Partitioning Traffic Engineering in Software Defined Wide Area Networks

Yufeng Xin and Yifei Wang
RENCI, University of North Carolina at Chapel Hill
Chapel Hill, NC, USA

***Abstract*—Traffic engineering (TE) is experiencing a surge in research and development interests due to the rise of software-defined networking and the massive increase in data volume in wide area networks. State-of-the-art TE systems address computational complexity and resiliency challenges in centralized controllers through aggregation or partitioning of the underlying network topology. This paper introduces a new TE load balancing method within a novel scalable SD-WAN TE controller framework that partitions traffic demands into multiple groups. Based on an optimization formulation with bandwidth and latency constraints, we show that effective partitioning algorithms allow solving a set of smaller optimization problems with much reduced computation time. Numerical analysis validates the viability of the proposed approach and demonstrates that a small number of traffic groups can achieve the desired TE performance.**

I. Introduction

Traffic engineering (TE) has long been a visionary goal in optimizing the performance and utilization of network operations. The traditional networks operate under the distributed control plane and protocols, which seriously limited the wide deployment and operation of TE systems [1].

In recent years, the emerging SDN (Software Defined Networking) paradigm reignited the research and development interests in TE, especially in large private Cloud wide-area networks (WAN) such as Google Cloud [2] and Microsoft Azure [3]. There are several main drives behind this renewed TE campaign. The exponentially growing data traffic volume over the WAN forced the Cloud providers to squeeze the network capacity as much as possible while not sacrificing the network congestion performance. The logically centralized controller permits solving global TE optimization problems periodically to cope with the dynamic demands and network states. The capability to configure a large number of switches in a short time (under a second) from the central SDN controller is a big advantage as it allows more computation time given to the TE solver between network re-optimization and reconfiguration cycles.

Yet, TE optimization in large-scale operational networks remains extremely challenging mainly because of its inherently high computational complexity rooted in the multi-commodity flow problem. The complexity of TE solutions is proportional (exponentially) to the size of the network, the size of the traffic matrix, and the constraints of the demands. To address the computational complexity and the *single-point-failure* problem of the centralized controller, the state-of-art TE systems scale down the TE model either by contracting the topology into multiple abstraction levels [4] or partitioning the topology into multiple segments, each of which is served by a separate SDN controller [3]. These topology abstraction and partitioning algorithms are heuristic in nature. They also introduce extra complexities in coordinating the routing and traffic forwarding between the segments, including avoiding the possible route loop and forwarding inconsistency. To further reduce the complexity, these solutions rely on a *path-based* flow maximization formulation that needs pre-computed paths between every node pair [5]. The link interdependency between the pre-computed paths could negatively affect the solution quality. Furthermore, they only focused on the bandwidth constraint. Other important constraints, such as latency, are largely ignored.

In this work, we tackle the TE complexity challenges from a different scaling dimension: partitioning the traffic demand instead. This is primarily based on the observation that solving the TE optimization problem with a small set of flows, even the one with multiple constraints on big networks, can be done very efficiently with a modern optimization solver in high-end computers. From the SDN controller perspective, this approach may ultimately lead to a viable TE load balancer architecture where multiple controllers co-exist to serve different demand groups in parallel. Such a TE load balancer system has other appealing properties that include much simplified inter-controller coordination and guaranteed loop-free routes. It also enables designing customized TE policies depending on the demand grouping strategies and the sequence of the group TE optimization and configuration.

We present the new TE load balancing method within a scalable SD-WAN control framework. Based on an *link-based* optimization formulation with both bandwidth and latency constraints, we show that effective partitioning algorithms allow solving a set of smaller optimization problems to generate the desired TE results with much reduced computation time. We studied multiple partitioning algorithms from the NP-Hard bin packing and number partitioning problems. We particularly studied two common TE objective functions in link utilization and cost and looked into the uneven link utilization phenomenon. Numerical analysis on realistic WAN topologies and heavy traffic loads validates the viability of the proposed approach with a small number of traffic groups.

