



HAL
open science

Virtual Multi-Topology Routing for QoS Constraints

Nicolas Huin, Jeremie Leguay, Sébastien Martin

► **To cite this version:**

Nicolas Huin, Jeremie Leguay, Sébastien Martin. Virtual Multi-Topology Routing for QoS Constraints. NOMS 2024-2024 IEEE/IFIP Network Operations and Management Symposium, May 2024, Seoul, South Korea. hal-04453270

HAL Id: hal-04453270

<https://inria.hal.science/hal-04453270>

Submitted on 12 Feb 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

Virtual Multi-Topology Routing for QoS Constraints

Nicolas Huin*, Sébastien Martin†, Jérémie Leguay†

**IMT Atlantique, IRISA UMR CNRS 6074, F-35700 Rennes, France*

†*Huawei Technologies Ltd., Paris Research Center, France.*

Abstract—Multi-topology routing (MTR) provides an attractive alternative to segment routing for traffic engineering when network devices cannot be upgraded. However, due to a high overhead in terms of link state messages exchanged by topologies and the need to frequently update link weights to follow evolving network conditions, MTR is often limited to a small number of topologies and the satisfaction of loose QoS constraints. To overcome these limitations we propose vMTR, an MTR extension where demands are routed over virtual topologies that are silent, i.e., they do not exchange LSA messages, and that are continuously derived from a very limited set of real topologies, optimizing each QoS parameter. In this context, we present a polynomial and exact algorithm for vMTR and, as a benchmark, a local search algorithm for MTR. We show that vMTR helps reducing drastically the number of real topologies and that it is more robust to QoS changes.

Index Terms—Multi-Topology Routing, Traffic Engineering, Quality of Service, Segment Routing.

I. INTRODUCTION

Traffic Engineering (TE) policies [1] are key to controlling how bandwidth is shared in IP (Internet Protocol) networks. These policies can be tuned to satisfy application requirements in terms of end-to-end delay, packet loss, and jitter, for instance. They can be optimized by a centralized controller and applied by routers to steer traffic with data plane mechanisms such as Segment Routing (SR) [2], [3] or MPLS (Multi-Protocol Label Switching). However, most IP networks today still rely on shortest path and hop-by-hop routing from Interior Gateway Protocols (IGP).

Extensions of IGP protocols such as Multi-Topology Routing (MTR) [4], [5] or FlexAlgo [6] provide simple solutions to create and operate multiple routing planes, or “soft” slices [7], to support various QoS and operational requirements. Each routing plane minimizes a different additive link metric (e.g., IGP cost, TE metric, QoS) and can exclude some links when needed (i.e., *constraints* in FlexAlgo or *exclusions* in MTR). In theory, MTR and FlexAlgo only require a software upgrade of devices to update the control plane, and they can still rely on standard IP forwarding in the data plane. However, in practice, this is only the case for MTR, as FlexAlgo is always shipped along with SRv6 or SR-MPLS. Indeed, FlexAlgo’s implementations rely on SR labels to identify routing planes and forward traffic consistently on top of them. When SR is available, TE policies can be configured at ingress nodes to provide source-based routing on top of existing IGP routing planes (or “algorithms” when FlexAlgo is rolled out). SR policies consist of a label stack that is injected in packet headers to encode the sequence of shortest-path segments that need to be

traversed by packets. In this case, a data plane upgrade, i.e., device replacement or microcode update whenever possible, is necessary to support SR forwarding. Indeed, many legacy devices cannot support SR, as mentioned in [8].

In this paper, we consider deployment scenarios where most devices still operate with legacy data planes (i.e., without SR support). In this context, IGP extensions like MTR can be used to create multiple routing planes, but most routers have not been replaced by SR-capable devices. The most popular IGP protocol is Open Shortest Path First (OSPF) [9]. In this protocol, routers periodically broadcast Link State Advertisement (LSA) messages that contain metrics related to their adjacent links. Upon reception of LSA messages, routers store them in an LSDB (Link State DataBase) and maintain a Shortest Path Tree (SPT) towards all the other nodes, which decides the next hop to use for all possible destinations. This information is stored in the data plane in a Forwarding Information Base (FIB) and used to forward packets. When only one link metric is used (e.g., IGP cost), a single routing plane is available. The set of link metrics is generally *designed* [10] to satisfy some traffic scenarios. However, with a single routing plane for all applications, it is, in general, not possible to satisfy a heterogeneous set of QoS requirements.

To increase routing options, MTR offers the possibility of executing multiple IGP instances in parallel, each working over a different metric. The SPT maintained for each instance, or topology, generally aims at satisfying a set of QoS requirements (e.g., cost, hop count, delay) and administrative constraints (e.g., exclusion of links or nodes). FlexAlgo [6] can be seen as an extension of MTR and adds the possibility to build a routing plane over 1) a given metric, 2) considering some *constraints* (i.e., exclusions of nodes/links), and 3) a particular shortest path algorithm. Each *algo* is defined by the tuple (metric, constraints, SPT algorithm). Each MTR topology, or “algo” in FlexAlgo, has its own IP prefix so that traffic can be forwarded consistently in a hop-by-hop fashion. In both cases, link metrics also need to be periodically advertised using LSAs so that routers can update their TE database. To meet a variety of QoS constraints from applications, a large number of MTR topologies may be required to satisfy all application requirements. However, multiplying the sets of TE metrics, i.e., MTR topologies, yields a significant network overhead as LSA messages are sent for each metric. On top of that, as network conditions fluctuate over time, topologies may require frequent updates from the management system to follow the evolution of traffic and network performance. However, link

weights can only be updated at a slow pace, and they cannot follow the evolution of network quality over time.

To overcome this issue, we propose a new scalable and versatile IGP extension, called virtual MTR (vMTR), to dynamically maintain a set of *virtual routing topologies*. This solution aims at increasing routing flexibility in multi-topology routing to follow evolving network conditions and support heterogeneous QoS requirements while reducing the overhead from LSAs and weight updates. The main idea is to route flows over virtual topologies that are silent, i.e., they do not exchange LSA messages, and that are continuously derived from a very limited set of *real* MTR topologies, i.e., exchanging LSAs. Each real topologies computes shortest paths for a basic QoS parameter, e.g. delay or packet loss, while virtual ones are designed to generate paths meeting multiple QoS constraints.

