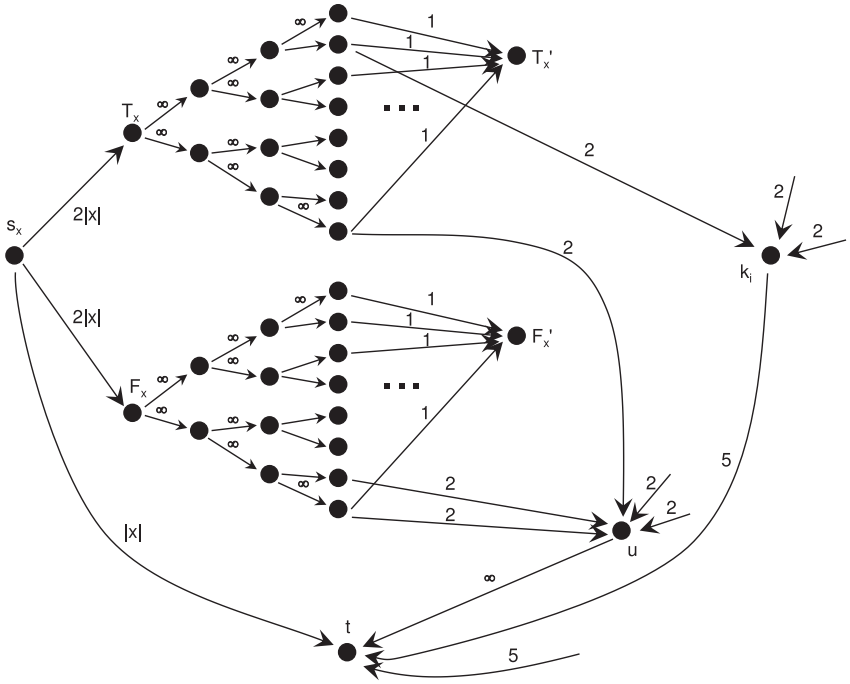


Fig. B12. Network for the 3SAT reduction.

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

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

Theorem 15. *VRA-PO-GW is NP-complete.*

Proof. This proof partially utilizes the idea of an NP-completeness proof presented in [9].

First we show that VRA-PO-GW is in NP, i.e., for an e_l set 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_l's 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 $NC^2 \subseteq P$. Then computing h_l and ultimately $\max_l h_l/c_l$ can also be done in polynomial time.

Now we prove the VRA-PO-GW in NP-hard. We will reduce 3SAT to VRA-PO-GW:

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.[3]

Question. Is F satisfiable?

For any 3SAT instance F we create a corresponding instance of VRA-PO-GW. Fig. B12 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 three global nodes, t, u and w. The number of the leafs of both trees directly downstream T_x and F_x are $|x|$ 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. 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 the $s_x t$ link, for which $w_{s_x t} = \log_2 |x| + 3$. The *demands* are as follows: for each variable x: $s_x \to t$: $4|x|$, $T_x \to T_x' : |x|$, $F_x \to F_x' : |x|$. $R = 1$. $\beta = 1$. This reduction is clearly polynomial.

We now prove that if the F 3SAT instance is satisfiable, then there is a suitable virtual link allocation for the VRA-PO-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 link: 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 $k_i t$ type links, where the capacity is 5.

We 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 arrives 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-PO-GW, then the F 3SAT instance 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-PO-GW. By symmetry, this statement also holds for the links between F_x and F_x'. It is also easy to see that $e_{s_x t} = 1$ and either ($e_{s_x T_x} = 1$ and $e_{s_x F_x} = 2$) or ($e_{s_x T_x} = 2$ and $e_{s_x F_x} = 1$): if it were