The rest of this paper is organized as follows. We first define the TE optimization problem with an integer linear programming (ILP) model in Section II. In Section III, we describe a new TE load balancer framework and present the associated demand partition algorithms. The performance evaluation results are presented in Section IV. We summarize the related work in TE optimization and recent advancement in Section V. The paper is concluded in Section VI.

II. Problem Formulation and Optimal Solution

In this section, we formally define the optimal traffic engineering problem with node-arc based integer linear programming (ILP) model and present a motivating example. We explicitly add both bandwidth and latency constraints into the formulation and consider two different objective functions. Considering latency constraints explicitly would also prevent solutions from the ones with very long paths.

A. Problem Formulation

A wide-area network is modeled as a bi-directional graph $G(V, E)$ with a set of nodes V connected by a set of links E. Each link $e(u, v) \in E$, with its start node u and end node v, has a set of properties such as the maximum available bandwidth capacity W_e and the minimum transportation latency L_e. It may also carry a cost function c_e.

The traffic demands are defined as a set of commodities D. Each $d(s, t) \in D$ is defined as a demand between a source node $s \in V$ and a destination node $t \in V$ with bandwidth requirement w_d and a latency constraint l_d. Together these demands are represented in a traffic matrix (TM). A demand $d(s, t)$ is routed over a path from the node s to the node t in the network where a binary variable $x_{e,d}$ is defined to represent if the edge e is on the path d in the TE solution.

We first define the ILP model using a global load balancing objective to minimize a utility function, as defined in (1).

$$\textit{minimize} \quad \sum_{d=1}^{|D|} \sum_{e=1}^{|E|} \frac{w_d x_{e,d}}{W_e} \tag{1}$$

$$\textit{s.t.} \quad \sum_{u \in V} x_{e(u,w),d} - \sum_{u \in V} x_{e(w,u),d} = 0, \forall d(s,t), w \neq s, t \tag{2}$$

$$\sum_{u \in V} x_{e(u,s),d} - \sum_{v \in V} x_{e(s,v),d} = -1, \forall d(s,t) \tag{3}$$

$$\sum_{u \in V} x_{e(u,t),d} - \sum_{v \in V} x_{e(t,v),d} = 1, \forall d(s,t) \tag{4}$$

$$\sum_{d=1}^{|D|} x_{e,d} w_d \leq W_e, \forall e \in E \tag{5}$$

$$\sum_{e=1}^{|E|} x_{e,d} L_e \leq l_d, \forall d \in D \tag{6}$$

$$x_{e,d} \in \{0, 1\}, \quad \forall e \in E, \forall d \in D \tag{7}$$

Constraints (2), (3), and (4) represent the flow conservation conditions for every flow $d(s, t)$ from the network flow model. Inequality (5) is the link capacity constraint for all the network links to carry the traffic matrix D. (6) is the latency constraint for every flow d. Finally, the integer constraint (7) represents that this is an unsplittable flow model.

The objective function can also be defined to minimize a global cost function in (8). This objective function may find important applications in some operational networks since the link cost function can be customized. For example, certain transit network service providers need to pay for the underneath leased lines of fibers that have monetary values associated with the link cost. The link cost can also be defined as proportional to a link latency property such that the overall traffic latency can be minimized. In the most basic definition, the link cost can be defined as one, which will lead to a constrained shortest hop routing solution.

$$\textit{minimize} \quad \sum_{d=1}^{|D|} \sum_{e=1}^{|E|} c_e x_{e,d} \tag{8}$$

