Node-Constrained Traffic Engineering: Theory and Applications

George Trimponias⬢, Yan Xiao, Xiaorui Wu, Hong Xu⬢, *Member, IEEE, ACM,* and Yanhui Geng⬢

Abstract— **Traffic engineering (TE) is a fundamental task in networking. Conventionally, traffic can take any path connecting the source and destination. Emerging technologies such as segment routing, however, use logical paths that are composed of shortest paths going through a predetermined set of middlepoints in order to reduce the flow table overhead of TE implementation. Inspired by this, in this paper, we introduce the problem of node-constrained TE, where the traffic must go through a set of middlepoints, and study its theoretical fundamentals. We show that the general node-constrained TE that allows the traffic to take any path going through one or more middlepoints is NP-hard for directed graphs but strongly polynomial for undirected graphs, unveiling a profound dichotomy between the two cases. We also investigate a variant of node-constrained TE that uses only shortest paths between middlepoints, and prove that the problem can now be solved in weakly polynomial time for a fixed number of middlepoints, which explains why existing work focuses on this variant. Yet, if we constrain the end-to-end paths to be acyclic, the problem can become NP-hard. An important application of our work concerns flow centrality, for which we are able to derive complexity results. Furthermore, we investigate the middlepoint selection problem in general node-constrained TE. We introduce and study group flow centrality as a solution concept, and show that it is monotone but not submodular. Our work provides a thorough theoretical treatment of node-constrained TE and sheds light on the development of the emerging node-constrained TE in practice.**

Index Terms— **Traffic engineering, node-constrained, segment routing, flow centrality, group maximum flow.**

I. INTRODUCTION

TRAFFIC engineering (TE) is an important task for network operators to improve network efficiency and application performance. TE is exercised in a wide range of networks, from carrier networks [19], [26] to data center backbones [28], [29]. Increasingly, TE is implemented using SDN (Software Defined Networking) given its flexibility.

G. Trimponias is with Huawei Noah's Ark Lab, Hong Kong (e-mail: g.trimponias@huawei.com).

Y. Xiao, X. Wu, and H. Xu are with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: yanxiao6-c@my.cityu.edu.hk; xiaorui.wu@my.cityu.edu.hk; henry.xu@cityu.edu.hk).

Y. Geng is with the Huawei Montreal Research Center, Montreal, QC H3N 1X9, Canada (e-mail: geng.yanhui@huawei.com).

Notable examples include Google's B4 [29] and Microsoft's SWAN [28]. Implementing TE in the data plane requires a large number of flow table entries on switches. This is because each switch on the path needs to have an entry per demand, i.e. ingress-egress switch pair, to forward its traffic to the next hop, and for a large-scale network there can be many demands [28], [29].

Segment routing [15]–[17] is a recently proposed routing architecture to tackle this challenge. Its key idea is to perform routing based on a sequence of logical segments formed by a set of *middlepoints*[1] between the ingress and egress nodes. A segment is the logical pipe between two middlepoints that may include multiple physical paths spanning multiple hops, and static hashing is used to load balance traffic among these paths. To simplify, usually only shortest paths are used between two middlepoints. Now with segment routing, instead of end-to-end paths, intermediate switches only need to know how to reach middlepoints in order to forward packets. This can greatly reduce the overhead and cost of TE [4], [26].

TE with segment routing is different from traditional TE, where the traffic from a source to a destination can use any path. This motivates us to introduce the class of *node-constrained* TE, which includes any TE variant where the traffic is constrained to go through one or more predetermined middlepoints. Segment routing corresponds to a specific variant of node-constrained TE, which only uses shortest paths between the middlepoints.

The general node-constrained routing is important for TE. First, like segment routing, the use of middlepoints saves precious flow table resources and reduces the overhead of implementing TE with finer granularity flow control. Second, the use of shortest paths in segment routing may limit the TE performance and robustness. Some shortest paths may involve the same link which degrades the throughput one can use effectively. Perhaps more importantly, operators prefer edge-disjoint paths over shortest paths for better diversity and robustness in cases of link failures [36], where failover can be done by routing through at least the remaining paths. Last but not least, the broader and fundamental node-constrained TE problem has not received much attention in the networking community, despite its significant application potential.

Inspired by the above, in this work we investigate the theoretical fundamentals of node-constrained TE. We consider two common types of TE introduced in §II depending on the

---

[1]This term is in accordance with the prior literature on segment routing.

objective: TE_{MF} maximizes the total throughput based on multi-commodity flow, while TE_{LU} minimizes the maximum link utilization. The two types are closely related.

Our analysis is organized in three parts. We start in §III with the most general node-constrained TE problem, where traffic can take any path as long as that path goes through a set of middlepoints. For directed graphs, we prove that the decision version of TE_{MF} is NP-hard, even with just a single middlepoint. Due to the connection between the decision version of maximum flow and TE_{LU}, this implies that TE_{LU} is NP-hard too. For undirected graphs, we show that TE is strongly polynomial, since it can be equivalently written as a special linear program via a polynomial transformation. Therefore, we establish that node-constrained TE is NP-hard and difficult to solve optimally in general, as most TE problems use directed graphs to model bidirectional links and traffic.

Given the hardness results, we next investigate in §IV a variant of node-constrained TE with shortest paths, where the traffic uses only shortest paths between any two middlepoints. We wish to see if this variant, inspired by segment routing, makes the TE problem easier to solve. We prove that both TE_{MF} and TE_{LU} can now be solved in weakly polynomial time by transforming them into linear programs, when the number of middlepoints per path is fixed. Our results thus provide a theoretical foundation for existing work that focuses on shortest path based segment routing [4], [26] and not the more general variant. We further note that this variant may end up with end-to-end paths that contain cycles, since the various segments may repeat the same edge. Cyclic paths are clearly bad for TE as the precious WAN bandwidth is wasted sending traffic back and forth. For this reason, we study a different variant that requires acyclic end-to-end paths, which is a specific case of node-constrained TE with shortest paths. We show that imposing this constraint generally renders TE NP-hard again.

Lastly, we study another fundamental and practical question in node-constrained TE: how to select the middlepoints that yield good TE performance? We investigate *flow centrality* as a potential solution approach to this problem in §V. Flow centrality, first introduced in 1991 [20], determines how important a node is in terms of the percentage of the maximum flow that can go through that node over all possible demands. It serves as a natural criterion for middlepoint selection in the general node-constrained TE, i.e. we can select as middlepoints the top-k nodes with the highest flow centrality.

Our analysis implies that the flow centrality is NP-hard to compute in directed graphs unlike other common centrality concepts (§VI), but strongly polynomial in undirected graphs. Furthermore, since flow centrality only concerns individual nodes, we propose *group flow centrality*, which generalizes flow centrality to a group of nodes in order to better solve the middlepoint selection problem. We introduce the related concept of N-group maximum flow, which corresponds to the problem of determining a set of middlepoints that maximizes the amount of flow that can go through any node in the set. We show it is NP-hard for directed graphs; furthermore, unlike other common group graph centralities it is monotone but

not submodular, which imples that the standard greedy algorithm [39] with $(1 - \frac{1}{e})$-approximation ratio is not applicable.

We make several contributions in this paper.

- We provide the first systematic study of node-constrained TE, which lays down the groundwork for understanding its theoretical fundamentals. Our analysis shows that node-constrained TE is generally NP-hard hard for directed graphs, except for the variant that only uses shortest paths between middlepoints. Our study further touches on the middlepoint problem in node-constrained TE. We study flow centrality and propose group flow centrality to select the best set of N nodes that maximizes the total flow, and analyze the computational complexity.

- Our theoretical results shed light on the development of the emerging node-constrained TE in practice. Our hardness results indicate that efficient heuristics or approximation algorithms are in urgent need in several cases, including general node-constrained TE problems with directed graphs which most TE problems use, and variants of node-constrained TE with acyclic paths. Middlepoint selection is another promising area for future work. More study is needed to make the graph theoretical approach feasible, for both flow centrality for individual nodes and group flow centrality which has not been well explored.

- Finally, some of our results are interesting in their own right in the corresponding theoretical contexts. For example, in §III we unveil a dichotomy between the directed and undirected cases in terms of the complexity of node-constrained TE. A similar dichotomy is found in §V, where we prove that flow centrality, a previously introduced but little understood graph centrality concept, is NP-hard to compute in directed graphs but strongly polynomial in undirected graphs.

II. BACKGROUND ON TRAFFIC ENGINEERING

We first introduce some background on traffic engineering (TE) in this section. In our work, we focus on two common types of TE depending on the objective criterion. The first maximizes the total throughput subject to the capacity and maximum demand constraints. Since it can be formulated as a maximum flow problem, we call it TE_{MF}. The second type minimizes the maximum link utilization, which acts as the system bottleneck. For this reason, we call it TE_{LU}.

The rest of this section is organized as follows. We introduce some preliminary notions and concepts in §II-A. We then present TE_{MF} in §II-B and TE_{LU} in §II-C. Lastly we show an interesting connection between the decision version of TE_{MF} and the optimal solution to TE_{LU} in §II-D.

A. Preliminaries

Assume a directed graph $G = (V, E)$, where V is the set of nodes and E the set of directed edges. Given a node $v \in V$, v^+ denotes the set of outgoing edges of node v, i.e., the subset of edges in E of the form (v, u), $u \in V$. Similarly, the set v^- denotes the set of incoming edges of v of the form (u, v), $u \in N$. The out-degree of v is defined as the cardinality $|v^+|$, whereas the in-degree is defined as the cardinality $|v^-|$.

A *flow network* $G = (V, E, c)$ is defined as a directed graph $G = (V, E)$, together with a non-negative function $c : V \times V \rightarrow \mathbb{R}_{\geq 0}$ that assigns to each edge $e \in E$ a non-negative capacity $c(e)$. If $(u, v) \notin E$, then we define $c(u, v) = 0$.

A *walk* in a directed graph is an alternating sequence of vertices and edges, $v_0, e_0, v_1, \ldots, v_{k-1}, e_{k-1}, v_k$, which begins and ends with vertices and has the property that each e_i is an edge from v_i to v_{i+1}. A *path* is a walk where all edges are distinct. A *simple path* is a path where all vertices are distinct. The term $u - v$ path (resp., simple path) refers to any valid path (resp., simple path) from u to v.

In flow networks, we usually distinguish between single-commodity and multi-commodity flows. For single-commodity flow problems, we consider a single commodity[2] that consists of a source $s \in V$ and a sink $t \in V$, where $s \neq t$. For multi-commodity flows, we assume L commodities of the form (s_i, t_i), where $s_i, t_i \in V$, $s_i \neq t_i$. Each commodity i is associated with a non-negative demand $D_i \geq 0$. For convenience, we also use the notation $\boldsymbol{s} = (s_1, \ldots, s_L)$ and $\boldsymbol{t} = (t_1, \ldots, t_L)$, and write $(\boldsymbol{s}, \boldsymbol{t})$ to denote the corresponding multi-commodity network.