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

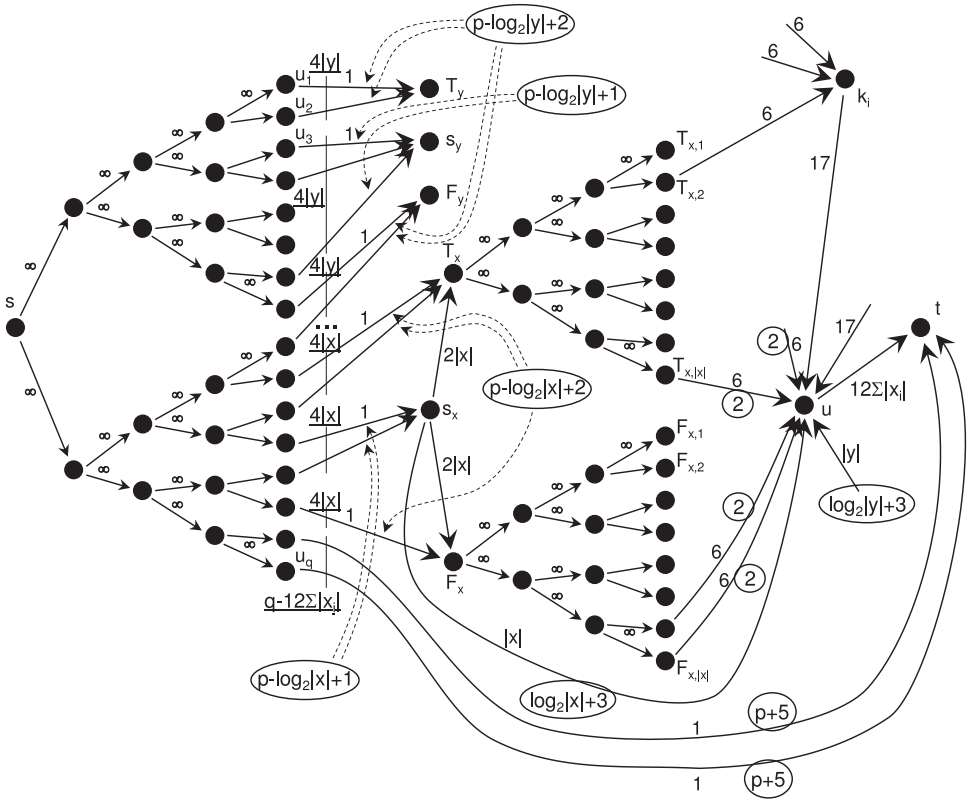


Fig. B13. Network for VRA-PO-GW-SD.

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 arrives 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. 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 that VRA-PO-GW-Q is NP-complete, too, by 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.

It can also be proven that VRA-PO is NP-complete. That proof, however, is quite long, hence it is omitted from this paper. Likewise, to save space only a sketch of the lengthy proof of the next theorem has been included here.

Theorem 16. *VRA-PO-GW-SD is NP-complete.*

Sketch of Proof. This proof is similar to the proof of Theorem 15. VRA-PO-GW-SD is in NP, for the same reasons as VRA-PO-GW is in NP.

The *network* is modified, and is shown in Fig. B13. The first difference between Figs. B12 and B13 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 are shown underlined. The rest of the 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 $T_{x,1}$, $T_{x,2}, \ldots, T_{x,|x|}$ and $F_{x,1}$, $F_{x,2}, \ldots, F_{x,|x|}$, respectively. They are connected to some k_i or to u, as described previously.

The link *capacities* are shown next to each link. Let $p = \max_{i=1\ldots n} \log_2 |x_i|$. All the *link weights* are 1 by default, except for the links where it is shown by the numbers in ellipse. Let the single *demand* be $s \to t$: q. $R = 1$. $\beta = 1$. This reduction is polynomial. The rest of this proof is similar to the proof of Theorem 15. □

B2. Inapproximability of VRA-PO

For stating theorems about the *approximability* of a problem the first step is to formulate them as an NP optimization (NPO) problem [32]. Again, several definition versions could have been listed here, but for simplicity we only list those two, which are crucial for the main inapproximability results:

Minimum Error with Virtual Resource Allocation–Peer Optimization with Given Weights (MIN-VRA-PO-GW, or shortly MVPG)

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

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 $E_n \leq |S_n| + R$ ($\forall n \in V$). (Here $|S_n|$ can be calculated from (V, F); E_n can be calculated from (V, F) and from the $\{e_l\}_{l\in F}$ set.)

Measure. $\beta = \max_l h_l/c_l$, the *maximum link utilization*. (h_l can be calculated from (V, F), the $\{e_l\}_{l\in F}$ set, the $\{w_l\}_{l\in F}$ set and from the set of demands.)

Goal. Minimize the measure.

Next we 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 solutions 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 the proof is essentially the same as the proof of VRA-PO-GW is in NP, presented in the proof of Theorem 15.