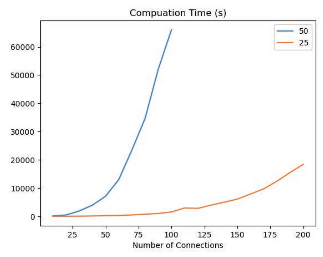
This ILP model has $2|E|$ variables and $|V| * |D| + 2|E|$ constraints. Typical WAN topologies have varying numbers of nodes and links. They are normally at least 2-connected to accommodate the failure protection and restoration requirement. Well-known Internet backbone network topologies that many studies used, such as the Internet 2, GEANT, Abilene, *et al.* , have a couple of dozens of nodes and relatively low link density (around 20% comparing to a full-mesh topology) [6]. The commercial Cloud WANs may have more nodes but similar connectivity [3]. Traffic demand volume and QoS characteristics are critical to TE. A most recent large-scale study concludes that Internet traffic volume follows long tail lognormal distribution where a small number of flows demand large bandwidth [7]. As the main purpose of TE is to make the network adaptive to network state and traffic changes, the TE solver needs to be invoked with new inputs repeatedly, where the time window ranges from a few minutes to dozen(s) minutes in the state-of-the-art system. All these factors are considered to guide our study in this work.

The Cloud WAN operates on aggregated traffic that supports multi-path routings between a pair of Cloud sites with complete controls. It makes more sense for them to adopt a splittable commodity flow formulation that essentially removes the integer constraints, enables the use of the much faster Linear Programming (LP) solution, and achieves high utilization. Since we are more interested in benchmarking the solution performance for general Internet backbone WANs, we assume all the demands are unsplittable. We also add the extra latency constraints to accommodate the QoS requirements common to many WAN applications.

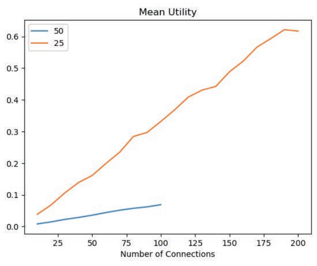
B. Optimal Solution and Traffic Load: Motivating Results

To gain more insights and motivate the main algorithmic idea presented in this paper, we first show some optimal solution results we obtained from solving the above TE optimization problem with a representative network topology and varying traffic demands.

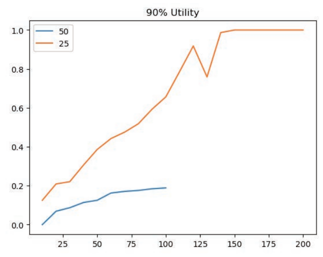
Figure 1 shows four solution metrics of our major concern for two network topologies of 25 and 50 nodes each and a



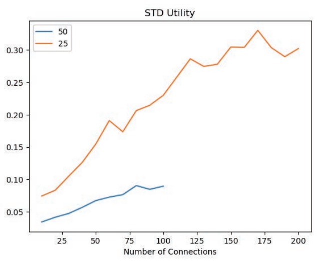
(a) Computation Time



(b) Mean Utility



(c) 90 Percentile Utility



(d) Utility Standard Deviation

Fig. 1: Load Balancing Optimal Solution

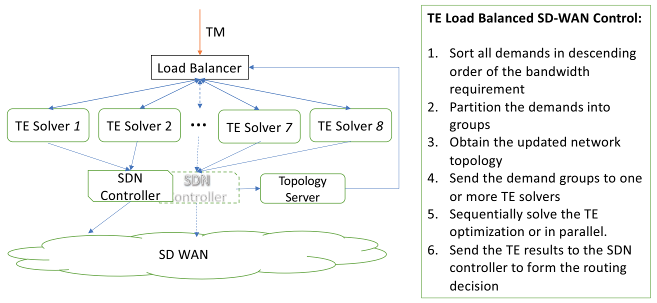
varying number of demands in the TM whose volumes follow a Lognormal distribution. Figure 1 (a) shows the computational time using a state-of-the-art MIP solver. Figure 1 (b) and Figure 1 (c) depict the mean utility and top 10% most occupied over all the links. Figure 1 (d) shows the standard deviation of the link utilization of all links. Unsurprisingly, the computational time increases so fast with more demands in the TM for the bigger network that we used up to 200 demands in the 25-node topology and only up to 100 demands in the 50-node topology. We note that in reality part of the nodes in a WAN are transit routers/switches that do not originate and receive traffic. The scale of the TM we considered in the study covers the extremely heavy traffic load situations.