B. TE Type 1: TE_{MF}

Let $\mathcal{P}_i$ be the set of all $s_i - t_i$ paths, and $\mathcal{P}_{i,e}$ the set of all $s_i - t_i$ paths that go through edge e. Then the maximum multi-commodity flow program can be expressed via the following path-based formulation:

$$\text{maximize} \quad \nu = \sum_{i=1}^{L} \sum_{p \in \mathcal{P}_i} f_i(p) \tag{1}$$

$$\text{subject to} \quad \sum_{i=1}^{L} \sum_{p \in \mathcal{P}_{i,e}} f_i(p) \leq c(e), \quad \forall e \in E \tag{2}$$

$$\sum_{p \in \mathcal{P}_i} f_i(p) \leq D_i \tag{3}$$

$$f_i(p) \geq 0, \quad \forall i \in \{1, \ldots, L\}, \forall p \in \mathcal{P}_i \tag{4}$$

In the above formulation, we divide the total flow into L subflows, one per commodity. The subflow f_i along path $p \in \mathcal{P}_i$ for commodity i is $f_i(p)$. Constraint (2) is a capacity constraint that the sum of all subflows on any edge cannot exceed the edge capacity. Constraint (3) describes the maximum demand D_i for commodity i[3]. Finally, constraint (4) imposes that each subflow should be non-negative. For any valid flow f, the value of a flow $\nu(f)$ is defined as the total sum of units that all subflows f_i send. The value of the maximum flow is denoted as ν_{max}. TE_{MF} is mostly used in data center backbone WANs [28], [29], where traffic is elastic, the operator controls not only the links but also the demands of applications, and the main objective is to fully utilize the expensive WAN links.

Note that even though the single-commodity maximum flow accepts various combinatorial algorithms [1], e.g., Ford-

Fulkerson or Edmonds-Karp, there is to date no combinatorial algorithm for the maximum multi-commodity flow even though the problem is known to be strongly polynomial [44]. Furthermore, even though a single-commodity network with integral capacities always accepts an integral maximum flow, this is not always the case with multi-commodity networks; in fact, the decision problem of integral multi-commodity flow is NP-complete even if the number of commodities is two, for both directed and undirected networks [11].

C. TE Type 2: TE_{LU}

TE_{LU} is mostly used in carrier networks [19], [26], where traffic demands are exogenous and inelastic, and the main objective thus is to control the congestion or link utilization in order to ensure the smooth operation of the network. The general form for this type of TE is:

$$\text{minimize} \quad \theta \tag{5}$$

$$\text{subject to} \quad \sum_{i=1}^{L} \sum_{p \in \mathcal{P}_{i,e}} f_i(p) \leq \theta \cdot c(e), \quad \forall e \in E \tag{6}$$

$$\sum_{p \in \mathcal{P}_i} f_i(p) \geq D_i, \quad \forall i \in \{1, \ldots, L\} \tag{7}$$

$$f_i(p) \geq 0, \quad \forall i \in \{1, \ldots, L\}, \forall p \in \mathcal{P}_i \tag{8}$$

The variable θ in objective (5) refers to the maximum link utilization, which must be minimized. Constraint (6) ensures that θ is at least as large as the maximum link utilization; constraint (7) ensures that each demand is satisfied; and the last constraint (8) is similar to TE_{MF} in §II-B.

D. Relationship Between TE_{MF} and TE_{LU}

A natural question is whether the two types of TE are related. To answer this question, we first introduce the *decision version* of the maximum multi-commodity flow problem.

Definition 1: [Decision version of maximum flow (DMF)] Given a flow network $G = (V, E, c)$ with a set of L commodities $(\boldsymbol{s}, \boldsymbol{t})$, each associated with a non-negative maximum demand $D_i \geq 0$, decide whether the maximum multi-commodity flow has a value of at least $\sum_{i=1}^{L} D_i$.

Note that if the answer to the decision problem DMF is a "yes", then by constraint (3) the maximum flow has to be exactly equal to $\sum_{i=1}^{L} D_i$. If the answer is no, then the maximum flow is strictly less than $\sum_{i=1}^{L} D_i$. Our next result establishes the relationship between the two types of TE:

Lemma 1: DMF accepts a "yes" answer, if and only if the system (5)–(8) for TE_{LU} accepts a solution $\theta^* \leq 1$.

Proof: Assume that DMF accepts a "yes" answer. Then there is a flow that respects constraints (2)–(4). That flow will then trivially satisfy constraints (2)–(4) with $\theta = 1$. Since the objective criterion of TE_{LU} minimizes over θ, the optimal solution to the TE program (5)–(8) will accept an optimal solution $\theta^* \leq 1$. For the reverse direction, assume that the system (5)–(8) accepts a solution $\theta^* \leq 1$. Then constraint (6) implies that the capacity constraints are satisfied for each edge,

---

[2]When it is clear from the context, we use the terms *commodities*, *demands*, and *flows* interchangeably.

[3]When the maximum demand D_i is infinite, the corresponding demand constraint (3) is trivially satisfied and can thus be removed.

thus the corresponding flow is a valid flow for system (1)–(4) with value $\sum_{i=1}^{L} D_i$. The maximum flow has then trivially a value of at least $\sum_{i=1}^{L} D_i$. $\qquad\square$

Lemma 1 shows that solving TE_{LU} immediately generates a "yes" or "no" answer to the DMF. Thus, the TE naturally encompasses the general DMF problem of Definition 1. This also suggests that hardness results on the DMF (Proposition 1) immediately imply hardness for TE_{LU}.

We conclude this section with two observations. First, even though we assumed a directed network throughout this section, it is possible to extend the definitions to undirected graphs as well[4]. The main difference is that an undirected edge is associated with a capacity, and flow can travel in both directions of a link, under the constraint that the sum of the flow value in the two edge directions does not exceed the capacity. Second, it is simple to reduce edge-constrained TE to node-constrained TE in both directed and undirected graphs. Indeed, we can replace any (directed or undirected) edge (u, v) by the two consecutive edges (u, z) and (z, v) by introducing a new node z. Then the traffic constrained to go through edge e can be equivalently characterized as the traffic going through node z. Node-constrained TE is thus at least as hard as edge-constrained TE, which is why we focus on the former.

III. GENERAL NODE-CONSTRAINED TRAFFIC ENGINEERING

In this section, we study the general (unrestricted) node-constrained TE problem, where traffic can take any path as long as that path goes through a set of middlepoints. We focus on the simplest setting where a path goes through a specific node w, in order to establish the hardness results. For directed graphs, we show in §III-A that the decision version of the maximum w-flow is NP-hard, which implies that TE_{LU} is NP-hard as well. On the other hand, we show in §III-B that in undirected graphs the problem is strongly polynomial after equivalently rewriting it as a special linear program. At the end of the section, we show that our results also extend to the general case where traffic goes through *at least* one node from the $k > 1$ middlepoints $w_1, \ldots, w_k$, where k is fixed and not part of the input. The practical significance of our results is that we rigorously establish that node-constrained TE is NP-hard and difficult to solve in general, as most TE problems use directed graphs to model bidirectional links and traffic.

A. The Directed Case

1) Hardness of TE_{MF}: The maximum multi-commodity flow f_{max} with value ν_{max} refers to the total flow over all possible paths that each commodity accepts. Assume instead that we focus on the maximum flow that can go through a specific network node $w \neq s, t$. Let $\mathcal{P}_i^w$ be the set of all $s_i - w - t_i$ paths (i.e. $s_i - t_i$ paths that go through w), and
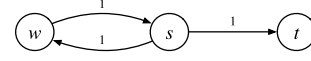
Fig. 1. A simple example that illustrates why the maximum w-flow requires paths but not simple paths.

$\mathcal{P}_{i,e}^w$ the set of all $s_i - w - t_i$ paths that also go through edge e. The path-based formulation then is:

$$\text{maximize} \quad \nu^w = \sum_{i=1}^{L} \sum_{p \in \mathcal{P}_i^w} f_i(p)$$

$$\text{subject to} \quad \sum_{i=1}^{L} \sum_{p \in \mathcal{P}_{i,e}^w} f_i(p) \le c(e), \quad \forall e \in E$$

$$\sum_{p \in \mathcal{P}_i^w} f_i(p) \le D_i$$

$$f_i(p) \ge 0, \quad \forall i \in \{1, \ldots, L\}, \ \forall p \in \mathcal{P}_i^w$$

We denote the maximum flow through any node w as the *maximum w-flow* f_{max}^w and denote its value by ν_{max}^w. Alternatively, we use the notation $s - w - t$ flow for single-commodity networks (or $\boldsymbol{s} - \boldsymbol{w} - \boldsymbol{t}$ for multi-commodity networks). Similarly, for single-commodity flows we also write $\nu_{max}^w(s, t)$ (or $\nu_{max}^w(\boldsymbol{s}, \boldsymbol{t})$ for multi-commodity networks) for the value of the maximum w-flow.

We emphasize three points. First, in the single-commodity case we always assume that $w \neq s, t$, even if not explicitly stated. Indeed, if either $w = s$ or $w = t$ then $\nu_{max} = \nu_{max}^w$. In this case, the problem is strongly polynomial and accepts combinatorial algorithms such as the Ford-Fulkerson algorithm. Second, to define the maximum w-flow we use paths, not simple paths. For traditional flow networks, the maximum flow can be defined in terms of simple paths rather than paths, since we can always remove any cycles in the path to make it simple, without affecting the maximum flow. As an example, consider the directed flow network in Figure 1 where all edges have unit capacity. The maximum $s - w - t$ flow uses the directed path $s \to w \to s \to t$ for a value of 1, and no simple path exists for a w-flow. On the other hand, the traditional maximum $s - t$ flow can use the trivial simple path $s - t$, with a value that also happens to be 1 in this example; the cycle $s \to w \to s$ is redundant. Hence, even though the maximum $s - t$ flow can be defined either in terms of paths or simple paths, the w-maximum flow is defined in terms of paths, which is why our formulations consider paths but not simple paths. This point is discussed in more detail at the end of §III-A. Third, for undirected networks we allow the flow to be sent along any directed path from the source to the destination, as long as no directed edge is repeated. For example, consider the undirected network $w - s - t$ with capacities 1 for the two edges (w, s) and (s, t). In that case, the maximum $s - w - t$ flow is 0.5, and uses the directed path $s \to w \to s \to t$. Thus, the path carrying the flow can contain the same undirected edge twice, but it has to pass this edge in different directions when going from the source to the destination.

A central result in graph theory that we will be using throughout the paper is the two node-disjoint path (2DP) problem due to Fortune, Hopcroft and Wyllie [18].

Theorem 1 (NP-hardness of 2DP [18]): Assume a directed graph $G = (V, E)$ and four distinct vertices $u_1, u_2, v_1, v_2 \in V$. It is NP-hard to decide whether there are two node-disjoint paths in G from u_1 to u_2 and from v_1 to v_2.

We now provide two lemmas. Some of the transformations involved are standard in the disjoint-path literature (e.g., [34], [43]). Nevertheless, given the central role of the two lemmas in the remainder of this paper and in order to make our work self-contained, we provide our own full proofs here.

Lemma 2: Deciding whether there exists a simple $s - w - t$ path in a directed graph $G = (V, E)$, where w, s, t are three distinct nodes in V, is NP-hard.