The Minimum Error with VRA-PO with Given Weights for a Single Demand (MIN-VRA-PO-GW-SD, or shortly MVPGS) version of the previous problem will also be important for the approximability results. MVGPS is essentially the same as MVPG, but it has only exactly one demand. This problem is an NPO as well, because the reasons listed for MVGP are all valid here, too.

The proof of Theorem 15 can be extended to show that generally the optimal solution cannot even be efficiently approximated:

Theorem 17. *No polynomial time algorithm exists that approximates the optimum of MVGP better than a factor of* 6/5.

Proof. This proof heavily relies on the proof of Theorem 15, using the same VRA-PO-GW instance bound to a 3SAT problem (see Fig. B12). This proof is also inspired by a similar reasoning presented in [9].

Finding an optimal resource allocation, which yields $\beta = 1$, is proven there to be NP-hard. We will now show that for the same instance any virtual resource allocation that results in $\beta > 1$ also results in $\beta \geq 6/5$.

We in fact prove an equivalent statement: for the given VRA-PO-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 k_i node 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 MVGP is not part of the a PTAS (Polynomial Time Approximation Scheme) class, as it is not possible to efficiently approximate the optimal solution within *every* constant ratio. The same is true for the single demand version of the problem, which will also be used later, in the proof of a stronger statement.

Theorem 18. *No polynomial time algorithm exists that approximates the optimum of MVGPS better than a factor of* 18/17.

The proof of this theorem relies on the proofs of Theorems 16 and 17, and is similar to the latter, but it is omitted to save space.

The next statement is the strongest in this section. It effectively states that no polynomial time algorithm exists that could approximate the optimal solution of MVGPS to *any* given constant ratio:

Theorem 19. *MVGPS is not part of the APX class.*

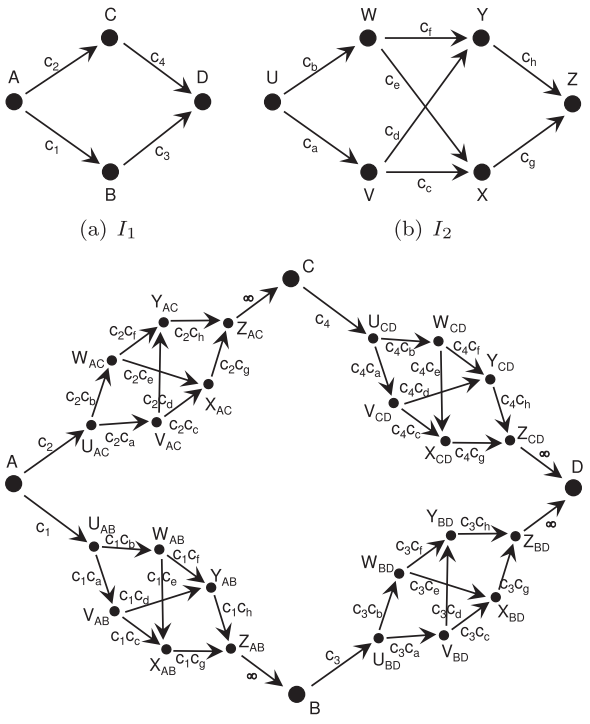


Fig. B14. MVGPS compounding.

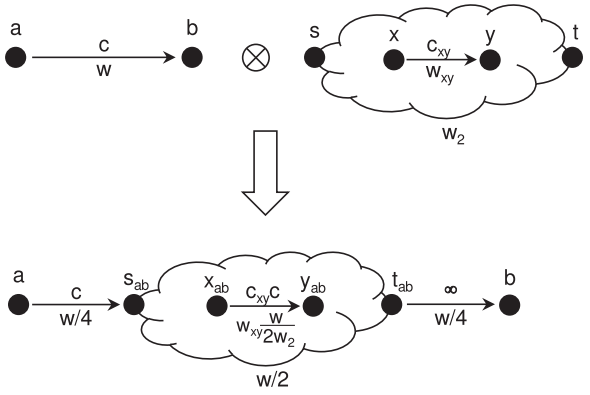


Fig. B15. $\otimes$ definition.

The proof of Theorem 19 applies the *inapproximability gap amplification technique*, which has recently been introduced in [19] to prove similar inapproximability for the OSPF ECMP link weight configuration problem.