Searching for a shortest path satisfying one or multiple end-to-end constraints such as latency or hop count usually involves algorithms relying on Lagrangian relaxation. These algorithms aim at finding feasible Lagrangian multipliers to modify link costs so that a shortest path provides a feasible solution. Well-known heuristic algorithms are LARAC [11], for one constraint, or GEN-LARAC [12] for multiple constraints. Inspired by this, we propose to derive virtual topologies from real ones applying a linear combination of their link metrics with designed multipliers. Compared to vanilla MTR, our vMTR solution reduces the overhead as each virtual topology is only defined by a set of multipliers, one per real IGP instance, and virtual topologies do not exchange LSAs. Furthermore, routing is automatically adjusted when basic QoS metrics vary and specific link metrics do not need to be updated when traffic conditions evolve.

In this paper, we first formulate the optimization problem to find optimal multipliers so as to minimize the number of virtual topologies in vMTR. We propose a polynomial and exact algorithm that finds the minimum set of Lagrangian multipliers to meet two QoS constraints for a set of demands. Then, as a benchmark, we revisit the IGP weight design problem [10] to minimize the number of topologies in MTR and derive a heuristic algorithm based on local search to solve the problem. Finally, we compare MTR and vMTR with numerical results and show that the number of real MTR topologies can be drastically reduced with vMTR when accepting all traffic demands. We also observe that vMTR topologies are more robust to QoS changes than MTR ones.

The rest of this paper is structured as follows. Related work is discussed in Sec. II. The system architecture and the problem formulation are introduced in Sec. III and IV. Sec. V presents algorithmic solutions. Sec. VI describes our numerical results and Sec. VII concludes this paper.

II. RELATED WORK

The design of link weights so that traffic requirements can be satisfied is known as the *IGP weight design* problem. This problem is a well-studied NP-hard problem [13].

Single topology routing. Fortz and Thorup [10], [14] have initially proposed a local search algorithm where the main

Symbol	Meaning
K	Set of demands
s_k, d_k	Source and destination for demand $k \in K$
r_a^t	Resource on arc $a \in A$ for topology t
T_Q	Set of <i>basic</i> topologies (each related to a different QoS metric)
T_D	Set of <i>designed</i> topologies
r_k^t	Max resource consumption for demand k over topology t
SPT^t	Shortest path tree on topology t
λ_t	Vector of $ T_Q $ multipliers to design vMTR topology t

TABLE I: List of key notations.

idea is to change a few link weights at each time step. They minimize a piece-wise linear cost function related to MLU (Maximum Link Utilization). Right after, a genetic algorithm has been proposed by Ericsson et al.[15] and later combined [16] with the original local search. Bley et al. [13] proposed an improved version of the local search algorithm. A math-heuristic based on column generation has also been proposed recently [17].

Other variants related to energy-minimization have been considered [18], [19]. Robust optimization variants have been studied to consider (weighted) link failures [20] or worst case variations of demands [21] (i.e., minimum / maximum values for each demand) and hoses [22] at ingress routers to bound aggregate traffic. Oblivious routing extensions where traffic can be load balanced over multiple paths have also been considered [23], [24] using the hose model. Other traffic polytopes have been used to model traffic uncertainty [25].

Beyond traditional optimization techniques, machine learning [26] have recently been used to solve the deterministic version of the problem. It is based on a smoothed formulation of the problem that is suitable for gradient descent.

Multi-topology routing. In the context of network slicing where end-to-end latency constraints must be met, a global optimization model along with results obtained with a genetic algorithm has been presented [7] to design multiple topologies (iteratively) and assign demands to them. Some papers have also proposed solution where topologies are designed offline to maximize edge-disjointness [27], [28] or path diversity [29]. These topologies are then used to route traffic based on actual QoS conditions. Other variants for MTR have been studied to consider robustness to traffic variations [30] and failure scenarios [31].

Our contributions. In this paper, we extend MTR with virtual topologies that are silent, i.e., they do not exchange LSA messages, and that are continuously derived from a very limited set of real IGP topologies, i.e., exchanging LSAs.

III. MTR AND vMTR FOR QoS CONSTRAINTS

This section introduces MTR, and the vMTR extension we propose, from a system perspective. As a preliminary step, we also first formulate the QoS routing problem they address.

A. QoS satisfaction problem

Let's represent a network as a directed graph $G = (V, A)$ where V is the set of nodes (i.e., routers) and A is the set

of arcs (i.e., links). A set of basic IGP topologies T_Q is associated with different QoS metrics such as delay, jitter or losses. On top of them, additional topologies, real for MTR or virtual for vMTR, can be designed to route demands that cannot be accepted on basic topologies. In this case, these topologies defined by the set of their link weights, also called resources, denoted r_a^t for arc $a \in A$ and topology $t \in T_Q$, need to be designed along with their mapping to demands. The set of traffic demands K is given as input. Each demand k goes from source node s_k to destination node d_k , and it has a set of QoS requirements that are defined by a maximum resource consumption r_k^t over basic (additive) QoS metrics from topology $t \in T_Q$. Note that packet loss can be made additive using the log function. The problem is to find, for every demand k , a routing path p_k satisfying a set of QoS constraints, as expressed as follows:

$$\sum_{a \in p_k} r_a^t \leq r_k^t \quad \forall t \in T_Q \quad (1)$$

We consider a practical use case scenario where the total amount of traffic with QoS constraints is much smaller than best effort traffic. QoS constrained traffic is prioritized as high priority traffic in the data plane and the sum of high priority traffic (i.e., QoS constrained traffic) is not enough to saturate the bottleneck capacity. For this reason, we omit capacity constraints in the design of topologies and we focus only on QoS constraints satisfaction.

B. MTR: Multi-Topology Routing

When relying on multi-topology routing, the goal is to design the minimum number of topologies so that all demands have their QoS requirements satisfied. As mentioned in Sec. II, this is typically realized by solving a series of inverse shortest path trees problems. When link weights are designed and deployed, traffic is routed over a set of T_D topologies, each one corresponding to a new IGP instance. Routers exchange periodically LSA messages to propagate weights and they maintain a minimum spanning tree SPT^t for every $t \in T_D$. Whenever network conditions evolve and the characteristic of links in terms of QoS changes, a significant update of the designed topologies, i.e. weights, may be required.

C. vMTR: virtual MTR

In the case of vMTR, additional virtual topologies need to be designed from the set of existing real topologies. The central idea behind vMTR is rooted in Lagrangian relaxation, similarly to the LARAC [11] or GEN-LARAC [12] algorithms that find a path satisfying one or multiple QoS constraints, respectively. Indeed, virtual topologies are derived from real ones using a set of Lagrangian multipliers. A set of multipliers, each one associated with a real topology, is used to calculate and maintain link weights in a virtual topology. In this way, virtual topologies are automatically updated when QoS metrics on real instances change. In addition, virtual topologies are completely silent and do not exchange LSAs.