The link utilization shows high variation, and part of the links would become fully saturated, especially in the high load scenario, while the average utilization can go beyond 60%. Since high network utilization may directly lead to congestion and long packet delay, the results demonstrate the value of obtaining the optimal solution. It also implies there is plenty of unused bandwidth in many links for less constrained demands. Overall the result shows the importance of comparing the detailed utilization distribution when we design and compare different TE algorithms.

The significant gap in the computational time between the 25-node and 50-node topologies validates the viable network partition based heuristics. It also suggests the potential to scale down the TE solver on the other dimension: the number of demands. When the number of demands is small, the TE solvers can obtain the optimal solution rather fast. For example, it only took less than 15 seconds for 10 demands in the 25-node network and less than 150 seconds in the 50-node network. The challenge is to identify effective grouping algorithms to achieve overall TE goals.

III. Load Balanced TE System and Algorithm Design

Our overarching goal is to develop a high-performance TE solver for a scalable SD-WAN control software framework, as shown in Figure 2. The left diagram depicts the targeted control software architecture that contains a TE load balancer and multiple TE solvers. The TE load balancer is responsible for partitioning a given TM into groups and dispatching traffic groups to the TE solvers for routing solutions. The routing results will be fed to the SDN controller to form the forwarding rules that are used to program the routers/switches in the network substrate. The *Topology Server* is responsible for providing the real-time network topology and state information to the TE component.



TE Load Balanced SD-WAN Control:

1. Sort all demands in descending order of the bandwidth requirement
2. Partition the demands into groups
3. Obtain the updated network topology
4. Send the demand groups to one or more TE solvers
5. Sequentially solve the TE optimization or in parallel.
6. Send the TE results to the SDN controller to form the routing decision

Fig. 2: TE Load Balanced SD-WAN Control

On the right, the proposed TE algorithm is presented in a sketch. The first step is to sort the demands by descending volume size. The second step partitions all the demands into groups according to the choice of the partition algorithm. The next step is to solve the TE optimization problem for each TE group, either sequentially, or in parallel with different network capacity assumptions, according to the model presented in the previous section. In real implementation and operation, traffic demands can be assigned to the groups using a simple hash function, just like a typical load balancer would do.

In this paper, we focus on the TE load balancer with effective traffic partition algorithm design and study its impacts on the overall TE performance. Our main interest is to explore if there exists a good grouping strategy, *i.e.*, a sweet spot to minimize the number of groups while maintaining the overall TE solution quality.

The intractability of the constrained unsplittable multi-commodity flow problem can be deduced from the well-known due NP-Hard problems: Bin Packing (BP) and Multiway Number Partitioning (MNP). If we treat the links in a cut of the graph as the capacitated bins, all the flows between the cut links can be treated as the items to fit into the bins. We can also try to partition these flows as evenly as possible just as what the MNP problem aims to solve.

There exists abundant effective heuristics and approximate algorithms for both BP and MNP problems [8], [9]. Given the multi-commodity nature of the TE problem, we focus on the offline algorithms that give better approximate ratios than the online heuristics.

There are three key steps shared by the representative offline approximate algorithms: sorting, partitioning, and series of TE optimization. The sorting by descending size corresponds to considering the flows of larger volumes first, which is intuitively beneficial to good TE solutions. The motivation for grouping items is to reduce the number of items and the number of different item sizes, which leads to less number of constraints and potentially good approximate ratios.

In this study, we designed three partitioning algorithms. We assume the number of partitions is $k > 0$.

1) **Linear Partition.** The sorted demands are evenly divided into k groups, each of which has at most $\frac{|D|}{k}$ items.
2) **Geometric Partition.** It proceeds in dividing the sorted demands into k groups such that the i^{th} group is assigned with the demands whose bandwidth requirements fall into the range $[B * |D|/2^{i+1}, B * |D|/2^{i})]$, where B is the minimum bandwidth requirement presented in D.
3) **k-way Partition.** All the demands are assigned to k groups so that the sum of the bandwidth requirements of all demands in each group is as even as possible. Here we use the efficient exact Karmarkar-Karp algorithm for the MNP problem [9].