Just like at the inapproximability proof in [19], we first introduce the $\otimes$ (compound) operator for MVGPS instances. From two instances I_1 and I_2 a new I instance can be crated by $I = I_1 \otimes I_2$, if both of the following conditions hold:

1. the demand to be transmitted in I_2 is 1,
2. R, the allowed maximum number of extra links is identical in I_1 and I_2.

An example of compounding is shown in Fig. B14. The capacities are shown next to the links and each link weight is one unit. In I_1 the demand is $A \to D$: 1, in I_2 it is $U \to Z$: 1. $R = 1$ in all these instances.

The formal definition of $I_1 \otimes I_2$ is the following (see Fig. B15). Take the I_1 network and replace each link in it with the following subnetwork. Let the original link in I_1 be ab with capacity c and

weight w, and the demand in I_2 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 w_2.

In $I_1 \otimes I_2$ create an as_{ab} link with capacity c and weight $w/4$. Also create a $t_{ab}b$ link with infinite capacity and $w/4$ weight. Between s_{ab} and t_{ab} insert the network of I_2: s replaced by s_{ab}, t becoming t_{ab}, x turning to x_{ab}, etc. For each xy link in I_2 with capacity c_{xy} and weight w_{xy} create an $x_{ab}y_{ab}$ link of capacity $c_{xy}c$ and weight $w_{xy}w/(2w_2)$. This way the total minimum weight of the ab subnetwork in $I_1 \otimes I_2$ 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 $I_1 \otimes I_2$ instance be equal to the demand of I_1, and R in the new instance be as it was in I_1 and I_2.

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

$$I_0 = \otimes^0 I = I$$

$$I_1 = \otimes^1 I = I \otimes I$$

$$I_2 = \otimes^2 I = I \otimes (\otimes^1 I) = I \otimes (I \otimes I)$$

$$\dots$$

$$I_k = \otimes^k I = I \otimes (\otimes^{k-1} I)$$

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

Now the following lemma can be proven:

Lemma 20. *Let I be an instance of MVGPS with $OPT(I) \geq 1$. Then $OPT(\otimes^k I) = (OPT(I))^{k+1}$ for any $k \in \mathbb{Z}^+$.*

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 we prove it for $k+1$.

First we 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 we 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 an ab link in I_0. Let $\beta^{k+1}_{as_{ab}} = \delta_{ab}$. There are two possibilities:

In the first case for all the ab links 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 ab link 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 $\beta_l^{k+1} < OPT(I)^{k+2}$. 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 MVGPS is not part of the APX class:

Proof (Theorem 19). The theorem states that for any constant factor $\alpha > 1$ there is no polynomial time algorithm that can find a solution to each MVGPS instance I with MLU less than $\alpha \cdot OPT(I)$. To show this, for each α we will create one MVGPS 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 16, with the network plot in Fig. B13. Divide each link capacity by q and let the demand be $s \to t$: 1. The rest of the instance (e.g. the link weights) are unaltered. We will call this instance I_0.

From the proof of Theorem 18 it follows that either $OPT(I_0) = 1$ or $OPT(I_0) = 18/17$ (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 MVGPS 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 20 $OPT(I_{k-1}) = OPT(I_0)^k$. This means that either $OPT(I_{k-1}) = 1$ or $OPT(I_{k-1}) = (18/17)^k \geq \alpha$ 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. □

References

[1] D. Wischik, M. Handley, M.B. Braun, The resource pooling principle, ACM SIGCOMM Comput. Commun. Rev. 38 (5) (2008) 47–52.

[2] IEEE Standard for Local and Metropolitan Area Networks-link Aggregation, IEEE Std 802.1AX-2008 (2008) 1–163.

[3] A. Vakali, G. Pallis, Content delivery networks: status and trends, IEEE Internet Comput. 7 (6) (2003) 68–74.

[4] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, A view of cloud computing, Commun. ACM 53 (4) (2010) 50–58.

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

[6] D. Kreutz, F.M.V. Ramos, P.E. Veríssimo, C.E. Rothenberg, S. Azodolmolky, S. Uhlig, Software-defined networking: a comprehensive survey, Proc. IEEE 103 (1) (2015) 14–76.

[7] J.T. Moy, OSPF Version 2, 2013, (RFC 2328).