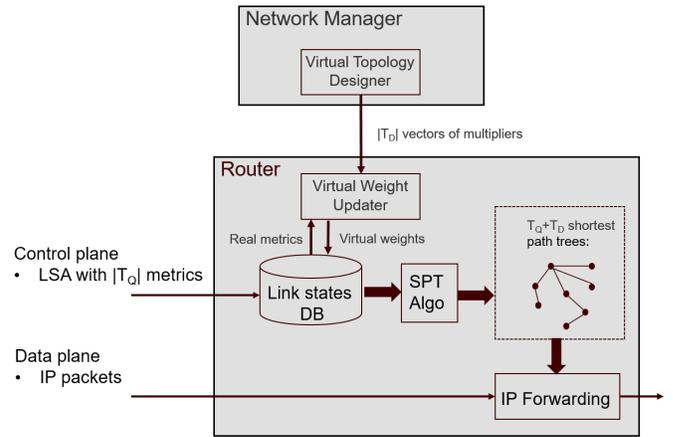


Fig. 1: System overview for vMTR.

For instance, a virtual topology can be created from two real topologies on loss and delay so that (virtual) link weights are defined by λ^l . Loss + λ^d . Delay with a vector λ of multipliers associated to the topology. Virtual link weights are used by the shortest path tree algorithm inside routers to find constrained paths with end-to-end loss and delay constraints. Recall that, in practice, packet loss is made additive using the log function. In the general case, metrics from real topologies can be combined with any algebraic operation. However, without loss of generality, we will consider in the rest of the paper a linear combination. Considering other operations may help to further increase routing diversity and constraint satisfaction, but at the cost of some extra complexity. Indeed, Lagrangian algorithms are actually heuristic to solve the constrained shortest path problem and may not find feasible solutions when they exist (see Sec. VI). However, by design, vMTR based on a linear combination is guaranteed to route all demands for which paths can be found with LARAC.

To create and manage virtual topologies, a virtual weight designer module sits in the network controller, as depicted in Fig. 1. It takes as inputs the network topology, link metrics related to “real” IGP instances (e.g., cost, capacity, delay), a traffic matrix with a set of demands and their attributes (e.g., source and, destination, QoS requirements). It produces a set of $|T_D|$ vectors of multipliers, one for every virtual topologies, along with the mapping of demands to each topology. In practice it sends to all routers $|T_D|$ vectors to create $|T_D|$ virtual topologies, each vector being of size $|T_Q|$, i.e., the number of existing / real IGP topologies. Every router continuously maintains a LSDB with the set of virtual link weights derived from real metrics. As virtual topologies are based on real ones, no specific LSAs need to be advertised. These topologies are almost immediately available to route traffic as routers do not need to wait for LSA exchanges. When metrics in real topologies are updated, all the virtual weights and their associated shortest path trees are updated locally by devices. Traditional Dynamic SPT [32] mechanisms implemented in IGP can be used to stabilize the system in case of frequent updates. No routing loops can occur as multipliers and virtual

metrics are positive. Unlike link weights in real topologies that need to be exchanged with LSAs, multipliers and the resulting virtual weights do not need to be integer. As for MTR, packets can be routed over a specific topology using QoS header fields like DSCP or ToS.

In terms of benefits and impact, one advantage of vMTR is the capability to explain a virtual topology as it is a combination of basic topologies related to metrics that users understand. This is impossible with MTR when weights are optimized by an algorithm. Actually, vMTR offers the possibility of manually or automatically setting multipliers to define a good tradeoff among basic topologies. An expert or customer can select the best multipliers according to his requirements. With regards to FlexAlgo, vMTR can be seen as a way to create new topologies (i.e., new metrics) over which "algorithms" are applied. We could even imagine an extension to the standard where multipliers are part of the algorithm definition and can be used to tune "algorithms" to meet QoS constraints. However, in this case, SR has to be rolled out.

Our proposition is in this paper two-fold: vMTR provides a way to create virtual topologies using multipliers and, on top of that, we propose in Sec. V-A an algorithm to automatically generate the said multipliers.

IV. TOPOLOGY DESIGN PROBLEMS

The optimization problem associated with the design of topologies for MTR, defined by link weights, consists in creating the minimum number of topologies such that all demands are accepted. This problem is similar in vMTR but differs in the way weights are computed. Indeed, weights of virtual topologies are derived from real topologies. This section presents the ILP (Integer Linear Programming) models for both cases.

In the rest of the paper, we will assume that MTR and vMTR are used to route demands that cannot be accepted over basic topologies $|T_Q|$ (each one maintaining a shortest path tree over a single QoS metric). Also, to ease the presentation, we will omit demands for which a constrained path cannot be found.

A. MTR

Let us introduce the variables to solve the topology design problem for MTR:

- $x_{a,k,t} \in \{0, 1\}$: equals 1 if the demand k is routed along a path using link a in the topology t , and 0 otherwise.
- $\pi_{u,v,t} \in \mathbb{R}^+$: is the potential of the node u , i.e., the distance between node u and v on the topology t .
- $u_{a,d,t} \in \{0, 1\}$: equals 1 if the link a belongs to a shortest path towards destination d on topology t , and 0 otherwise.
- $w_{a,t} \in \mathbb{R}^+$: the weight assigned to link $a \in A$ on the topology t .
- $y_{k,t} \in \{0, 1\}$: equals 1 if the demand k is assigned to the topology t (thus feasible for this topology), and 0 otherwise.
- $z_t \in \{0, 1\}$: equals 1 if the topology t is deployed.

Let us consider d_K the set of nodes that are destinations of at least one demand. We denote by $\omega^-(v)$ (resp. $\omega^+(v)$) the set of ingoing (resp. outgoing) links of node v .

As we do not know the needed number of topologies, we can consider an upper bound to create the right number of variables. This upper bound can be set by the number of designed topologies returned by the local search algorithm presented in the next section.

The ILP model is given by Fig. 2. This model is an extension of the well-known model from the literature (e.g., [30], [7]) to jointly create multiple topologies. The main goal is to minimize the number of designed topologies. Constraints (3) link the activation of topologies and demands routed on top of them. Inequalities (4) ensure that each demand is assigned to one designed topology. Equalities (5) are the well-known flow conservation constraints. Constraints (6) and (8) link routing variables and shortest path variables. Inequalities (7) guarantee that the induced shortest path tree is a tree, i.e., there is at most one outgoing link at each node. Constraints (9) and (10) ensure that the shortest path tree follows the weight w . Inequalities (11) are resource constraints.

B. vMTR

For vMTR we need to add the following variables:

- $\lambda_{t,\bar{t}} \in \mathbb{R}$: the multiplier associated with the basic topology $t \in T_Q$ and the virtual topology $\bar{t} \in T_D$.

and the following constraints:

$$w_{a,\bar{t}} = \sum_{t \in T_Q} \lambda_{t,\bar{t}} w_{a,t} \quad \forall a \in A, \forall \bar{t} \in T_D \quad (12)$$

Remark that in the case of vMTR, we complete the designed virtual topologies with real MTR topologies when all demands cannot be accepted with vMTR. To manage the two kinds of designed topologies, we penalize in the objective function the activation of MTR topologies.