After the groups are formed, the load balanced TE optimization shown in Figure 2 can be conducted in a parallel or even an asynchronous fashion, in which a TE solver can solve a much smaller TE optimization problem at different time scales with dynamic traffic arrival patterns. However, since all the servers work on the same network topology, there is an extra challenge to provide a differentiated capacitated network model to different TE servers that take the TE interdependency between the different traffic groups into consideration. On the other hand, a sequential of k optimization problems can be solved in a centralized fashion.

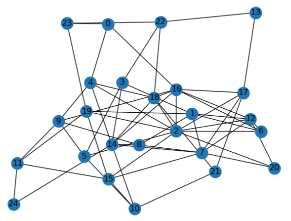
IV. Experiments and Evaluation

We evaluate the performance of the TE load balancer with different partitioning algorithms on a couple of meshed topologies and different traffic loads. We implement an ILP optimization solver for our TE model using the powerful Google OR-Tools in Python with the SCIP MIP optimizer [10]. The solver is implemented in Python 3.8 and all the experiments were conducted in Longleaf cluster, a Linux based computing cluster at the University of North Carolina at Chapel Hill. The servers in the cluster are equipped with $2.3 - 2.5GHz$ Intel processors, $24.75 - 30M$ cache, and $256 - 754GB$ RAM.

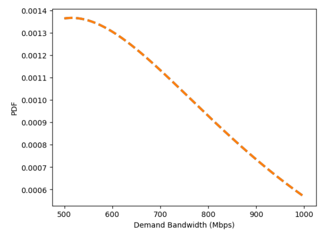
A. Network Topology and Traffic Matrix

We use the Erdős–Rényi model with edge probability $p = 0.2$ to generate random graphs in our performance study that mimic the scale of the well-known Internet backbone network topologies [6]. We made sure the generated graph is at least 2-connected. Figure 3a shows such a random topology of 25 nodes used in the study. To further emulate an operational network, we randomly assign the available bandwidth capacity for each link from the range $[5000Mbps, 10000Mbps]$. Each link is also assigned a random latency from the range $[10ms, 25ms]$.

As we discussed earlier, we use the LogNormal distribution to generate the demanded bandwidth for the demands in the traffic matrix. Specifically, we used the parameters $\mu = 750Mbps$ and $\delta = 0.5$. Figure 3b shows the PDF of the distribution. Each demand is also assigned a random latency requirement.



(a) Topology (N=25, p=0.2)



(b) PDF of the Traffic Demands

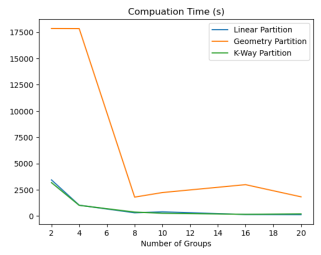
Fig. 3: Network Topology and Traffic Matrix

B. Load balancing

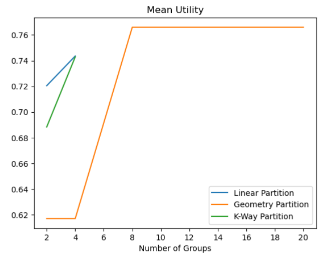
We already showed the optimization solutions in Figure 1 in the previous section. Figure 4 shows the performance for the high load of 200 demands under different partitioning algorithms with the same load balancing objective. We note, after each TE optimization step, the network state is updated before the next step in the optimization sequence until the final group is finished or no optimization solution is found. The partitioning group size ranges from 2 to 20.

Among the three partitioning algorithms, *Geometric Partition* outperforms the others significantly. It is the only one that can find the optimal solutions for all the group numbers. The other two schemes fail to find feasible solutions after the number of groups is greater than 4. Secondly, for *Geometric Partition*, eight appears to be the sweet spot group number. The computational time sharply dropped and then flattened out with eight groups. The computation time from the other two schemes is lower mainly because they failed to find a feasible solution for the last few groups and stopped early. In terms of utilization performance, the mean utilization increases with more groups as expected. However, it maxes out at eight groups. Compared to the global optimal solution, the increase is mild. At the same time, the 90% utility remains close to 100%. The utilization standard deviation dropped some and maxed out at eight groups too.