Proof: Finding whether there is a simple $s - t$ path going through a node w is equivalent to determining whether there exist two node-disjoint paths from s to w and from w to t (excluding of course node w). We prove that the latter problem is NP-hard by a reduction from the NP-hard 2DP problem.

Consider a directed graph $G = (V, E)$ and 4 distinct nodes $u_1, u_2, v_1, v_2 \in V$. We introduce a new node w and create the directed edges $e_1 = (u_2, w)$ and $e_2 = (w, v_1)$. We now argue that there are two node-disjoint paths, path P_1 from u_1 to u_2 and path P_2 from v_1 to v_2, if and only if there is a simple $u_1 - w - v_2$ path. First, assume the former condition is true. Then P_1 cannot go through node w via edge e_1; otherwise, that path would also have to use node v_1 after w given u_2 is the end node. Similarly, we argue that P_2 cannot go through node w via edge e_2. But then we can form a new path P' from u_1 to v_2 by concatenating path P_1, edge e_1, edge e_2, and path P_2. P' does not repeat any node since the node disjoint paths P_1 and P_2 do not contain w, hence it is a simple path. For the reverse direction, we note that if there exists a simple $u_1 - w - v_2$ path P, then P will necessarily contain edges e_1 and e_2. By removing these two edges, we get two node-disjoint paths, one from u_1 to u_2 and another from v_1 to v_2, given that P is simple. $\square$

Lemma 3: Deciding whether there exists a $s - w - t$ path in a directed graph $G = (V, E)$, where w, s, t are three distinct nodes in V, is NP-hard.

Proof: Deciding whether there is a $s - t$ path going through w is equivalent to deciding whether there are two edge-disjoint paths from s to w and from w to t. We argue that the latter problem is NP-hard by a reduction from the 2DP problem.

Indeed, consider a graph $G = (V, E)$, and three distinct nodes $s, t, w \in V$. We construct a new graph $G' = (V', E')$ from $G = (V, E)$ as follows. For each node $v \in V$ we introduce two nodes $v_{in}, v_{out} \in V'$ as well as an edge $e' = (v_{in}, v_{out}) \in E'$ connecting them. For each edge $e = (u, v) \in E$, we introduce an edge $e' = (u_{out}, v_{in}) \in E'$. The construction is illustrated in Figure 2.

We now claim that there exist two edge-disjoint paths in graph G' from s_{out} to w_{in} and from w_{out} to t_{in} (equivalently, from s_{in} to w_{in} and from w_{out} to t_{out}), if and only if there exist two node-disjoint paths in G from s to w and from w to t. First, consider two node-disjoint paths in G, namely, $s, u_1, \ldots, u_l, w$ and $w, v_1, \ldots, v_m, t$, where all intermediate
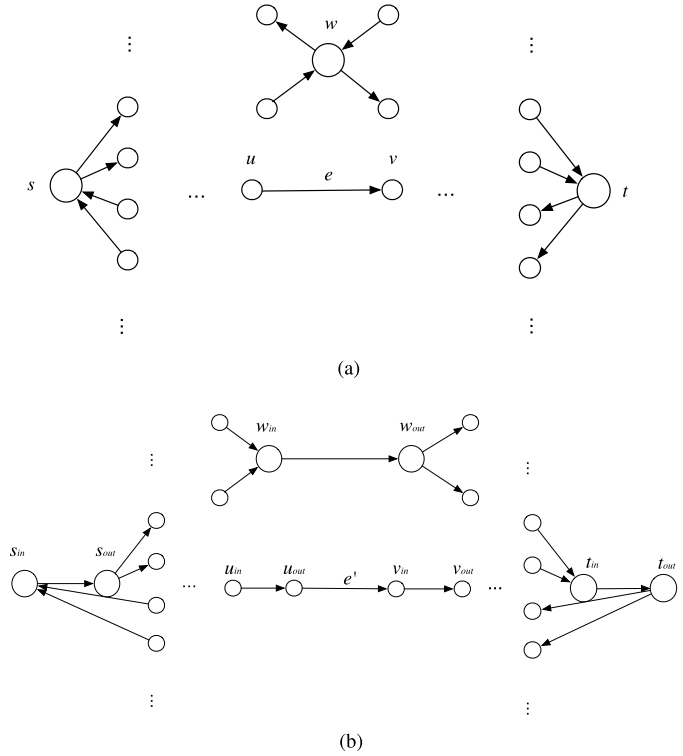

Fig. 2.   Illustration of Lemma 3. (a) G. (b) G'.

nodes u_i and v_j are distinct. It is easy to see that the paths $s_{out}, u_{1,in}, u_{1,out}, \ldots, u_{l,in}, u_{l,out}, w_{in}$ and $w_{out}, v_{1,in}, v_{1,out}, \ldots, v_{m,in}, v_{m,out}, t_{in}$ in G' are: (1) valid since they use existing edges in G', and (2) edge-disjoint since the set of nodes on the first path and the second path are disjoint.

For the reverse direction, consider two edge-disjoint paths in G' from s_{out} to w_{in} and from w_{out} to t_{in}. We then argue that these paths must have the previous form $s_{out}, u_{1,in}, u_{1,out}, \ldots, u_{l,in}, u_{l,out}, w_{in}$ and $w_{out}, v_{1,in}, v_{1,out}, \ldots, v_{m,in}, v_{m,out}, t_{in}$. The reason is that any pair of nodes (v_{in}, v_{out}) can only be reached from other nodes in V' via v_{in} and can only reach other nodes in V' via v_{out}. So, a path will necessarily consist of consecutive pairs of nodes of the form (v_{in}, v_{out}) (with the exception of the two endpoints). Furthermore, any such pair (v_{in}, v_{out}) can (1) appear at most once on either path, and (2) cannot appear on both paths. The reason is that going from v_{in} to v_{out} requires edge v_{in}, v_{out}, but the two paths are edge-disjoint. We thus conclude that $s, u_1, \ldots, u_l, w$ and $w, v_1, \ldots, v_m, t$ in G are node-disjoint paths. $\square$

We next provide definitions and results for the maximum w-flow that are reminiscent of results in traditional single-commodity maximum flow. For this purpose, we adapt some standard concepts from maximum flow theory [1]. In particular, we extend the concept of an $s - t$ *augmenting path* to an $s - w - t$ *augmenting path*, which corresponds to a directed path from s to t through middlepoint w in the residual network with forward and backward edges. One significant difference is that the *cut* is now defined as a collection of edges rather than a collection of nodes. The reason for this will become apparent shortly. We focus first on the $s - w - t$ flow in single-commodity networks.

Definition 2: A $s - w - t$ edge-cut is a subset of edges $\mathcal{C}^w \subseteq E$ such that removing the edges in $\mathcal{C}^w$ from the graph results in no $s - w - t$ paths, i.e., there are no $s - w - t$ paths in the graph $G' = (V, E - \mathcal{C}^w)$. The value $c(\mathcal{C}^w)$ of the edge-cut is defined as the sum of the capacities of all edges in $\mathcal{C}^w$.

Lemma 4: Let f^w be any $s-w-t$ flow, and $\mathcal{C}^w$ any $s-w-t$ cut. Then $\nu^w(f^w) \leq c(\mathcal{C}^w)$.

Proof: First, note that the flow f^w is the sum of individual subflows, each going through a distinct $s - w - t$ path p. Each of these individual subflows must go through at least one edge $e \in \mathcal{C}^w$, otherwise there would be a $s - w - t$ path in the graph $G' = (V, E - \mathcal{C}^w)$, which would be a contradiction. So, let $\mathcal{F}_e$ be the set of subflows that go through e. Then we have:

$$\sum_{e \in \mathcal{C}^w} \nu^w(\mathcal{F}_e) \leq \sum_{e \in \mathcal{C}^w} c(e) \Leftrightarrow$$
$$\sum_{e \in \mathcal{C}^w} \nu^w(\mathcal{F}_e) \leq c(\mathcal{C}^w) \Leftrightarrow$$
$$\nu^w(f^w) \leq c(\mathcal{C}^w) \quad (9)$$

Note that in the last inequality we use that $\sum_{e \in \mathcal{C}^w} \nu^w(\mathcal{F}_e) = \nu^w(f)$, due to the fact that the path for each individual subflow must go through at least one edge in $\mathcal{C}^w$. □

Lemma 5: Given a directed graph $G = (V, E, c)$ with integral capacities and three distinct nodes s, w, t, we can construct an integral w-flow. Furthermore, the constructed flow is positive if and only if the minimum edge-cut in G is non-empty.

Proof: Consider the variant of the well-known Ford-Fulkerson algorithm for (single-commodity) maximum flow [41], where at each round the algorithm picks an augmenting $s - w - t$ path rather than a $s - t$ path in the residual graph, provided that the selected augmenting path increases the flow through w. The augmenting $s - w - t$ path algorithm eventually terminates, since (i) the w-flow increases by at least one unit at each iteration, and (ii) the maximum possible w-flow is upper-bounded (e.g., by the sum of capacities of the outgoing edges from s). Note that the augmenting $s - w - t$ path algorithm terminates, if and only if there is an $s-w-t$ edge-cut $\mathcal{C}^w$ in the graph where each edge $e \in \mathcal{C}^w$ is saturated. The constructed flow is integral, since at each step the flow on any edge is integral. Furthermore, the flow will be zero, if and only if the minimum edge-cut is the empty set, since in that case there are no $s - w - t$ paths in the original graph G. □

We are now ready to prove that the decision version of the maximum w-flow is NP-hard.

Proposition 1: Given a multi-commodity flow network $G = (V, E, c)$ with directed edges, the decision version of maximum w-flow is NP-hard.

Proof: We show that even the single-commodity version is NP-hard. Our strategy is to reduce the $s - w - t$ path problem in Lemma 3 to the maximum w-flow problem. In this direction, we start with a directed graph $G = (V, E)$ and three distinct nodes $s, t, w \in V$. We subsequently construct in polynomial time a flow network G' from G by considering a single commodity from s to t of unit demand $D = 1$, and by associating each edge $e \in E$ with a unit capacity. Our claim

is that there is a path from s to t through w in graph G, if and only if $\nu_{max}^w(s, d) \geq 1$ in flow network G'.