V. ALGORITHMIC SOLUTIONS

This section presents algorithms for MTR and vMTR. While multiple algorithms exist for MTR (see Sec. II), we build upon one of them [13] to focus on QoS constraint satisfaction and minimize the number of designed topologies.

A. MTR Algorithm

We developed a greedy algorithm to design the sets of weights, i.e., topologies, as shown in Algo. 1. It iteratively generates random weight vectors, fine-tuned to maximize the number of accepted demands using a local search algorithm. MTR topologies are created one by one, each time trying to accept a maximum number of demands. If the local search fails to generate a weight vector accepting any demands, it falls back to computing a valid resource-constrained shortest path for one of the remaining demands (using the TAMCRA algorithm [33]), then uses that path for building a valid weight vector from a new random one.

$$\begin{aligned}
\min \quad & \sum_{t \in T_D} z_t & (2) \\
& \sum_{k \in K} y_{k,t} \leq z_t & \forall t \in T_D \quad (3) \\
& \sum_{t \in T_D} y_{k,t} \geq 1 & \forall k \in K \quad (4) \\
& \sum_{a \in \omega^+(v)} x_{a,k,t} - \sum_{a \in \omega^-(v)} x_{a,k,t} = \begin{cases} y_{k,t} & \text{if } v = s_k, \\ -y_{k,t} & \text{if } v = d_k, \\ 0 & \text{otherwise;} \end{cases} & \forall v \in V, \forall k \in K, \forall t \in T_D \quad (5) \\
& x_{a,k,t} \leq u_{a,d_k,t} & \forall a \in A, \forall k \in K, \forall t \in T_D \quad (6) \\
& \sum_{a \in \omega^+(v)} u_{a,d,t} \leq 1 & \forall v \in V, \forall d \in d_K, \forall t \in T_D \quad (7) \\
& u_{a,d,t} \leq \sum_{k \in K: d=d_k} x_{a,d_k,t} & \forall a \in A, \forall d \in d_K, \forall t \in T_D \quad (8) \\
& w_{(u,v),t} - \pi_{u,d,t} + \pi_{v,d,t} \geq 1 - u_{(u,v),d,t} & \forall (u,v) \in A, \forall d \in d_K, \forall t \in T_D \quad (9) \\
& w_{(u,v),t} - \pi_{u,d,t} + \pi_{v,d,t} \leq M(1 - u_{(u,v),d,t}) & \forall (u,v) \in A, \forall d \in d_K, \forall t \in T_D \quad (10) \\
& \sum_{a \in A} \sum_{t \in T_D} r_a^t x_{a,k,t} \leq r_k^t & \forall k \in K, \forall t \in T_D \quad (11)
\end{aligned}$$

Fig. 2: Integer linear problem for the design of topologies in MTR.

The greedy algorithm uses a run-of-the-mill local search that encodes a solution by its weight vector, and the neighborhood of a solution is the set of weight vectors that differ from the solution by a single element. The fitness function used to evaluate a solution computes the shortest path tree according to the weight vector and counts the number of demands with a path respecting their required QoS.

Bley et al. [13] already proposed a local search algorithm for the IGP weight design problem, but we consider different assumptions. Firstly, while they consider an extended neighborhood (compared to ours) that takes into account load balancing, we only focus on the weight vector variation. Secondly, while they consider weights in the range $[0, 20]$, we consider the whole range of possible OSPF weights ($[0, 2^{16}]$). The last point is particularly important since the definition of their neighborhood considers atomic changes in the weight (± 1), which is quite effective considering the range of the weights. However, since we allow a much larger range of weights, we need to define a neighborhood that guarantees changes in the shortest path trees induced by the weights.

We thus generate meaningful neighbors (see line 25) computing the minimum increase or decrease in weight needed to change the shortest path trees. We call these values the up and the down delta-weight of a link, respectively. We denote by $\pi(u, v)$ the distance of the shortest path from u to v with the current link weights.

First, we define the down delta-weight $\delta_v^-(a)$ of a link $a = (s_a, d_a)$ as the weight decrease needed to insert the link a in the current shortest path tree T_v , rooted at v , in the place of the link $a' = (s'_a, d_a)$. Formally, we are looking for a value

$\delta_v^-(a)$ such that

$$\begin{aligned}
\pi(v, s_a) + w_{(a)} - \delta_v^-(a) &< \pi(v, t_a), \\
\delta_v^-(a) &= \pi(v, s_a) + w_{(a)} - \pi(v, t_a)
\end{aligned}$$

Second, we define the up delta-weight $\delta_v^+(a)$ of a link $a = (s_a, d_a)$ as the weight increase needed to remove the link from the current shortest path tree SPT_v , rooted at v . Formally, we are looking for a value $\delta_v^+(a)$ such that

$$\pi(v, s_a) + w_{(a)} + \delta_v^+(a) > \min_{a' \in \omega^-(d_a) \setminus \{a\}} (\pi(v, s_{a'}) + w_{(a')}).$$

Thus, the up delta weight of a link a is given by

$$\delta_v^+(a) = \min_{a' \in \omega^-(d_a) \setminus \{a\}} (\pi(v, s_{a'}) + w_{(a')}) - \pi(v, s_a) - w_{(a)} \quad (13)$$

$$= \min_{a' \in \omega^-(d_a) \setminus \{a\}} \delta_v^-(a') \quad (14)$$

Once we computed the delta-weight of each link, we can propagate them (see line 40) up the tree. Indeed, any weight change occurring in the higher depth of the tree will impact the distance of the descendants. Given a link $a_1 = (s_1, d_{a_1})$ and its swapped link $a_2 = (s_2, d_{a_1})$, we can propagate their up and down delta-weight up to their lowest common ancestor in the tree. Formally, if u is the lowest common ancestor of s_{a_1} and s_{a_2} in SPT , and $p_T(v_1, v_2)$ is the path from v_1 to v_2 in SPT , we have the following properties:

$$\begin{aligned}
\forall a \in p_T(u, s_1), \delta^+(a) &= \delta^+(a_1), \\
\forall a \in p_T(u, s_2), \delta^-(a) &= \delta^-(a_2)
\end{aligned}$$

Finally, we need to consider all possible shortest path trees in the graph. We can combine their up and down delta-weight by taking the minimum over all trees.