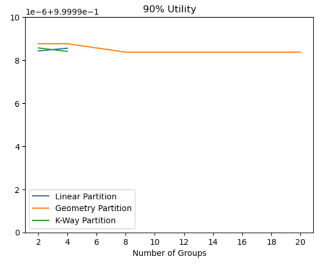
To further illustrate the performance of these algorithms, we show the results with 100 demands in Figure 5. All the 100 demands are drawn from the same traffic volume distribution. With the reduced traffic load, the *k-way Partition* also reaches the feasible solution while the linear partitioning stops after 10 groups. Again eight is the best group number for both feasible algorithms. The *k-way partition* outperforms the *Geometric Partition* marginally in both computational time and utilization metrics.



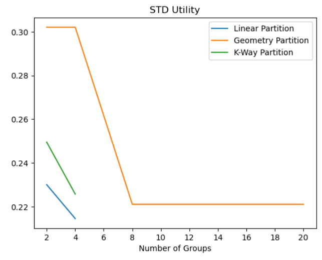
(a) Computation Time



(b) Mean Utility

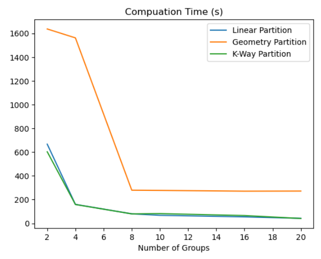


(c) 90 Percentile Utility

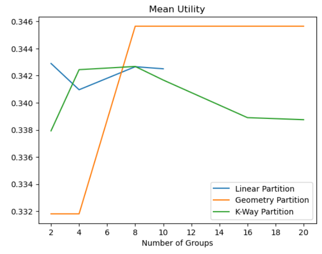


(d) Utility Standard Deviation

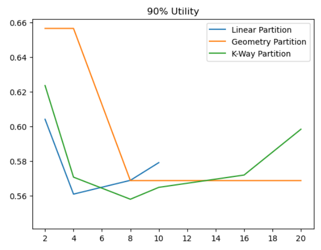
Fig. 4: Load Balancing Heuristic Solution (N= 25, 200 Demands)



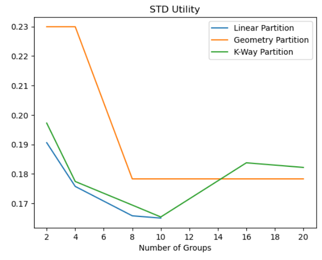
(a) Computation Time



(b) Mean Utility



(c) 90 Percentile Utility



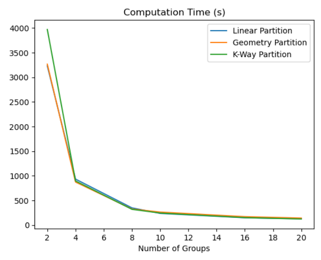
(d) Utility Standard Deviation

Fig. 5: Load Balancing Heuristic Solution (N= 25, 100 Demands)

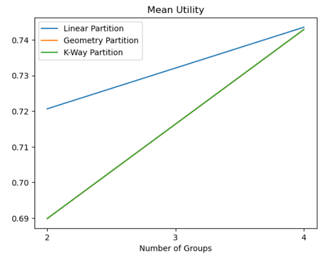
C. Minimizing Cost

Minimum cost is one of the fundamental objective functions for the general multi-commodity flow problem. In Section II, we explained several use cases with a minimum cost objective function in operational networks. In Figure 6, we show the numerical results with the Expression (8) as the objective function. Due to the page limit, we only show the results with 200 demands in the 25-node topology.