First, consider a path $P = (s, e_1, \ldots, e_m, t)$, so that every edge e_i in the path appears only once and node w appears in the path. It is then possible to send one unit of flow from s to t, given the unit capacities. Thus, the maximum flow is at least 1. For the reverse direction, assume there is a maximum flow no less than 1 in G'. Since we only have one commodity and integral capacities, Lemma 5 implies that there exists an integral w-flow (for infinite demands). This flow is positive (and thus has value at least 1) because the maximum flow is no less than 1 by assumption, and thus the minimum edge-cut cannot be the empty set (recall we route flow along $s - w - t$ paths). For $D = 1$, there is thus an integral maximum w-flow of value 1. Moreover, each edge can be used at most once in that flow due to its unit capacity. Now, consider any path $P_{(s,w)}$ that carries the integral flow from s to w, and any path $P_{(w,d)}$ that carries the integral flow from w to d. Since each edge is used at most once, this means that the union $P_{(s,w)} \cup P_{(w,d)}$ (1) goes from s to d through w, and (2) visits any edge at most once; hence, it is a $s - w - t$ path. □

2) Hardness of TE_{LU}: For the flow network G with the L commodities, we define the TE_{LU} in the following manner:

$$\min \quad \theta \quad (10)$$

$$\text{subject to} \quad \sum_{i=1}^{L} \sum_{p \in \mathcal{P}_{i,e}^w} f_i(p) \leq \theta \cdot c(e), \quad \forall e \in E \quad (11)$$

$$\sum_{p \in \mathcal{P}_i^w} f_i(p) \geq D_i, \quad \forall i \in \{1, \ldots, L\} \quad (12)$$

$$f_i(p) \geq 0, \quad \forall i \in \{1, \ldots, L\}, \forall p \in \mathcal{P}_i^w \quad (13)$$

We get the following corollary:

Corollary 1: In directed graphs, it is NP-hard to solve the TE_{LU} (10)–(13).

Proof: Proposition 1 shows NP-hardness for the decision version of the maximum w-flow. We can then easily extend Lemma 1 to the case of a maximum w-flow to prove that the TE_{LU} (10)–(13) is NP-hard, since it is at least as hard as the decision version of the maximum w-flow. □

a) Simple Paths vs. Paths vs. Walks: Finally, we discuss in more detail our definition of w-flow. In principle, we can define the w-flow in terms of simple paths, paths or general paths with edge repetitions (walks). For walks, the maximum w-flow is in fact polynomial. To see why, consider the single-commodity $s - t$ maximum flow problem with middlepoint w (the multi-commodity case is similar). We first solve the multi-commodity flow problem with source-destination pairs (s, w) and (w, t), maximizing the minimum of the amount of two flows. We can subsequently get $s - w - t$ paths after decomposing the two flows into paths.

Interestingly, our definition excludes walks. The reason for this choice is twofold. First, in computer networks routing loops are considered bad and are avoided by most routing algorithms, since they can lead to redundant use of the precious bandwidth resources and, more seriously, to endless routing loops and failure of packet delivery [33]. Thus, in practical networking settings TE with simple paths is the primary

option. Second, our work is inspired to a significant degree by prior work on flow centrality [20] which explicitly defines flow in terms of simple paths. Based on Lemma 2, we can show similar to the case of paths that the decision problem of w-flow with simple paths is also NP-hard in directed graphs. This suggests there is no difference in terms of hardness if we use simple paths or paths; we select the latter because paths are more general than simple paths.

B. The Undirected Case

We demonstrate an interesting dichotomy between directed and undirected graphs. In detail, the maximum multi-commodity $s - w - t$ flow in an undirected graph can be computed in strongly polynomial time. Note the main difference between the directed and the undirected flow is that the former assumes separate capacities for each direction (u, v) and (v, u) whereas the latter assumes a single capacity for the undirected edge e, which upper bounds the *total* flow that we can send in both directions (but not in any individual direction).

Proposition 2: The maximum multi-commodity flow ν_{max}^w in any flow network $G = (V, E)$ with undirected edges, where $w \in V$, can be computed exactly in strongly polynomial time.

Proof: Assume a multi-commodity undirected graph $G = (V, E)$ with L commodities of the form (s_i, t_i). For simplicity, we assume infinite maximum demands D_i (so that the maximum demand constraints are trivially satisfied). To prove the claim, we construct a directed graph $G' = (V', E')$ from G as follows. We first replace each undirected edge $(u, v) \in E$ by two directed edges (u, v) and (v, u) edges with infinite capacities. We next introduce L new nodes $z_1, \ldots, z_L$ (one for each commodity), and for each z_i we add the two directed edges (s_i, z_i) and (t_i, z_i). Finally, we introduce a node z and L directed edges (z_i, z) from each z_i to z. Thus, we have that $V' = V \cup \{z_1, \ldots, z_L, z\}$ and $E' = E_G \cup (\cup_i \{(s_i, z_i), (t_i, z_i), (z_i, z)\})$, where E_G are the edges that we got by replacing each undirected edge in E by two directed edges. The capacities of the newly constructed edges of the form $(s_i, z_i), (t_i, z_i), (z_i, z)$ are infinite. The construction is illustrated in Figure 3.

Next, we claim that the maximum flow in G can be computed by the following arc-based linear program for G':

$$\text{maximize} \quad \mathcal{V} = \sum_{i=1}^{L} \sum_{e \in w^+} f_i(e)$$

$$\text{subject to} \quad \sum_{i=1}^{L} [f_i(u, v) + f_i(v, u)] \leq c(e), \quad \forall e = (u, v) \in E \tag{14}$$

$$\sum_{e \in u^+} f_i(e) = \sum_{e \in u^-} f_i(e), \quad \forall i,$$
$$\forall u \in V', w \neq u \neq z \tag{15}$$

$$f_i(e) \geq 0, \quad \forall e \in E', \; \forall i \in \{1, \ldots, L\} \tag{16}$$

$$f_j(s_i, z_i) = 0, \quad \forall i, j \in \{1, \ldots, L\} \text{ with } i \neq j \tag{17}$$

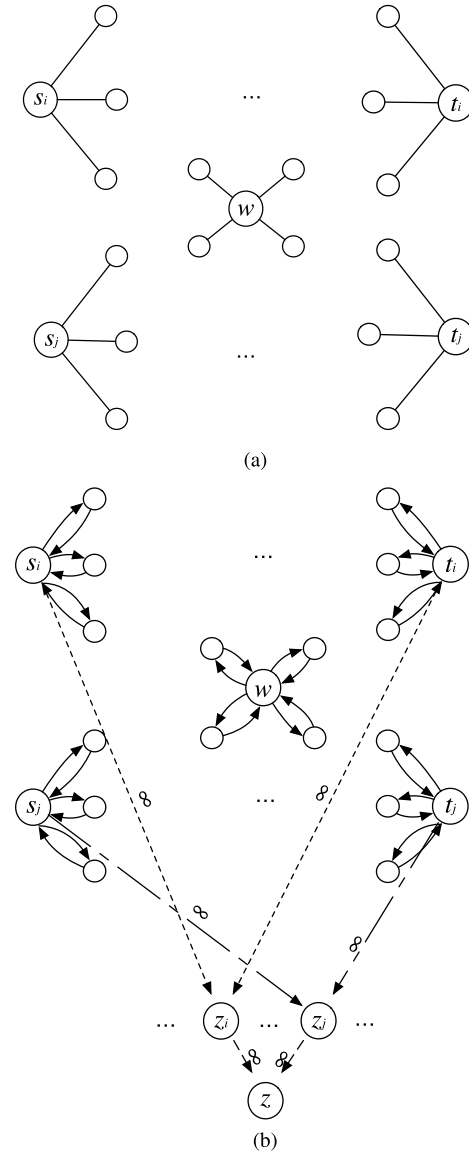$$f_i(s_i, z_i) = f_i(t_i, z_i), \quad \forall i \in \{1, \ldots, L\} \tag{18}$$

Fig. 3. Illustration of Proposition 2. (a) G. (b) G'.

The above LP computes the maximum flow $\mathcal{V}^*$ from w to z, where the total flow is composed of L separate subflows, one for each commodity. The subflow for commodity i can be sent from w to z_i through either s_i or t_i. Constraints (14), (15), (16) are the link capacity, node conservation and positive flow constraints, respectively. The link capacity constraint reflects the fact that the sum of flow units in each of the two directed edges does not exceed the capacity of the original undirected edge. Constraint (17) implies that node z_i can only receive flow from commodity i. Constraint (18) is especially important because it ensures that the subflow for commodity i sent through s_i is the same as the one sent through t_i.

Now, we establish the equivalence between the original problem and the above LP by showing that (i) $2 \cdot \nu_{max}^w \leq \mathcal{V}^*$, and (ii) $\mathcal{V}^* \leq 2 \cdot \nu_{max}^w$. We start with (i). Assume any maximum flow through w with value ν_{max}^w in G. We first construct the corresponding flow in G'. We note that the flow in G consists of L subflows, one for each commodity i, that send flow from

s_i to t_i through $s_i - w - t_i$ paths. The idea is then to reverse the direction of each subflow in the part from s_i to w_i, so that it now sends to the opposite direction. We then send $\nu(f_i)$ units of flow from s_i to z_i, $\nu(f_i)$ units of flow from t_i to z_i, and $2 \cdot \nu(f_i)$ units of flow from z_i to z. Note that that is a valid flow since it respects all constraints in the LP, and it has a value $2 \cdot (\nu(f_1) + \cdots + \nu(f_L)) = 2 \cdot \nu_{max}^w$. But then the maximum flow will be at least as large, hence $2 \cdot \nu_{max}^w \leq \mathcal{V}^*$.

For the reverse direction (ii), assume a maximum flow in G'. Then for each commodity i half units are sent from w to s_i and half from w to t_i (and subsequently z_i and z) due to constraint (18). As before, we next reverse the flow f_i in all paths from w to s_i. For each edge of G we then send on each direction as many units of flow as we send in G' (after reversing the direction from w to s_i). The key is that in graph G' the same amount of flow is sent from w to s_i and w to t_i, which implies that the constructed flow in G will respect all capacity and conservation constraints, including for node w. Note however a caveat: it is not obvious whether we can always combine directed paths from s_i to w with directed paths from w to t_i so that *all* resulting $s_i - w - t_i$ "paths" are valid paths without edge repetitions. If that is possible, then the constructed flow in G is a valid flow, since it consists of $s_i - w - t_i$ paths without edge repetitions. But we need to address the case where combining directed paths from s_i to w with directed paths from w to t_i will result in at least one non-valid $s_i - w - t_i$ path with edge repetitions.

To show why this does not negatively affect our argument, let P_1 be *any* $s_i - w$ and P_2 *any* $w - t_i$ path in G' whose concatenation is not a valid $s_i - w - t_i$ path. In this case, let $e = (u, v)$ be a common edge of P_1 and P_2 (such an edge must occur by assumption). Path P_1 can then be represented as $s_i \rightsquigarrow_{P_1} u \rightarrow v \rightsquigarrow_{P_1} w$, where the notation $x \rightsquigarrow_P y$ refers to the path segment from node x to node y along path P. Note that we may have that x coincides with y, in which case $x \rightsquigarrow_P y = x$. Similarly, path P_2 can be represented as $w \rightsquigarrow_{P_2} u \rightarrow v \rightsquigarrow_{P_2} t_i$. The trick is to consider the $s_i - w - t_i$ path $P' = s_i \rightsquigarrow_{P_1} u \rightsquigarrow_{P_2^{-1}} w \rightsquigarrow_{P_1^{-1}} v \rightsquigarrow_{P_2} t_i$, where $x \rightsquigarrow_{P^{-1}} y$ refers to the path from x to y along the reversed edges of P. We first discuss the case where P' is a valid path without edge repetitions. We then send in graph G as many units of flow along P' as we did along P_1 or P_2 in G'. The critical observation is that an undirected edge in G will receive *at most* as much flow as the sum of its two directed edges in G', since all we do is change the direction along which we send the flow and possibly remove certain path segments. The trick is illustrated in Figure 4. In the case where the resulting path P' is not valid and contains edge repetitions, we repeat the step in Figure 4 on the new $s_i - w$ and $w - t_i$ paths until they have no common edge. The process is guaranteed to stop after a finite number of iterations since after each step the total length of the new $s_i - w$ and $w - t_i$ paths decreases.