Algorithm 1 Greedy algorithm for MTR

```
1: function GREEDY( $G = (V, A), d, l, K$ )
2:    $\mathcal{T} \leftarrow \emptyset$  ▷ Set of weight-demands pairs
3:   while  $K \neq \emptyset$  do
4:      $w \leftarrow \{\mathcal{U}(0, 65535) | a \in A\}$  ▷ random values
5:      $w \leftarrow \text{LOCALSEARCH}(G, w, d, l, K)$ 
6:      $K' \leftarrow \text{GETACCEPTEDDEMANDS}(G, w, d, l, K)$ 
7:     if  $K' = \emptyset$  then
8:        $p \leftarrow \text{TAMCRA}(G, d, l, K)$ 
9:        $w \leftarrow \{\mathcal{U}(0, 65535) | a \in A\}$  ▷ random values
10:      for all  $a \in p$  do
11:         $w_a \leftarrow 1$ 
12:       $K' \leftarrow \text{GETACCEPTEDDEMANDS}(G, w, d, l, K)$ 
13:       $\mathcal{T} \in \mathcal{T} \cup (w, K')$ 
14:       $K \leftarrow K \setminus K'$ 
15: function LOCALSEARCH( $G = (V, A), w, d, l, K$ )
16:    $\tilde{K} \leftarrow \text{GETACCEPTEDDEMANDS}(G, w, d, l, K)$ 
17:    $(w^*, K^*) \leftarrow (w, \text{GETACCEPTEDDEMANDS}(G, w, d, l, K))$ 
18:   for  $i \leftarrow 0$  to  $\text{maxIte}$  do
19:      $\mathcal{W} \leftarrow \text{GENERATENEIGHBORHOOD}(w, G, d, l)$ 
20:      $w \leftarrow \arg \max_{w \in \mathcal{W}} \text{GETACCEPTEDDEMANDS}(G, w, d, l)$ 
21:      $\tilde{K} \leftarrow \text{GETACCEPTEDDEMANDS}(G, w, d, l, K)$ 
22:     if  $|\tilde{K}| > |K^*|$  then
23:        $(w^*, K^*) \leftarrow (w, \tilde{K})$ 
24:   return  $w^*$ 
25: function GENERATENEIGHBORHOOD( $w, G = (V, A)$ )
26:    $(\delta^+, \delta^-) \leftarrow \text{COMPUTEDELTAWEIGHTS}(G, w)$ 
27:    $\mathcal{W} \leftarrow \emptyset$ 
28:   for all  $a \in A$  do
29:      $w^- \leftarrow w$ 
30:      $w_a^- \leftarrow w_a + \delta^-(a)$ 
31:      $w^+ \leftarrow w$ 
32:      $w_a^+ \leftarrow w_a + \delta^+(a)$ 
33:      $\mathcal{W} \leftarrow \mathcal{W} \cup \{w^+, w^-\}$ 
34:   return  $\mathcal{W}$ 
35: function COMPUTEDELTAWEIGHTS( $G = (V, A), w$ )
36:    $\delta^+ \leftarrow \{\infty | a \in A\}$ 
37:    $\delta^- \leftarrow \{\infty | a \in A\}$ 
38:   for all  $v \in V$  do
39:      $T, d \leftarrow \text{DIJKSTRASHORTESTPATHTREE}(G, v, w)$  ▷ Get shortest path tree with distances
40:     for all  $(s_a, t_a) \in A \setminus T$  do
41:        $v_A \leftarrow \text{LOWESTCOMMONANCESTOR}(T, s_a, t_a)$ 
42:       for all  $a' \in p_T(v_A, s_a)$  do
43:          $\delta^-(a') \leftarrow \min(\delta^-(a'), d(s_a) + w(a) - d(t_a))$ 
44:       for all  $a' \in p_T(v_A, t_a)$  do
45:          $\delta^+(a') \leftarrow \min(\delta^+(a'), d(s_a) + w(a) - d(t_a))$ 
```

B. vMTR Algorithm

The weight design problem for vMTR consists in maximizing the number of accepted demands (first objective) while minimizing the number of virtual topologies (second objective). To create a virtual topology with vMTR over 2 basic topologies (e.g., delay and loss), we consider a single multiplier λ that provides the following virtual weight $r_a^t = r_a^1 + \lambda r_a^2$ for each link $a \in A$. A multiplier λ is said to be feasible for demand k if the shortest path p over link metrics r_a^t satisfies the maximum constraint on the end-to-end consumption for both resources, i.e., on r_k^1 and r_k^2 .

In the following, we first present some properties for this

particular case with 2 resources and then, we propose an exact algorithm that solves the problem in polynomial time.

Lagrangian properties. As one multiplier can cover several demands, a natural question is: does there exist one or several intervals of feasible multipliers for a given demand? In this section, we provide an answer to this question that allows solving the vMTR problem in polynomial time afterward.

Let us introduce the Constraint Shortest Path problem (CSP) where the goal is to find a path minimizing a resource under an additive end-to-end resource constraint. In [34], the authors propose and analyze the LARAC algorithm, a polynomial time heuristic solution for the CSP problem based on Lagrangian relaxation. This algorithm computes a multiplier λ^* and the solution it returns is the shortest path p_{λ^*} in the graph where the cost of each link $a \in A$ is given by $r_a^1 + \lambda^* r_a^2$.

The most interesting claim proved in [34] for us is given below.

Claim 1. [34] *If $\lambda < \lambda^*$, then $\sum_{a \in p_\lambda} r_a^2 \geq r_k^2$ and if $\lambda > \lambda^*$, then $\sum_{a \in p_\lambda} r_a^2 \leq r^2$*

This claim is the core of our vMTR algorithm.

We denote by λ_k^{1*} (resp. λ_k^{2*}) the multiplier provided by LARAC algorithm when resource 2 (resp. resource 1) is minimized and resource 1 (resp. resource 2) is constrained. As proved in [34] this multiplier is optimal.

The following proposition ensures that, for a given demand, only one range of multiplier is feasible for the two resources.

Proposition 1. *For a given demand $k \in K$, a multiplier λ_k is feasible for k , if*

$$\lambda_k^{2*} \leq \lambda_k \leq \frac{1}{\lambda_k^{1*}}.$$

Proof. Let us consider the shortest path p where link costs $r_a^2 + \lambda_k^{1*} r_a^1, \forall a \in A$ are considered. The path p is also the shortest path in the graph where link costs are $r_a^1 + \frac{1}{\lambda_k^{1*}} r_a^2, \forall a \in A$. Indeed, for each $a \in A$, dividing $r_a^2 + \lambda_k^{1*} r_a^1$ by λ_k^{1*} induced $r_a^1 + \frac{1}{\lambda_k^{1*}} r_a^2$. Thus, thanks to the claim 1, all $\lambda_k^2 \geq \lambda_k^{2*}$ induced a shortest path where resource 2 is respected and all $\frac{1}{\lambda_k^1} \leq \frac{1}{\lambda_k^{1*}}$ induced a shortest path where resource 1 is respected. Thus all λ_k such that $\lambda_k^{2*} \leq \lambda_k \leq \frac{1}{\lambda_k^{1*}}$ are feasible for the two resources. \square

For a given multiplier λ , we may have several paths that are the shortest, but one of them can be unfeasible for resource 1 or 2. To address this issue, we modified the Dijkstra algorithm to find all shortest paths between two nodes in a graph and we slightly changed the λ value by adding an ϵ until all shortest paths are valid or only one path is the shortest.