[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), 2002, (ISO/IEC 10589:2002).

[9] B. Fortz, M. Thorup, Increasing internet capacity using local search, Comput. Optim. Appl. 29 (2004) 13–48.

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

[11] A. Maghbouleh, Metric-based traffic engineering: panacea or snake oil? a real-world study, in: Proceedings of the 27th North American Network Operators Group Meeting, NANOG27, 2003.

[12] S. Vissicchio, O. Tilmans, L. Vanbever, J. Rexford, Central control over distributed routing, in: Proceedings of ACM SIGCOMM, 2015, pp. 43–56.

[13] O. Tilmans, S. Vissicchio, L. Vanbever, J. Rexford, Fibbing in action: on-demand load-balancing for better video delivery, in: Proceedings of ACM SIGCOMM, 2016, pp. 619–620.
[14] K. Németh, A. Kőrösi, G. Rétvári, Optimal OSPF traffic engineering using legacy equal cost multipath load balancing, in: Proceedings of IFIP Networking Conference, 2013, pp. 1–9.
[15] K. Németh, A. Kőrösi, G. Rétvári, Enriching the poor man's traffic engineering: virtual link provisioning for optimal OSPF TE, in: Proceedings of the 16th International Telecommunications Network Strategy and Planning Symposium (Networks), 2014, pp. 1–7.
[16] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, X. Xiao, Overview and Principles of Internet Traffic Engineering, 2002, (RFC 3272).
[17] Y. Wang, Z. Wang, Explicit routing algorithms for internet traffic engineering, in: Proceedings of the Eight International Conference on Computer Communications and Networks, 1999, pp. 582–588.
[18] Z. Wang, Y. Wang, L. Zhang, Internet traffic engineering without full-mesh overlaying, in: Proceedings of INFOCOM 2001, 1, 2001, pp. 565–571.
[19] M. Chiesa, G. Kindler, M. Schapira, Traffic engineering with equal-cost-multipath: an algorithmic perspective, in: Proceedings of IEEE INFOCOM 2014, 2014, pp. 1590–1598.
[20] B. Fortz, M. Thorup, Internet traffic engineering by optimizing OSPF weights, in: Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2000, 2, 2000, pp. 519–528 vol.2.
[21] M. Pióro, Á. Szentesi, J. Harmatos, A. Jüttner, P. Gajowniczek, S. Kozdrowski, On open shortest path first related network optimisation problems, Perform. Eval. 48 (1–4) (2002) 201–223. Performance Modelling and Evaluation of ATM & IP Networks.
[22] Cisco Systems, Inc, Cisco Nexus 7000 Series NX-OS Unicast Routing Command Reference, 2016. Chapter: M Commands, Section: maximum-paths (EIGRP, IS-IS, RIP, OSPF, OSPFv3) URL http://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus7000/sw/unicast/command/reference/n7k_unicast_cmds/l3_cmds_m.html.
[23] Current Analysis, Inc., Product Assessment: Ericsson – SmartEdge Series, 2009. URL: http://archive.ericsson.net/service/internet/picov/get?DocNo=13/28701-FGB101647&Lang=EN&HighestFree=Y.
[24] Juniper Networks, Inc., JunosE™Software for E Series™Broadband Services Routers, Command Reference A to M, 2013, Release 14.3.x, Section: commands/maximum-paths, URL https://www.juniper.net/techpubs/en_US/junose14.3/information-products/topic-collections/command-reference-a-m/sw-cmd-ref-a-m.pdf.
[25] I. Pepelnjak, Unequal Cost Load-sharing, 2007. URL: http://blog.ipspace.net/2007/02/unequal-cost-load-sharing.html.
[26] M. Pióro, D. Medhi, Routing, Flow, and Capacity Design in Communication and Computer Networks, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.
[27] V.S. Mirrokni, M. Thottan, H. Uzunalioglu, S. Paul, A simple polynomial time framework for reduced-path decomposition in multipath routing, in: Proceedings of the Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM, 1, 2004, pp. 739–749.
[28] B. Vatinlen, F. Chauvet, P. Chrétienne, P. Mahey, Simple bounds and greedy algorithms for decomposing a flow into a minimal set of paths, Eur. J. Oper. Res. 185 (3) (2008) 1390–1401.
[29] T. Hartman, A. Hassidim, H. Kaplan, D. Raz, M. Segalov, How to split a flow? in: Proceedings of the IEEE INFOCOM, 2012, pp. 828–836.