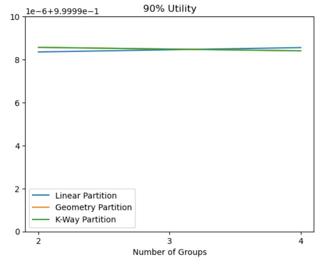
Compared with the model using the load balancing objective function, the minimum cost model incurs much less computation time for the optimal solution and therefore the partitioning based heuristics. However, when the number of groups exceeds four, all three partitioning schemes failed to find a feasible solution for the last one or two groups. This result with the high traffic load is not completely surprising as the demands are not as spreading as the load balancing model. As a result, we can see that the mean utilization is higher and there are more saturated links. Among the three partitioning algorithms, the *k-way Partition* and *Geometric Partition* perform the same while the *Linear Partition* causes higher utilization and higher cost. For more groups, since only the last one or a few groups fail to obtain a feasible solution, the partial solution is still very useful to satisfy the majority of the traffic demands.



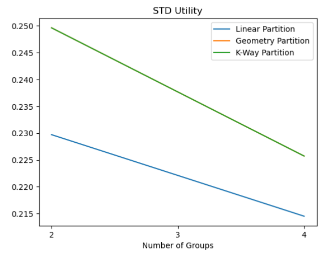
(a) Computation Time



(b) Mean Utility



(c) 90 Percentile Utility



(d) Utility Standard Deviation

Fig. 6: Minimum Cost Solution (N= 25, 200 Demands)

V. Related Work

There are two representative TE implementations in traditional networks operated under the distributed control plane and protocols: IP network TE and MPLS TE. The IP network TE manipulates the IP link weights to influence the results of the routing protocols, like the IGP protocols like OSPF or IS-IS [11]. The MPLS network enables explicit TE by establishing end-to-end rate-limited tunnels through label switching protocols [12]. The challenges include the far-from-optimal network utilization and slow routing converges in IP networks and the high cost and operational complexity in MPLS networks [1]. They rely on the constrained shortest path (CSP) algorithms to find the routing solution [13]. CSP optimization problem itself is NP-complete and leads the development of several heuristics based on shortest path or k-shortest path algorithms [14].

The logically centralized controller and separation of control plane and forwarding plane in SDN make the global TE optimization possible for an entire traffic matrix. The TE solver has become a core function in large private Cloud SD-WAN (Software Defined WAN) control software [2], [15]. As the traffic matrix and network states may be highly dynamic, these systems run TE re-optimization periodically, at a frequency ranging from three minutes to fifteen minutes.

Computational complexity and operational complexity are the critical factors behind the TE solutions. The fundamental optimization problem behind TE is the multi-commodity flow problem. It becomes a popular choice to use a path-based formulation rather than the link based formulation to reduce the number of variables in the model. These solutions pre-compute a set of paths (tunnels) between every pair of source-destination nodes and allow traffic splitting among paths [16], [17]. Apparently, the impact of path computation on the TE performance is profound but can only be evaluated experimentally.

To further reduce the computational complexity, one common option is to scale down topology size either by contracting the topology into multiple abstraction levels [4] or partitioning the topology into multiple segments [3]. TE solutions have also been extended to include the network availability to make sure enough network capacity is reserved to achieve congestion free under network failures [18]. More details on the state-of-art TE sysems can be found in a recent comprehensive review paper [5].

Most recent TE solutions focus on the layer-3 networks with a goal to maximize the splittable traffic flows under the link capacity and TM constraints. For commercial WAN ISP and the research and educational (R&E) networks, provisioning on-demand QoS-guaranteed layer-2 or MPLS connections is a important service. Unfortunately this would add the integer constraints on the unsplittable flows, which leads to an even harder integer linear programming (ILP) problem formulation. However, such kinds of demands are not as dynamic so a less frequent optimization cycle is needed.

Depending on the types of the objective functions and constraints, there are many variants with vastly different computational complexities. The optimization objective functions could be minimizing the total cost, maximizing the total flows, or minimizing the network utilizations, which may have different performance implications on the TE metrics [19].

VI. Conclusions

This paper presents the algorithmic design and performance evaluation of a scalable traffic engineering load balancing solution for the software defined WAN. We model the TE problem with a MIP formulation under different objective functions and both bandwidth and latency constraints on the demands. We studied the computational time and the network utilization distribution performance, including the mean, 90%, and standard deviation, of the global optimal and our proposed traffic partition based solutions. The results show that there exists a viable traffic partition strategy that achieves a good tradeoff between the TE performance and computational complexity under realistic backbone network scale and traffic demand distribution.