In the following, for a given demand k , we denote $\lambda_k^{\min} = \lambda_k^{2*}$ and $\lambda_k^{\max} = \frac{1}{\lambda_k^{1*}}$. Thanks to the convergence of LARAC algorithm [34], λ_k^{\min} is minimum for each demand $k \in K$ and λ_k^{\max} is maximum for each demand $k \in K$.

From Proposition 1, we can derive an efficient polynomial algorithm to solve the weight design problem for vMTR. As a pre-processing step, we compute the multipliers λ_k^{\min}

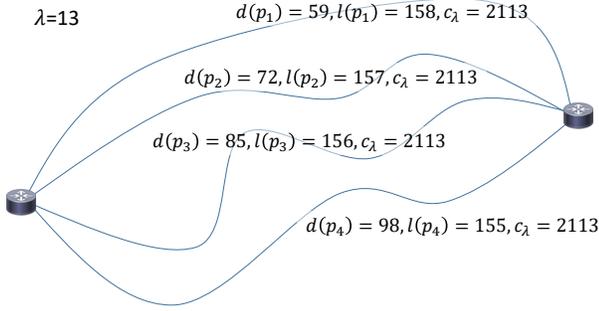


Fig. 3: Example of non shortest paths uniqueness with $\lambda = 13$.

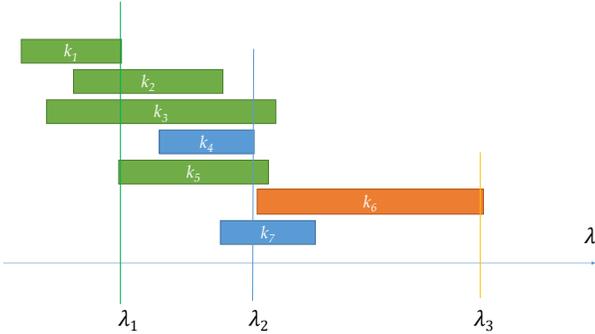


Fig. 4: Example with 6 demands. Intervals between λ_k^{\min} and λ_k^{\max} are represented for each demand k . The first multiplier λ_1 covers the green demands, the second multiplier λ_2 covers the blue demands and λ_3 covers the orange demand.

and λ_k^{\max} for each demand $k \in K$. Each demand where $\lambda_k^{\min} > \lambda_k^{\max}$ is discarded as it cannot be routed with vMTR. At the end of the procedure, the pool of discarded demand is processed with the MTR algorithm (Sec. V-A). The overall goal is to minimize the number of MTR topologies, as they generate LSAs and do not adapt to evolving network conditions.

Exact algorithm. The goal of this algorithm is to find a set of multipliers Λ such that all demands are covered. A multiplier λ covers a demand k if λ is feasible for the demand (see Proposition 1). At each step, we find a new multiplier to add in Λ such that some uncovered demands become covered with the new multiplier. More formally. Let \bar{K} be the set of demands covered at the current step, and respectively, $K \setminus \bar{K}$ the set of uncovered demands.

Algo. 2 describes the whole algorithm and an example is given in Fig. 4. First, the algorithm orders the λ_k^{\max} in increasing order. Then, at each step, it selects a $\lambda_{k^*}^{\max}$ to add in Λ such that all uncovered commodities have an associated λ_k^{\max} greater or equal to the selected $\lambda_{k^*}^{\max}$. It continues until all demands are covered.

Complexity and generalization. To summarize the approach above, the problem is solved in two steps. First, we define the space of valid multipliers for each demand. Second, we search the minimum number of multiplier vectors such that

Algorithm 2 Weight design algorithm for vMTR

```

1: function ( $G = (V, A), K$ )
2:    $\bar{K} = \emptyset, \Lambda = \emptyset$ 
3:    $\sigma$  ordering of demands  $k \in K$  in increasing order of  $\lambda_k^{\max}$ 
4:   while  $K \setminus \bar{K} \neq \emptyset$  do
5:      $\lambda^*$  the smallest  $\lambda_k^{\max}$  such that  $k \in K \setminus \bar{K}$ 
6:     add in  $\bar{K}$  all demands covered by  $\lambda^*$  not in  $\bar{K}$ 
7:     add  $\lambda^*$  in  $\Lambda$ 
8:   return  $\Lambda$ 

```

all demands are covered.

The dimension of the space of valid multipliers is equal to $|T_Q| - 1$. For the case with 2 basic topologies, we have only 1 dimension, which can be represented by an interval graph and the problem is polynomial as presented above. If we consider 3 basic topologies, we have 2 dimensions that can be represented by a plane. In this case, we can consider that the area of valid multipliers, vector of size 2, represents a rectangle, which is a particular case of our problem. This kind of graph is called boxicity 2-graph. For the interval graph or a boxicity 2-graph, the selection of the minimum number of multiplier vectors consists in solving a clique cover problem in the graph $H = (K, I)$ where K is the set of demands and I is the set of links. Two demands are linked if and only if they share a common valid multiplier vector. This problem is NP-hard for a boxicity 2-graph which is a particular case of the problem when 3 basic topologies are considered and polynomial for an interval graph [35].

VI. PERFORMANCE EVALUATION

We now compare MTR and vMTR over a set of instances generated from SNDlib [36].

Description of instances. These instances contain the geographical coordinates for each node and the link capacities between them. We selected 15 large topologies. For each network, we have considered two resources, the loss and the delay. For each link, the packet loss is set to be inversely proportional to the link capacity and the delay is considered to be proportional to the distance. To avoid trivial instances, we generated all demands between each pair of routers. We set the maximum end-to-end loss (resp. delay) to one of the shortest path on the loss (resp. delay) minus ϵ . This ensures that the demands cannot be routed on basic topologies, i.e., loss or delay. For each demand, we solved optimally, i.e., not using LARAC, the constraint shortest path problem to ensure that a solution exists. If no solution exists for a demand then we did not consider it. All instances are available at [37].