[30] G. Rétvári, J.J. Bíró, T. Cinkler, On shortest path representation, IEEE/ACM Trans. Netw. 15 (6) (2007) 1293–1306.
[31] Maple (mathematical software), 2017. URL http://www.maplesoft.com/products/Maple/.
[32] V. Kann, On the Approximability of NP-complete Optimization Problems, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden, 1992 Ph.D. thesis.
[33] LEMON Graph Library – Library for Efficient Modeling and Optimization in Networks, Version 1.3.1, 2017. URL: http://lemon.cs.elte.hu/.
[34] 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, H. Ümit, An open source traffic engineering toolbox, Comput. Commun. 29 (5) (2006) 593–610.
[35] TOTEM Project: Toolbox for Traffic Engineering Methods, Latest version (3.2) released at June 6, 2008. URL http://totem.run.montefiore.ulg.ac.be/.
[36] H. Ümit, Interior Gateway Protocol Weight Optimization Tool, 2007. URL: http://www.poms.ucl.ac.be/totem/.
[37] S.D. Maesschalck, D. Colle, I. Lievens, M. Pickavet, P. Demeester, C. Mauz, M. Jaeger, R. Inkret, B. Mikac, J. Derkacz, Pan-European optical transport networks: an availability-based comparison, Photonic Netw. Commun. 5 (3) (2003) 203–225.
[38] A. Sridharan, R. Guérin, C. Diot, Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks, in: Proceedings of the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications, INFOCOM, 2, 2003, pp. 1167–1177.
[39] D. Xu, M. Chiang, J. Rexford, Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering, IEEE/ACM Trans. Netw. 19 (6) (2011) 1717–1730.
[40] M. Chiesa, G. Rétvári, M. Schapira, Lying your way to better traffic engineering, in: Proceedings of the ACM CoNEXT, 2016.
[41] S.I. Nikolenko, K. Kogan, A. Fernández Anta, Network simplification preserving bandwidth and routing capabilities, in: Proceedings of the 2017 IEEE International Conference on Computer Communications, INFOCOM, 2017.
[42] K. Németh, G. Rétvári, Traffic splitting algorithms in multipath networks: is the present practice good enough? in: Proceedings of the 15th International Telecommunications Network Strategy and Planning Symposium, NETWORKS, 2012, pp. 1–6.
[43] S. Kandula, D. Katabi, S. Sinha, A. Berger, Dynamic load balancing without packet reordering, SIGCOMM Comput. Commun. Rev. 37 (2) (2007) 51–62.
[44] I. Pepelnjak, Improving ECMP Load Balancing with Flowlets, 2015. URL: http://blog.ipspace.net/2015/01/improving-ecmp-load-balancing-with.html.
[45] A. Kabbani, B. Vamanan, J. Hasan, 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, CoNEXT '14, ACM, New York, NY, USA, 2014, pp. 149–160.

Krisztián Németh received his M.Sc. degree in Information Technology Engineering from the Technical University of Budapest, Hungary in 1998. Currently he is an assistant lecturer at the Budapest of University Technology and Economics, Department of Telecommunication and Media Informatics, finishing his Ph.D. His research interests include Traffic Engineering, Quality of Service and performance evaluation in IP networks.

Attila Kőrösi is an Assistant Research Fellow at the MTA-BME Information Systems Research Group, Hungary. He received a M.Sc. degree in Mathematics from the Budapest University of Technology and Economics in 2007. He has experiments in Wolfram Mathematica, C, JAVA and Linux. His research interest is mathematical modeling of routing protocols, FIB compression and data-centers. He has published several papers on P2P based multimedia services and their stochastic models.

Gábor Rétvári received the M.Sc. and Ph.D. degrees in electrical engineering from the Budapest University of Technology and Economics (BME), Budapest, Hungary, in 1999 and 2007, respectively. He is now a Senior Research Fellow at the Department of Telecommunications and Media Informatics, BME. His research interests include routing and switching in packet networks, switch and router design, clouds and software-defined networks, and the networking applications of information theory and computational geometry. He is a Perl hacker, maintaining numerous open source scientific tools written in Perl, C and Haskell.