In principle, we can do this for all non-valid $s_i - w - t_i$ paths to get $s_i - w - t_i$ paths which are valid, without violating any capacity constraint. Our argument shows that there must exist a valid multi-commodity w-flow in G, which carries the same amount of flow from s_i to t_i as the flow from w to s_i in G' (note we do not actually need to explicitly construct that
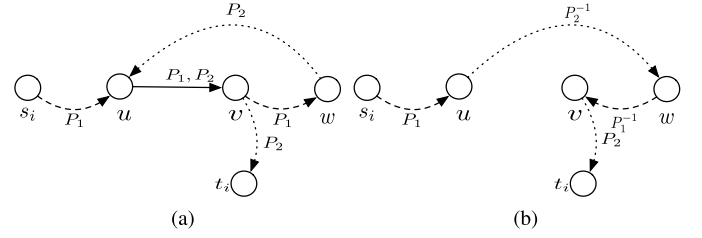

Fig. 4. Getting a valid $s_i - w - t_i$ path for undirected graph G from invalid path in directed graph G' in Proposition 2. (a) G'. (b) G.

flow in G as we just care about ν_{max}^w). Concretely, commodity i sends in G the same number of flow units as w sends to s_i (or t_i) in G'. Obviously, the value of the constructed w-flow in G is half that in G', thus for the maximum w-flow in G we must have that $\mathcal{V}^* \leq 2 \cdot \nu_{max}^w$.

Finally, given that (i) the constraints of the LP have a size that is polynomial in $|V|$ and $|E|$, and (ii) the constraint matrix of the LP only contains entries $-1, 0, 1$, we deduce that computing the maximum $s - w - t$ flow in an undirected graph can be solved in strongly polynomial time [44]. $\qquad \square$

C. Extending to Many Middlepoints

We conclude this section with a last remark. So far we have discussed node-constrained TE where the traffic has to go through a specific node w. This is mostly for analytical convenience. What about the general case where the traffic can go through *at least* one node from a set of more than one middlepoints? Obviously, for directed graphs the problem remains NP-hard. For undirected graphs, it is possible to show that when the number of nodes is fixed and not part of the input, then TE remains strongly polynomial. Indeed, we can generalize the proof of Proposition 2 by assuming k nodes $w_1, \ldots, w_k$ and constructing the same directed graph G'. We can then show similar to Proposition 2 that the maximum multi-commodity flow that goes through at least one node in the set $\{w_1, \ldots, w_k\}$ is equal to half the maximum multi-commodity flow from the sources $w_i, 1 \leq i \leq k$, to their common destination z in the constructed directed graph G'. We thus rigorously establish that node-constrained TE is NP-hard and difficult to solve in general, as most TE problems use directed graphs to model bidirectional links and traffic.

IV. NODE-CONSTRAINED TRAFFIC ENGINEERING WITH SHORTEST PATHS

The previous section shows that general node-constrained TE is NP-hard in directed graphs. We now consider variants of node-constrained TE where only shortest paths between two middlepoints are used in TE, and investigate if these variants are easier to solve. In §IV-A we formulate node-constrained TE with shortest paths using a fixed number of middlepoints, and show that it is weakly polynomial. In this sense, our results provide for the first time a theoretical foundation for existing work that focuses on shortest path based segment

routing [4], [26].[5] Given that TE with shortest paths may result in end-to-end paths that contain cycles, in §IV-B we study a more specific variant of acyclic node-constrained TE with shortest paths, and show that it is generally NP-hard. Note that in this section we focus on directed graphs, since segment routing typically considers directed graphs to model network traffic.

A. Variant With Shortest Paths

Assume K middlepoints in total with a specific ordering, where $1 \leq K \leq |V|$. We assume that each end-to-end path can use up to $M \leq K$ of these middlepoints respecting the ordering as the K input middlepoints. Note that prior works typically assume small values of M; for instance M can be as small as 1, in which case each end to end path consists of 2 segments [4]. For a segment $s \in S$ between an ingress node and a middlepoint, two middlepoints, or a middlepoint and an egress node, there are multiple paths in general. We assume, for simplicity, that routing is done by ECMP over all shortest paths of a segment. ECMP routes a flow based on static hashing of the five tuples in the packet header, and in general can distribute traffic evenly when the number of flows is large. This is consistent with prior work [4]. We use T_i to denote the complete set of logical tunnels formed by segments in S that can be used for commodity i, with up to M middlepoints. A tunnel involves only ingress/egress switch, and the intermediate middlepoints. This can be constructed offline efficiently.

Let $G_{t,s}$ denote if a tunnel t uses segment s or not, and $I_{p,e}$ denote if path p uses link e or not. Furthermore, let $\hat{P}_s$ be the set of all shortest paths for segment s, and $f_i(t)$ represent the flow in tunnel t for commodity i. The split ratio $x_{i,t}$ for i on tunnel t is defined as the ratio $x_{i,t} = \frac{f_i(t)}{\sum_{t \in T_i} f_i(t)}$. The node-constrained TE_{LU} problem with shortest paths can be formulated similar to §II-C, where the set of paths $\mathcal{P}_i$ for commodity i is now replaced by the set of logical tunnels T_i:

$$\min \quad \theta \tag{19}$$

$$\text{s.t.} \quad \sum_{i=1}^{L} \sum_{t \in T_i} \sum_{s \in S_t} \sum_{p \in \hat{P}_s} f_i(t) \frac{I_{p,e}}{|\hat{P}_s|} \leq \theta \cdot c(e),$$
$$\forall e \in E, \tag{20}$$

$$0 \leq f_i(t), \quad \forall i \in \{1, \ldots, L\}, t \in T_i, \tag{21}$$

$$\sum_{t \in T_i} f_i(t) \geq D_i, \quad \forall i \in \{1, \ldots, L\}. \tag{22}$$

The capacity constraint (20) indicates that the total traffic routed to link e from across all flows, tunnels, segments, and shortest paths, cannot exceed θ times the link capacity. Since ECMP is used for routing within any segment s, each shortest path p of segment s receives flow equal to $\frac{f_i(t)}{|\hat{P}_s|}$. Regarding the TE asymptotic complexity, we have the following result when M is fixed and not part of the input:

Proposition 3: For fixed M with respect to the input graph G, the TE_{LU} problem described by (19)-(22) can be solved in weakly polynomial time.

Proof: The number of commodities L cannot exceed $|V| \cdot (|V| - 1)$, and the number $|T_i|$ of tunnels per commodity i is upper bounded by $\binom{K}{0} + \cdots + \binom{K}{M}$, where $K \leq |V|$. For fixed M w.r.t. the input graph G, $|T_i|$ has polynomial size w.r.t the graph. Finally, the number S_t of segments per tunnel cannot exceed $K + 1 \leq |V| + 1$, since a tunnel can use at most all K middlepoints. For the inner sum $\sum_{p \in \hat{P}_s} \frac{I_{p,e}}{|\hat{P}_s|}$, note that it basically denotes the percentage of shortest paths for segment s that use link e. This percentage naturally appears in the definition of betweenness centrality [6], and we can compute it in polynomial time by using the techniques therein.[6]

Thus, we have proved that for fixed M, the LP has a polynomial number of variables, and a polynomial number of constraints whose coefficients can be computed in polynomial time. The proposition then immediately follows by standard results in linear programming [31], [32]. □

Given the connection between TE_{MF} and TE_{LU}, we can similarly show that TE_{MF} can be solved in weakly polynomial time. As we observed, the TE problem is naturally related to the betweenness centrality that we discuss in §VI-B. Thus, constraint (20) reveals interesting connections between the popular centrality metric and node-constrained TE with shortest paths.

Note that the TE is not polynomial just for the specific setting where there are K middlepoints and each logical path can use up to M of them. In particular, it remains polynomial if we constrain each logical path to contain all K middlepoints in the same order (instead of up to M), where K can be as large as $|V|$. Interestingly, even this simple variant can become NP-hard if we impose simple constraints, as we show next.

B. Acyclic Variant With Shortest Paths

One challenge with node-constrained TE with shortest paths is that it may produce source-destination paths with edge repetitions, i.e., walks. Even in the case of just one middlepoint per path, it is possible that a (simple) shortest path from the source s to a middlepoint M shares an edge e with a shortest path from M to the destination d. So, even though the paths for any given segment are simple, the resulting $s - d$ path may not even be a path. This may reduce the performance of TE, because it increases the link load on the reused edges and may lead to higher link utilization. Figure 5 depicts a directed graph with $n > 2$, where the shortest path from w to t is $w - u_1 - u_2 - t$. The shortest paths from s to w and w to t trivially overlap, since they share the edge $u_1 - u_2$.

In that case a natural question arises: what if we consider node-constrained TE with shortest paths, under the condition that the resulting walk from the source to the destination is a path or even a simple path? As our subsequent analysis shows, traffic engineering generally becomes NP-hard, even for just

---

[5]We emphasize that the problem of determining an *optimal* set of middle-points of a given size in segment routing is hard, and a large part of the prior literature on segment routing has tried to address that question. However, this problem is out of the scope of this work.

[6]Even though the number of shortest paths between two nodes can be exponential, the percentage of shortest paths that go through a specific node is polynomially computable. This is why betweenness centrality is polynomially computable [6].
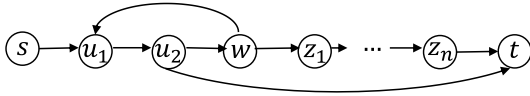
Fig. 5. Cycle in shortest-path based segment routing.

one commodity, and even in the special case where the traffic must use all middlepoints in the input order. To prove this fact, we first introduce the following fundamental result due to Eilam-Tzoreff [9].

Theorem 2 (NP-hardness of kDSP [9]): Given a graph $G = (V, E)$ and k pairs of distinct vertices (u_i, v_i), $1 \leq i \leq k$, the kDSP problem of computing k pairwise disjoint shortest paths P_i between u_i and v_i is NP-complete, when k is part of the input. This result holds for all four versions of the kDSP problem, namely, node or edge-disjoint paths for directed or undirected graphs.

Proposition 4: The TE_{MF} and TE_{LU} problems in a directed graph with K middlepoints (1) *using only shortest paths and* (2) *only allowing paths or simple paths from a source to a destination that use all middlepoints are NP-hard, even for just one commodity, when K is part of the input.*

Proof sketch: We can show the statement by making similar arguments as for general node-constrained TE in §III. For TE_{MF}, the idea is to first show that we can solve kDSP if and only if we can solve the corresponding TE_{MF} formulation.

Indeed, assume a directed graph $G = (V, E)$, two distinct nodes s, t in V, and K distinct nodes $s \neq M_i \neq t$ in V, $1 \leq i \leq K$. Consider we do node-constrained routing from s to t using nodes M_i as our K middlepoints. We will show that the TE_{MF} problem is NP-hard by a reduction from the kDSP problem.