Numerical results. Fig. 5 presents violin plots to summarize results over all 15 instances. We can see, in Fig. 5a, that vMTR reduces the number of real topologies compared to MTR, from an average of 8.41 to 5.41. It also reduces the maximum number of needed topologies from 20 to 12, and it also avoid the need for any real topology in some cases. vMTR requires an average of 4.08 virtual topologies. The decrease in the number of real topologies can be easily explained by looking at the number of average demands per topologies in

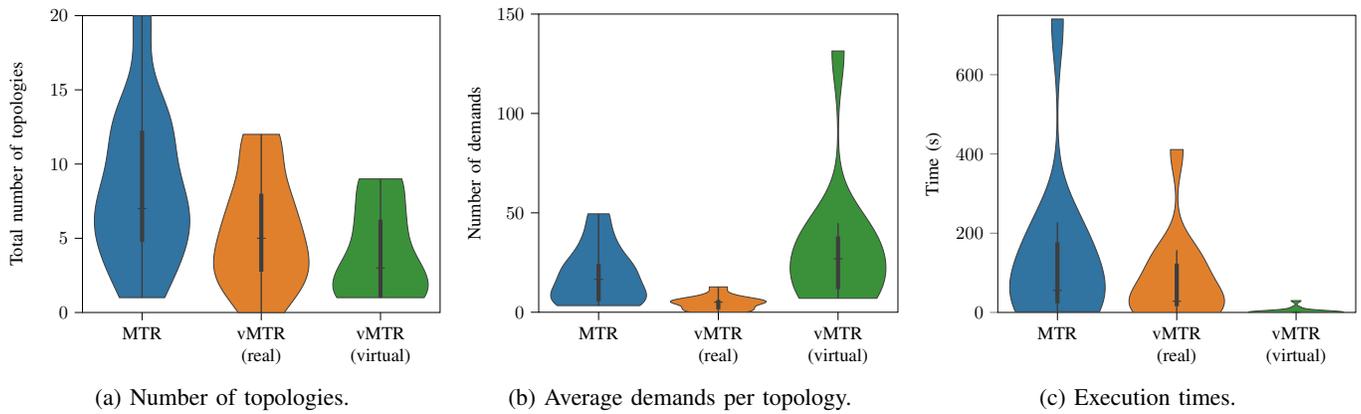


Fig. 5: Comparison of MTR and vMTR using violin plots.

Fig. 5b. vMTR virtual topologies can accept more demands than MTR real ones: in average, a vMTR topology holds 32.7 demands while, in average, MTR topologies hold 18.7 demands. In some cases, we can even have virtual topologies with 131.4 demands. However, real topologies in vMTR hold less demands because the remaining demands not covered by a virtual topology are harder to cover with the greedy MTR algorithm. Recall that as our implementation of vMTR with a linear combination of real topologies is based on the same principle as LARAC, which is a heuristic solution to the CSP problem, some demands cannot be satisfied with virtual topologies and real ones need to be added to ensure 100% traffic acceptance.

Fig. 5c shows the execution times for both solutions. For vMTR, we present the time required to compute lambda multipliers for virtual topologies as well as the time needed by the greedy algorithm to provision the remaining demands over real topologies. vMTR, once again, shows an improvement over MTR with an average time of 86.2s over 141.3s, and it also decreases the maximum time from 740.5s to 411.1s. The computational time to compute only the virtual topologies is really small. It is due to the polynomiality of the algorithm.

Finally, we study the robustness of the two approaches

against changes of QoS metrics. To this end, we define the *QoS robustness* of a demand, for a given QoS metric, as the ratio of its provisioned path QoS over its end-to-end QoS requirement. A QoS robustness of 1 means that the provided path QoS equals the required QoS, and any increase of this QoS metric in the path's link would invalidate the path. While we can see in Fig. 6 that both solutions provision demands on strict paths (ratio of 1), on average, vMTR provides paths that are more robust to QoS changes than MTR (0.75 and 0.79 for vMTR vs 0.85 and 0.88 for MTR, for delay and losses, respectively). Remark that the robustness is slightly better from the delay point of view if we compare with the Loss. This is due to the distribution of metric values in our instances. For the two solutions, we have a better maximum robustness for the delay than for the loss. We then have more flexibility to improve the robustness of the delay.

VII. CONCLUSION

We introduced a new IGP extension, called virtual MTR (vMTR), to dynamically maintain a set of virtual routing topologies. This solution aims at increasing routing flexibility in multi-topology routing to follow evolving network conditions and support heterogeneous QoS requirements while reducing the overhead from LSAs and weight updates. We presented a polynomial and exact algorithm for vMTR when virtual topologies are derived from two real topologies and a local search algorithm for MTR as benchmark. We showed that vMTR can help reducing the number of real topologies and is more robust to QoS changes.

Future work along these lines include several directions. First, we may propose an efficient algorithm for the design of MTR topologies jointly with the creation of virtual ones in vMTR. We can also design lambda multipliers considering traffic scenarios or uncertainty sets on traffic. Also of interest, we may consider non additive and non linear ways to combine real topologies. Finally, we may develop efficient algorithms for the hard case where more than two real topologies are used to derive virtual ones.

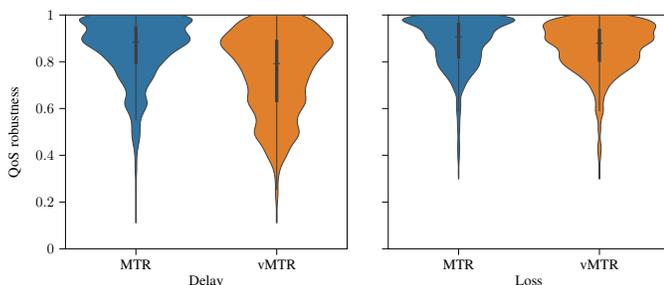


Fig. 6: Violin plots of the QoS robustness which is, for a demand and a metric, the ratio between the end-to-end QoS of the provided paths and the requested QoS. A value of 1 for a given metric means that the path of a demand is at its maximum resource consumption.