Compared to the existing path and topology slicing based solutions, our proposed traffic partition scheme not only provides good scalable TE solution quality, but also avoids the extra complexities in managing the abstracted topology and routing operations. For future work, we will extend the performance study to networks of larger size and different topological characteristics. We will further explore more effective parallelization methods for TE optimization.

Acknowledgments

This work is funded by US NSF award OAC-2029278.

References cited

[1] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in sdn-openflow networks," *Comput. Netw.*, vol. 71, oct 2014.

[2] *B4 and after: Managing Hierarchy, Partitioning, and Asymmetry for Availability and Scale in Google's Software-Defined WAN.* ACM, 2018.

[3] U. Krishnaswamy, R. Singh, N. Bjørner, and H. Raj, "Decentralized cloud wide-area network traffic engineering with BLASTSHIELD," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22).* Renton, WA: USENIX Association, Apr. 2022.

[4] F. Abuzaid, S. Kandula, B. Arzani, I. Menache, M. Zaharia, and P. Bailis, "Contracting wide-area network topologies to solve flow problems quickly," in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21).* USENIX Association, Apr. 2021.

[5] R. Singh, N. Bjørner, and U. Krishnaswamy, "Traffic engineering: from isp to cloud wide area networks," in *Proceedings of the Symposium on SDN Research*, 2022, pp. 50–58.

[6] P. Kumar, C. Yu, Y. Yuan, N. Foster, R. Kleinberg, and R. Soulé, "Yates: Rapid prototyping for traffic engineering systems," in *Proceedings of the Symposium on SDN Research*, ser. SOSR '18. New York, NY, USA: Association for Computing Machinery, 2018.

[7] M. Alasmar, R. Clegg, N. Zakhleniuk, and G. Parisis, "Internet traffic volumes are not gaussian—they are log-normal: An 18-year longitudinal study with implications for modelling and prediction," *IEEE/ACM Transactions on Networking*, vol. 29, no. 3, 2021.

[8] N. Karmarkar and R. M. Karp, "An efficient approximation scheme for the one-dimensional bin-packing problem," in *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, 1982.

[9] N. Karmarker and R. M. Karp, "The differencing method of set partitioning," EECS Department, University of California, Berkeley, Tech. Rep. UCB/CSD-83-113, 1983.

[10] "Google or-tools," https://developers.google.com/optimization.

[11] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional ip routing protocols," *IEEE Communications Magazine*, vol. 40, no. 10, 2002.

[12] G. Swallow, "Mpls advantages for traffic engineering," *IEEE Communications Magazine*, vol. 37, no. 12, 1999.

[13] S. Balon, J. Lepropre, O. Delcourt, F. Skivee, and G. Leduc, "Traffic engineering an operational network with the totem toolbox," *IEEE Transactions on Network and Service Management*, vol. 4, no. 1, 2007.

[14] J. W. Guck, A. Van Bemten, M. Reisslein, and W. Kellerer, "Unicast qos routing algorithms for sdn: A comprehensive survey and performance evaluation," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 1, 2018.

[15] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, aug 2013.

[16] S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic engineering in software defined networks," in *Proceedings IEEE INFOCOM*, 2013.

[17] E. Danna, S. Mandal, and A. Singh, "A practical algorithm for balancing the max-min fairness and throughput objectives in traffic engineering," in *Proceedings IEEE INFOCOM.* IEEE, 2012.

[18] J. Bogle, N. Bhatia, M. Ghobadi, I. Menache, N. Bjørner, A. Valadarsky, and M. Schapira, "Teavar: striking the right utilization-availability balance in wan traffic engineering," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019, pp. 29–43.

[19] F. Boavida, T. Plagemann, B. Stiller, C. Westphal, and E. Monteiro, Eds., *How Well Do Traffic Engineering Objective Functions Meet TE Requirements?* Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.