Concretely, assume for instance the kDSP node-disjoint problem in Theorem 2. We construct a new graph G' as follows. For each i, $1 \leq i \leq K - 1$, we introduce a new node M_i along with the two directed edges $e_{in}^i = (v_i, M_i)$ and $e_{out}^i = (M_i, u_{i+1})$. Moreover, we associate each edge in G' with a unit capacity, and we assume the single commodity (s, t) with source $s = u_1$ and destination $t = v_K$. We now argue that there are K node-disjoint shortest paths between u_i and v_i, if and only if TE_{MF} with the single commodity (s, t) and the $K - 1$ middlepoints $M_1, \ldots, M_{K-1}$ accepts a positive solution.

This can be proven using similar techniques as in §IV-B. The only difference is that an edge-cut now corresponds to a set of edges whose removal results in no path (or simple path) using shortest paths from the source to the destination through the middlepoints. NP-hardness for TE_{LU} follows immediately by Lemma 1, in a similar spirit as Corollary 1. □

Proposition 4 assumes that the number of middlepoints K is part of the input since in general we can have up to $|V|$ middlepoints. What about the case when k is fixed? When $k = 2$, [9] provides a polynomial algorithm for the undirected case of kDSP, whereas the complexity for the directed case when $k = 2$ remains open[7]. On the other hand, only few partial results are known when k is fixed and greater than 2 [3].

---

[7]The problem appears to have been answered in the affirmative in [3], when the length of each edge is positive.

V. APPLICATION TO FLOW CENTRALITY

Having studied various node-constrained TE formulations, in this section we move to another fundamental and practical question: how to actually select the middlepoints that lead to good TE performance in the first place? We draw upon a concept called flow centrality from graph theory as an intuitive solution approach. Flow centrality, first introduced by Freeman et al. [20], characterizes a node's significance in terms of the maximum flow that can go through that node in the underlying flow network. Flow centrality is closely related to node-constrained TE for two reasons. First, it involves by definition node-constrained traffic. Second, flow centrality and its group extension answer the question of how to select the middlepoints in the general node-constrained TE (§III), in order to maximize the traffic through these points. Note that flow centrality is naturally related to the first TE type TE_{MF}, since it cares about the maximum flow objective.

The rest of this section is organized as follows. We provide two flow centrality definitions for single- and multi-commodity flow networks and discuss their computational complexity in §V-A. Furthermore, we introduce and analyze the group flow centrality and N-group maximum flow in §V-B, and show that unlike other common graph centralities it does not fall under the framework of submodular function maximization.

A. Flow Centrality

The original flow centrality [20] of a node $w \in V$ in a flow network $G = (V, E, c)$ is:

$$\gamma(w) = \frac{\sum\limits_{s', t' \in V - \{w\} \mid s' \neq t'} \nu_{\max}^w (s', t')}{\sum\limits_{s', t' \in V - \{w\} \mid s' \neq t'} \nu_{\max} (s', t')}, \qquad (23)$$

where $\nu_{max}(s', t')$ is the maximum flow in the single-commodity flow network with commodity (s', t'), and $\nu_{max}^w(s', t')$ is the maximum flow through node w in the single-commodity flow network with commodity (s', t'). Thus, the flow centrality of node w represents the percentage of the maximum flow that can go through w for a demand chosen uniformly at random.

For multi-commodity networks with given commodities, we introduce an alternative definition:

$$\widetilde{\gamma}(w) = \frac{\nu_{max}^w(\boldsymbol{s}, \boldsymbol{t})}{\nu_{max}(\boldsymbol{s}, \boldsymbol{t})}, \qquad (24)$$

which denotes the ratio of the maximum multi-commodity flow $\nu_{max}^w(\boldsymbol{s}, \boldsymbol{t})$ that can go through node w to the maximum multi-commodity flow, assuming we are given $(\boldsymbol{s}, \boldsymbol{t})$.

The basic difference in the two definitions is that the former considers equiprobably all possible source-destination pairs, while the latter focuses on the actual commodities in the flow network. Thus, the former is based on the single-commodity formulation, and the latter on the multi-commodity one.

For directed graphs, we show:

Proposition 5: Given a flow network $G = (V, E, c)$ with directed edges and a node $w \in V$, it is NP-hard to compute $\gamma(w)$ or $\widetilde{\gamma}(w)$.
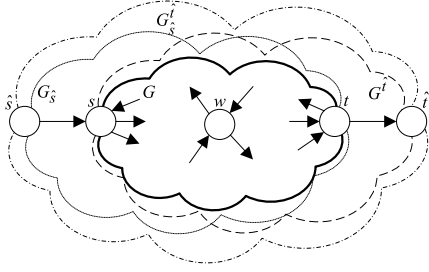
Fig. 6.   Illustration of Proposition 5.

Proof: For the first statement, our strategy will be to show that computing $\gamma(w)$ cannot be less hard than computing $\nu_{max}^w(s,t)$, for any source-destination pair (s,t) with $s \neq t$ and $s, t \neq w$. This, in turn, establishes NP-hardness for $\gamma(w)$. For convenience, we introduce the shorthand notation $\mathcal{S}^w(G) = \sum_{s',t' \in V-\{w\}|s' \neq t'} \nu_{max}^w(s',t')$ to represent the numerator for G.

In this direction, we consider the new flow network $G_{\hat{s}}$, which we construct from G by introducing a new node $\hat{s}$ and a directed edge from $\hat{s}$ to s with capacity equal to the sum of capacities of outgoing edges from s, i.e. $c((\hat{s},s)) = \sum_{e=(s,v) \in E} c(e)$. We now consider the quantity $\mathcal{S}^w(G_{\hat{s}})$ for the new flow network $G_{\hat{s}}$. $G_{\hat{s}}$ contains the same source-destination pairs as G, plus pairs of the form $(\hat{s}, t'), \forall t' \in V$. The reason is that node $\hat{s}$ can only act as a source but not a destination for nodes in G. As a result, we get:

$$\mathcal{S}^w(G_{\hat{s}}) = \mathcal{S}^w(G) + \sum_{t' \in V} \nu_{max}^w(\hat{s}, t'). \tag{25}$$

Next, we consider in a similar fashion the flow network $G^{\hat{t}}$, which we construct from G by introducing a new node $\hat{t}$ and a directed edge from t to $\hat{t}$ with capacity equal to the sum of capacities of incoming edges to t, i.e. $c((t,\hat{t})) = \sum_{e=(v,t) \in E} c(e)$. We then proceed as before to show:

$$\mathcal{S}^w(G^{\hat{t}}) = \mathcal{S}^w(G) + \sum_{s' \in V} \nu_{max}^w(s', \hat{t}). \tag{26}$$

Finally, we consider the flow network $G_{\hat{s}}^{\hat{t}}$, which we construct from G by applying both steps described above for $G_{\hat{s}}$ and $G^{\hat{t}}$. The various graph constructions are illustrated in Figure 6. Let's now focus on $\mathcal{S}^w(G_{\hat{s}}^{\hat{t}})$. This quantity will contain the source destination pairs (i) within G, (ii) from $\hat{s}$ to nodes in G, (iii) from $\hat{t}$ to nodes in G, and (iv) from $\hat{s}$ to $\hat{t}$. But (iv) is precisely the same as from s to t due to the value of capacities of the new edges we introduced. Thus, we can write:

$$\mathcal{S}^w(G_{\hat{s}}^{\hat{t}}) = \mathcal{S}^w(G) + \sum_{t' \in V} \nu_{max}^w(\hat{s}, t')$$
$$+ \sum_{s' \in V} \nu_{max}^w(s', \hat{t}) + \nu_{max}^w(s,t). \tag{27}$$

Combining (25), (26), (27), we get:

$$\nu_{max}^w(s,t) = \mathcal{S}^w(G_{\hat{s}}^{\hat{t}}) - \mathcal{S}^w(G_{\hat{s}}) - \mathcal{S}^w(G^{\hat{t}}) + \mathcal{S}^w(G). \tag{28}$$

(28) suggests that computing $\gamma(w)$ (and in turn $\mathcal{S}^w$) cannot be less hard than computing $\nu_{max}^w(s,t)$, since being able to compute the former implies that we can easily compute the latter via (28), which only involves polynomial-time constructions. Given it is NP-hard to compute ν_{max}^w by Proposition 1, computing $\gamma(w)$ is also NP-hard.

For the second statement, note again that $\widetilde{\gamma}(w)$ is a fraction of two terms. The denominator can be computed in strongly polynomial time [44]. If $\widetilde{\gamma}(w)$ were polynomially computable, then we could compute ν_{max}^w in strongly polynomial time as the product of $\widetilde{\gamma}(w)$ and ν_{max}, which is a contradiction since by Proposition 1 it is NP-hard to compute ν_{max}^w. □

For undirected graphs, we can similarly prove that:

Corollary 2: Given a flow network $G = (V, E, c)$ with undirected edges and a node $w \in V$, we can compute the flow centrality $\gamma(w)$ in Equation (23) or the flow centrality $\widetilde{\gamma}(w)$ in Equation (24) in strongly polynomial time.

Proof: It follows immediately from the fact that $\nu_{max}^w(s,t)$ is strongly polynomially computable in undirected graphs. □

B. Group Flow Centrality and N-Group Maximum Flow

In this section, we introduce the concept of multi-commodity group flow centrality, which is a generalization of the multi-commodity flow centrality $\widetilde{\gamma}$ through node w in Equation (24) to a group of nodes C. In this work, we focus on the group extension of $\widetilde{\gamma}$ since in many practical applications we are given the commodities and must decide on a good set of nodes to select as middlepoints. Similar definitions exist in prior work on graph centrality; for instance, group betweenness centrality [12] naturally generalizes betweenness centrality [21] to a group of nodes. Group flow centrality is based on the concept of group maximum flow.

Definition 3: The group multi-commodity maximum flow $\mathcal{GF} : 2^V \to \mathbb{R}_{\geq 0}$ in a multi-commodity flow network with commodities $(\boldsymbol{s}, \boldsymbol{t})$ is a function which, for any group of nodes $C \subseteq V$, returns the maximum multi-commodity flow $\nu_{max}^C(\boldsymbol{s}, \boldsymbol{t})$ that can go through any node in C, i.e. $\mathcal{GF}(C) = \nu_{max}^C(\boldsymbol{s}, \boldsymbol{t})$.

Definition 4: The group flow centrality of a group $C \subseteq V$ in a multi-commodity flow network is defined as:

$$\tilde{\gamma}(C) = \frac{\nu_{max}^C(\boldsymbol{s}, \boldsymbol{t})}{\nu_{max}(\boldsymbol{s}, \boldsymbol{t})}. \tag{29}$$

The group flow centrality represents the percentage of the maximum multi-commodity flow that goes through any node in group C for a given set of demands. It is obviously NP-hard for directed graphs as a generalization of the $\widetilde{\gamma}$ flow centrality which is NP-hard by Corollary 2. But then the group flow centrality is also NP-hard for directed graphs by Equation (29).