REFERENCES

- [1] N. Wang, K. H. Ho, G. Pavlou, and M. Howarth, "An overview of routing optimization for internet traffic engineering," *IEEE Communications Surveys Tutorials*, vol. 10, no. 1, pp. 36–56, 2008.
- [2] P. L. Ventre, S. Salsano, M. Polverini, A. Cianfrani, A. Abdelsalam, C. Filsfils, P. Camarillo, and F. Clad, "Segment routing: a comprehensive survey of research activities, standardization efforts, and implementation results," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 182–221, 2020.
- [3] D. Wu and L. Cui, "A comprehensive survey on segment routing traffic engineering," *Digital Communications and Networks*, 2022.
- [4] P. Psenak, S. Mirtorabi, A. Roy, L. Nguyen, and P. Pillay-Esnault, "Multi-topology (MT) routing in OSPF. IETF RRC 4915." Tech. Rep., 2007.
- [5] T. Przygienda, N. Shen, and N. Sheth, "M-ISIS: multi topology (MT) routing in Intermediate System to Intermediate Systems (IS-ISs). IETF RFC 5120." Tech. Rep., 2008.
- [6] P. Psenak, S. Hegde, C. Filsfils, K. Talaulikar, and A. Gulko, "IGP Flexible Algorithm," RFC 9350, Feb. 2023.
- [7] N. Huin, S. Martin, J. m. Leguay, and S. Cai, "Network slicing with multi-topology routing," in *2021 IFIP Networking Conference (IFIP Networking)*. IEEE, 2021, pp. 1–2.
- [8] Y. Tian, Z. Wang, X. Yin, X. Shi, Y. Guo, H. Geng, and J. Yang, "Traffic engineering in partially deployed segment routing over ipv6 network with deep reinforcement learning," *IEEE/ACM Transactions on Networking*, vol. 28, no. 4, pp. 1573–1586, 2020.
- [9] J. T. Moy, *OSPF: anatomy of an Internet routing protocol*. Addison-Wesley Professional, 1998.
- [10] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc. IEEE INFOCOM*, vol. 2, 2000, pp. 519–528.
- [11] A. Juttner, B. Szviatovski, I. Mécs, and Z. Rajkó, "Lagrange relaxation based method for the qos routing problem," in *Proc. IEEE INFOCOM*, vol. 2, 2001, pp. 859–868.
- [12] Y. Xiao, K. Thulasiraman, and G. Xue, "Gen-larac: A generalized approach to the constrained shortest path problem under multiple additive constraints," in *International Symposium on Algorithms and Computation*. Springer, 2005, pp. 92–105.
- [13] A. Bley, "Routing and capacity optimization for IP networks," in *Operations Research Proceedings 2007*. Springer, 2008, pp. 9–16.
- [14] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS weights in a changing world," *IEEE journal on selected areas in communications*, vol. 20, no. 4, pp. 756–767, 2002.
- [15] M. Ericsson, M. G. C. Resende, and P. M. Pardalos, "A genetic algorithm for the weight setting problem in OSPF routing," *Journal of combinatorial optimization*, vol. 6, no. 3, pp. 299–333, 2002.
- [16] L. S. Buriol, M. G. Resende, C. C. Ribeiro, and M. Thorup, "A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing," *Networks: An International Journal*, vol. 46, no. 1, pp. 36–56, 2005.
- [17] A. Benhamiche, M. Chopin, and S. Martin, "Unsplittable shortest path routing: Extended model and matheuristic," in *Proc. CoDIT*, 2023.
- [18] A. Capone, C. Cascone, L. G. Gianoli, and B. Sanso, *OSPF optimization via dynamic network management for green IP networks*. IEEE, 2013.
- [19] E. Amaldi, A. Capone, and L. G. Gianoli, "Energy-aware IP traffic engineering with shortest path routing," *Computer Networks*, vol. 57, no. 6, pp. 1503–1517, 2013.
- [20] B. Fortz, M. Thorup *et al.*, "Robust optimization of OSPF/IS-IS weights," in *Proc. INOC*, vol. 20, 2003, pp. 225–230.
- [21] J. Vallet and O. Brun, "Online OSPF weights optimization in IP networks," *Computer Networks*, vol. 60, pp. 1–12, 2014.
- [22] R. S. Ranaweera, I. M. Kamrul, and E. Oki, "Preventive start-time optimization of OSPF link weights against link failure for hose model," in *Asia-Pacific Conference on Communications (APCC)*. IEEE, 2012, pp. 698–702.
- [23] A. Altin, B. Fortz, and H. Ümit, "Oblivious OSPF routing with weight optimization under polyhedral demand uncertainty," *Networks*, vol. 60, no. 2, pp. 132–139, 2012.
- [24] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. van der Merive, "A flexible model for resource management in virtual private networks," *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 4, p. 95–108, Aug. 1999.
- [25] J. Moulrierac and T. K. Phan, "Optimizing IGP link weights for energy-efficiency in multi-period traffic matrices," *Computer Communications*, vol. 61, pp. 79–89, 2015.
- [26] M. Kodialam and T. Lakshman, "Network link weight setting: A machine learning based approach," in *Proc. IEEE INFOCOM*, 2022, pp. 2048–2057.
- [27] N. Wang, K.-H. Ho, and G. Pavlou, "Adaptive multi-topology IGP based traffic engineering with near-optimal network performance," in *IFIP NETWORKING*. Springer, 2008, pp. 654–666.
- [28] N. Wang, K. H. Ho, and G. Pavlou, "Ample: an adaptive traffic engineering based on virtual routing topologies," *IEEE Communications Magazine*, vol. 50, no. 3, pp. 185–191, 2012.
- [29] E. Mirzamany, A. Lasebae, and O. Gemikonakli, "An efficient traffic engineering based on multi-topology routing for future internet," *Computer Networks*, vol. 70, pp. 170–178, 2014.
- [30] X. Wang, S. Wang, and L. Li, "Robust traffic engineering using multi-topology routing," in *GLOBECOM 2009-2009 IEEE Global Telecommunications Conference*. IEEE, 2009, pp. 1–6.
- [31] M. Menth and R. Martin, "Network resilience through multi-topology routing," in *DRCN*, 2005, pp. 271–277.
- [32] D. Garg *et al.*, "Dynamizing dijkstra: A solution to dynamic shortest path problem through retroactive priority queue," *Journal of King Saud University-Computer and Information Sciences*, vol. 33, no. 3, pp. 364–373, 2021.
- [33] H. De Neve and P. Van Mieghem, "Tamcra: a tunable accuracy multiple constraints routing algorithm," *Computer Communications*, vol. 23, no. 7, pp. 667–679, 2000.
- [34] X. Ying, T. Krishnaiyan, X. Guoliang, A. Jüttner, and S. Arumugam, "The constrained shortest path problem," *AKCE International Journal of Graphs and Combinatorics*, vol. 2, no. 2, pp. 63–86, 2005.
- [35] H. Imai and T. Asano, "Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane," *J. Algorithms*, vol. 4, pp. 310–323, 1983.
- [36] S. Orłowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "SNDlib 1.0—Survivable network design library," *Networks: An International Journal*, vol. 55, no. 3, pp. 276–286, 2010.
- [37] N. Huin, S. Martin, and J. Leguay, "Loss/delay instances for vIGP," Jan. 2024. [Online]. Available: <https://doi.org/10.5281/zenodo.10469172>