An important question concerns the selection of a group of nodes of at most a given size which achieves the largest possible maximum flow, or, equivalently, maximizes the group flow centrality. For this purpose, we introduce the N-*group maximum multi-commodity flow* with $N \in \mathbb{N}^+$:

$$\mathcal{GF}^N = \max_{C \subseteq V : |C| \leq N} \nu_{max}^C(\boldsymbol{s}, \boldsymbol{t}). \tag{30}$$
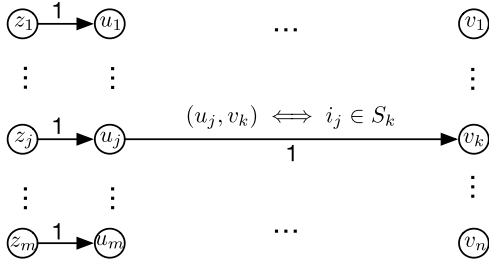
Fig. 7. Illustration of Proposition 6.

For single-commodity networks, the question is trivial since we can always put the source into the set C, but the problem turns out to be NP-hard in directed multi-commodity networks.

Proposition 6: The N-group maximum multi-commodity flow is NP-hard for directed graphs.

Proof: We prove NP-hardness by reduction from the maximum coverage problem (MCP) [39], which is NP-hard. Assume a set of m items $I = \{i_1, \ldots, i_m\}$ and a collection of n sets $S = \{S_1, \ldots, S_n\}$, where each S_i contains elements from I. Given a positive integer $N \leq n$, the MCP tries to select a subset of $S' \subseteq S$ of cardinality $|S'| \leq N$ such that the maximum number of elements are covered, i.e. the union of the selected sets has maximal size. We can reduce the MCP to the N-group maximum single-commodity flow by constructing a directed graph $G = (\mathcal{M}, E)$ as follows. $\mathcal{M}$ contains a pair of nodes z_j and u_j for each item i_j that appears in I, and one node v_k for each set S_k. We denote the set of nodes z_j as Z, the set of nodes u_j as U, and the set of nodes v_k as V, so that $\mathcal{M} = Z \cup U \cup V$. We introduce m edges of the form (z_j, u_j), $j = \{1, \ldots, m\}$. We add edge (u_j, v_k), if and only if set S_k contains item i_j. All edges have a capacity of 1. Finally, for each pair (i_j, v_k) with the property that set S_k contains item i_j, we consider a commodity (z_j, v_k) with source z_j and destination v_k. Thus, we add commodity (z_j, v_k), if and only if the graph contains edge (u_j, v_k). All commodities have infinite demand. The polynomial construction is depicted in Figure 7.

We now prove that the maximum coverage problem has value C_{max}, if and only if the N-group maximum flow has a value of C_{max}. Assume first that the MCP has a value of C_{max}. We can then construct a corresponding flow in G as follows. For each item i_j that is covered in the optimal MCP solution, we randomly pick one set S_k that covers i_j in the solution (there may be more than one sets covering i_j). We then send one unit of flow for commodity (z_j, v_k) from z_j to node u_j, and subsequently from u_j to v_k. The constructed multi-commodity flow is valid since it is easy to verify that it respects all capacity and conservation constraints. Based on that flow, we can also form a N-group flow. Indeed, by definition the MCP contains at most N sets S_i, so there can be at most N nodes of the form v_k participating in the flow. Since the entire flow has to pass through these nodes, we claim that the above flow is a N-group flow passing through (at most) N nodes of the form v_k. So, the N-group multi-commodity flow is at least C_{max}.

For the reverse direction, we argue that the N-group maximum flow cannot be greater than C_{max}. To show this, assume

a group $\mathcal{G}$ of at most N nodes that can accept a flow of value $C_N > C_{max}$. The nodes in $\mathcal{G}$ can belong to either Z or U or V. However, note that the flow through node z_j (for any commodity of the form (z_j, v_k)) is equal to the flow through u_j. Hence, we can replace any node z_j appearing in $\mathcal{G}$ by node u_j, without affecting the value of the group flow. Let's thus write $\mathcal{G} = U_N \cup V_N$, where $U_N \subseteq U$ and $V_N \subseteq V$. We also define $U' \subseteq U - U_N$ to be the subset of nodes in $U - U_N$ that transmit some positive (non-zero) flow to any node in V_N. We next argue that for any node $u_j \in U_N$, the total flow through u_j over all commodities with source z_j must be one. If that were not the case, we could form a N-group flow of higher value by increasing the flow for any commodity (z_j, v_k) until the total flow on edge (z_j, u_j) becomes one. This is possible since any commodity of the form (z_j, v_k) can only route its flow through u_j. This process respects all capacity constraints and achieves a higher group flow, which is a contradiction since we assumed a maximum N-group flow. Similarly, we argue that the total flow over all commodities from any node $u_j \in U'$ to the set of nodes in V_N must be one; otherwise, we could get a maximum N-group flow of higher value by first removing any flow (if any) through u_j to nodes in $V - V_N$, and subsequently increasing the flow for any commodity (z_j, v_k) where $v_k \in V_N$ until the total flow through node u_j becomes 1. This is always possible and respects all capacity constraints. The two observations imply that in the maximum N-group flow, the total flow over all commodities to nodes in U_N or U' must be one, and as a result C_N is an integer, equal to the cardinality $|U_N| + |U'|$. Based on the N-group maximum flow, we can form a coverage for the original problem as follows. For each $u_j \in U_N$, we pick any node $v_k \in V$ where u_j transmits some positive flow (at least one such node must exist); similarly, for each $u_j \in U'$, we pick any node $v_k \in V_N$ where u_j transmits some positive flow. Based on that, each item i_j, where $u_j \in U_N$ or $u_j \in U'$, is assigned to set S_k corresponding to the v_k above. The constructed coverage has value C_N since it uses all C_N items in $U_N \cup U'$. Moreover, it uses at most N sets S_k, since the number of sets cannot exceed the sum $|U_N| + |V_N| \leq N$. This is a contradiction, since we assumed that the maximum coverage has value $C_{max} < C_N$. $\square$

An interesting corollary of Proposition 6 is the following.

Corollary 3: The N-group maximum flow is not possible to approximate in directed graphs within $1 - \frac{1}{e} + o(1)$, unless P=NP.

Proof: From Proposition 6 we can compute the MCP by solving an N-group multi-commodity maximum flow problem. In particular, the MCP has a value equal to C_{max}, if and only if the N-group maximum flow has the same value C_{max}. If we were able to approximate the latter problem within $1 - \frac{1}{e} + o(1)$, that would imply that we can use the above construction to approximate the MCP within a factor of $1 - \frac{1}{e} + o(1)$, which is however not possible unless P=NP [14]. $\square$

A natural question is whether group multi-commodity flow falls under the paradigm of submodular function maximization. First, the MCP that we used in the reduction of Proposition 5 falls under this paradigm. Second, similar results already
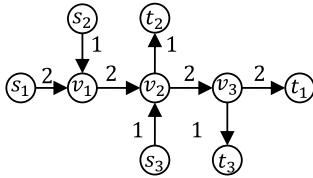
Fig. 8.   Illustration of Lemma 6.

exist in graph centrality theory. For instance, group closeness centrality is shown to be NP-hard, monotone, and submodular [7]. This is also true for group betweenness centrality [8].

Definition 5: Consider a finite set of elements U and a function $g : 2^U \to \mathbb{R}_{\geq 0}$. We call g monotone if adding an element to any set S cannot cause the function to decrease, i.e., $g(S \cup \{v\}) \geq g(S)$ for all $v \in U$ and $S \in 2^U$. Furthermore, we call g submodular if the marginal gain from adding an element to any set S is at least as high as the marginal gain from adding the same element to a superset of S, i.e., $g(S \cup \{v\}) - g(S) \geq g(T \cup \{v\}) - g(T)$ for all $v \in U - T$ and pairs of sets $S \subseteq T$.

Interestingly, our next result shows that multi-commodity group flow is not submodular even though it is monotone.

Lemma 6: The function $\mathcal{GF} : 2^V \to \mathbb{R}_{\geq 0}$ is monotone but not submodular in directed or undirected graphs.

Proof: For monotonicity, note that adding a node can never decrease the maximum group flow, since an additional node does not decrease the number of available paths for the flow (either it increases them or it leaves their number unchanged).

To show that $\mathcal{GF}$ is not submodular in directed graphs, it is sufficient to construct a proper counterexample. To this goal, consider the directed multi-commodity flow network of Figure 8 with three commodities: commodity (s_1, t_1) with a demand of 2 units, and commodities (s_2, t_2) and (s_3, t_3), each with a unit demand. Assume the following two sets $S = \{s_1\}$ and $T = \{s_1, s_2\}$, with $S \subset T$. For the corresponding group multi-commodity flows it holds that $\mathcal{GF}(S) = \mathcal{GF}(T) = 2$. The reason is that edge (v_1, v_2) with a capacity of 2 acts as the bottleneck that limits the total network flow to a maximum of 2. Consider now node s_3. We can again argue that $\mathcal{GF}(S \cup \{s_3\}) = 2$, since edge (v_2, v_3) with capacity 2 acts as the bottleneck that limits the total group flow to 2 units. However, note that $\mathcal{GF}(T \cup \{s_3\}) = 3$: commodity (s_1, t_1) sends one unit of flow along $s_1 \to v_1 \to v_2 \to v_3 \to t_1$, commodity (s_2, t_2) sends one unit along $s_2 \to v_1 \to v_2 \to t_2$, and commodity (s_3, t_3) sends one unit along $s_3 \to v_2 \to v_3 \to t_3$. This suggests that $\mathcal{GF}(T \cup \{s_3\}) - \mathcal{GF}(T) = 3 - 2 = 1 > \mathcal{GF}(S \cup \{s_3\}) - \mathcal{GF}(S) = 1 - 1 = 0$, which implies that the group multi-commodity flow $\mathcal{GF}$ is not a submodular function. Finally, to show that $\mathcal{GF}$ is not submodular in undirected graphs, we can use exactly the same counterexample as above, except that the edges are undirected this time. $\square$

One implication of Lemma 6 is that we cannot use the standard greedy algorithm by Nemhauser et al. [39] to find a $(1 - \frac{1}{e})$-approximate solution to the group flow. We note nevertheless that the greedy algorithm would in any case not be practical for directed graphs, given that computing the maximum w-flow is already NP-hard by Proposition 1.

VI. RELATED WORK

A. Segment Routing and TE

Segment routing [15]–[17] is a recent paradigm that facilitates packet forwarding via a series of segments. Segment routing has been explored with TE. Bhatia et al. [4] apply 2-segment routing to TE, where any logical path contains only one middlepoint and thus two segments. Hartert et al. [25], [26] propose heuristics to solve various TE problems with segment routing. Contrary to these works, our goal is to study the fundamentals of several variants of node-constrained TE for both directed and undirected graphs. Aubry et al. [2] propose to use segment routing for continuous monitoring of the data plane of the network with a single box. Segment routing is used to force probe packets to traverse specific paths. Giorgetti et al. [24] propose algorithms for segment routing label stack computation that guarantee minimum label stack depth. Such use cases are beyond our work.

TE has been extensively studied in carrier networks [10], [19], [26], [27], [30], [45], and recently in data center backbone WANs [23], [28], [29], [36] with software defined networking [13]. In general it is assumed that TE can use any valid path in the network, or any path from a predetermined set of paths. Node-constrained TE is clearly different.

B. Graph Centrality

The centrality concept from graph theory and network analysis [40] identifies the most important vertices in a graph. Centrality was first developed in social network analysis [5], [21] to determine the most influential nodes. We review two relevant centrality metrics. *Betweenness* centrality characterizes the power of a node in terms of the number of shortest paths that go through that node for a randomly picked source-destination pair. Brande's algorithm can compute this centrality in polynomial space and time [6]. *Closeness* centrality [22] of a node is calculated as the sum of the length of the shortest paths between that node and all other nodes in the graph. As opposed to the aforementioned *individual* centrality, the *group* centrality of a group of nodes $C \subseteq V$ refers to the combined centrality of the group [12]. Group betweenness centrality can be approximated within a factor $1 - \frac{1}{e}$ to the optimal [8], [42]. Group closeness centrality can also be approximated within a factor of $1 - \frac{1}{e}$ using the standard greedy algorithm [7].

Note that graph centralities have been applied to routing in some SDN problems, such as service chain embedding [38] and incremental SDN deployment [35], [37]. Solutions to these problems are based on degree centralities, and use greedy approximation algorithms exploiting submodularity [37].

VII. CONCLUSION

In this work we study node-constrained TE, where the traffic is constrained to go through specific middlepoints. We show that the general node-constrained TE problem is NP-hard for directed graphs, but strongly polynomial for undirected graphs. Furthermore, node-constrained TE with shortest paths is weakly polynomial, but the acyclic variant with shortest paths can become NP-hard. Lastly we address the question

of middlepoint selection, by applying flow centrality and additionally analyzing group flow centrality in general node-constrained TE.

Our work is important because node-constrained TE has wide applicability in emerging networking technologies such as segment routing. Our hardness results hint at the practical limitations of many variants of node-constrained TE. For this reason, an important direction for future research is the development of practical algorithms for the various computationally hard TE variants as well as the group multi-commodity flow.

Acknowledgment

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Upper Saddle River, NJ, USA: Prentice-Hall, 1993.

[2] F. Aubry *et al.*, "SCMon: Leveraging segment routing to improve network monitoring," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–9.

[3] K. Bérczi and Y. Kobayashi, "The directed disjoint shortest paths problem," Egerváry Res. Group, Budapest, Hungary, Tech. Rep. TR-2016-13, 2016.

[4] R. Bhatia, F. Hao, M. Kodialam, and T. V. Lakshman, "Optimized network traffic engineering using segment routing," in *Proc. IEEE INFOCOM*, Apr./May 2015, pp. 657–665.

[5] P. Bonacich, "Power and centrality: A family of measures," *Amer. J. Sociol.*, vol. 92, no. 5, pp. 1170–1182, 1987.

[6] U. Brandes, "A faster algorithm for betweenness centrality," *J. Math. Sociol.*, vol. 25, no. 2, pp. 163–177, 2001.

[7] C. Chen, W. Wang, and X. Wang, "Efficient maximum closeness centrality group identification," in *Proc. Australas. Database Conf.*, 2016, pp. 43–55.

[8] S. Dolev, Y. Elovici, R. Puzis, and P. Zilberman, "Incremental deployment of network monitors based on group betweenness centrality," *Inf. Process. Lett.*, vol. 109, no. 20, pp. 1172–1176, 2009.

[9] T. Eilam-Tzoreff, "The disjoint shortest paths problem," *Discrete Appl. Math.*, vol. 85, no. 2, pp. 113–138, 1998.

[10] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "MATE: MPLS adaptive traffic engineering," in *Proc. IEEE INFOCOM*, Apr. 2001, pp. 1300–1309.

[11] S. Even, A. Itai, and A. Shamir, "On the complexity of time table and multi-commodity flow problems," in *Proc. 16th Annu. Symp. Found. Comput. Sci.*, Oct. 1975, pp. 184–193.

[12] M. G. Everett and S. P. Borgatti, "The centrality of groups and classes," *J. Math. Sociol.*, vol. 23, no. 3, pp. 181–201, Aug. 1999.

[13] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN," *ACM Queue*, vol. 11, no. 12, pp. 20:20–20:40, Dec. 2013.

[14] U. Feige, "A threshold of ln n for approximating set cover," *J. ACM*, vol. 45, no. 4, pp. 634–652, 1998.

[15] C. Filsfils *et al.*, "Segment routing with MPLS data plane," in *Proc. Internet Eng. Task Force, Internet Draft (Work Prog.)*, 2014, p. 2. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-spring-segment-routing-mpls/

[16] C. Filsfils, P. Francois, and S. Previdi, "Segment routing use cases," IETF draft-filsfils-spring-segment-routing-use-cases-01, Oct. 2014.

[17] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois, "The segment routing architecture," in *Proc. IEEE Globecom*, Dec. 2015, pp. 1–6.

[18] S. Fortune, J. Hopcroft, and J. Wyllie, "The directed subgraph homeomorphism problem," *Theor. Comput. Sci.*, vol. 10, no. 2, pp. 111–121, 1980.

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

[20] L. C. Freeman, S. P. Borgatti, and D. R. White, "Centrality in valued graphs: A measure of betweenness based on network flow," *Social Netw.*, vol. 13, no. 2, pp. 141–154, Jun. 1991.

[21] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, no. 1, pp. 35–41, Mar. 1977.

[22] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social Netw.*, vol. 1, no. 3, pp. 215–239, 1979.

[23] A. Ghosh, S. Ha, E. Crabbe, and J. Rexford, "Scalable multi-class traffic management in data center backbone networks," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 12, pp. 2673–2684, Dec. 2013.

[24] A. Giorgetti *et al.*, "Path encoding in segment routing," in *Proc. IEEE Globecom*, Dec. 2015, pp. 1–6.

[25] R. Hartert, P. Schaus, S. Vissicchio, and O. Bonaventure, "Solving segment routing problems with hybrid constraint programming techniques," in *Proc. Int. Conf. Princ. Pract. Constraint Program.*, 2015, pp. 592–608.

[26] R. Hartert *et al.*, "A declarative and expressive approach to control forwarding paths in carrier-grade networks," in *Proc. ACM SIGCOMM*, 2015, pp. 15–28.

[27] J. He, M. Bresler, M. Chiang, and J. Rexford, "Towards robust multi-layer traffic engineering: Optimization of congestion control and routing," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 5, pp. 868–880, Jun. 2007.

[28] C.-Y. Hong *et al.*, "Achieving high utilization with software-driven WAN," in *Proc. ACM SIGCOMM*, 2013, pp. 15–26.

[29] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN," in *Proc. ACM SIGCOMM*, 2013, pp. 3–14.

[30] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the tightrope: Responsive yet stable traffic engineering," in *Proc. ACM SIGCOMM*, 2005, pp. 253–264.

[31] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica*, vol. 4, no. 4, pp. 373–394, 1984.

[32] L. G. Khachiyan, "Polynomial algorithms in linear programming," *USSR Comput. Math. Math. Phys.*, vol. 20, no. 1, pp. 53–72, 1980.

[33] J. Kurose and K. Ross, *Computer Networking: A Top-Down Approach*, 7th ed. London, U.K.: Pearson, 2016.

[34] A. S. Lapaugh and R. L. Rivest, "The subgraph homeomorphism problem," *J. Comput. Syst. Sci.*, vol. 20, no. 2, pp. 133–149, 1980.

[35] D. Levin, M. Canini, S. Schmid, F. Schaffert, and A. Feldmann, "Panopticon: Reaping the benefits of incremental SDN deployment in enterprise networks," in *Proc. USENIX ATC*, 2014, pp. 333–345.

[36] H. H. Liu, S. Kandula, R. Mahajan, M. Zhang, and D. Gelernter, "Traffic engineering with forward fault correction," in *Proc. ACM SIGCOMM*, 2014, pp. 527–538.

[37] T. Lukovszki, M. Rost, and S. Schmid, "It's a Match!: Near-optimal and incremental middlebox deployment," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 46, no. 1, pp. 30–36, Jan. 2016.

[38] T. Lukovszki and S. Schmid, "Online admission control and embedding of service chains," in *Proc. 22nd Int. Colloq. Struct. Inf. Commun. Complex.*, vol. 9439, 2015, pp. 104–118.

[39] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—I," *Math. Program.*, vol. 14, no. 1, pp. 265–294, 1978.

[40] M. Newman, *Networks: An Introduction*. Oxford, U.K.: Oxford Univ. Press, 2010.

[41] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Upper Saddle River, NJ, USA: Prentice-Hall, 1982.

[42] R. Puzis, D. Yagil, Y. Elovici, and D. Braha, "Collaborative attack on Internet users' anonymity," *Internet Res.*, vol. 19, no. 1, pp. 60–77, 2009.

[43] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*. Berlin, Germany: Springer-Verlag, 2003.

[44] E. Tardos, "A strongly polynomial algorithm to solve combinatorial linear programs," *Oper. Res.*, vol. 34, no. 2, pp. 250–256, 1986.

[45] H. Wang *et al.*, "COPE: Traffic engineering in dynamic networks," in *Proc. ACM SIGCOMM*, 2006, p. 194.

George Trimponias received the five-year Diploma degree in electrical and computer engineering from the National Technical University of Athens, Greece, and the Ph.D. degree from the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology. He is currently a Researcher with the Huawei Noah's Ark Lab, Hong Kong. His research interests include combinatorial optimization, artificial intelligence and machine learning, and game theory.
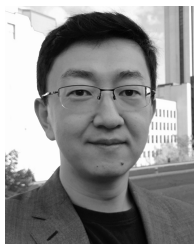
Yan Xiao received the B.Eng. and M.Eng. degrees from the College of Computer and Information, Hohai University, Nanjing, China, in 2013 and 2016, respectively. She is currently pursuing the Ph.D. degree with the Department of Computer Science, City University of Hong Kong. Her research interests include data mining, deep learning-based software engineering, software testing, and graph theory.

Hong Xu received the B.Eng. degree from The Chinese University of Hong Kong in 2007, and the M.A.Sc. and Ph.D. degrees from the University of Toronto in 2009 and 2013, respectively. He is currently an Assistant Professor with the Department of Computer Science, City University of Hong Kong. His research area is computer networking, particularly data center networks and big data systems. He is a member of the ACM. He was a recipient of an Early Career Scheme Grant from the Hong Kong Research Grants Council in 2014. He received the Best Paper Awards from the ACM TURC 2017 (Sigcomm China), IEEE ICNP 2015, and ACM CoNEXT Student Workshop 2014.

Xiaorui Wu received the B.Eng. degree from the University of Electronic Science and Technology of China in 2016. Since then, he has been with the Department of Computer Science, City University of Hong Kong, as a Ph.D. candidate. His research interests include distributed deep learning, model compression, and big data systems.

Yanhui Geng received the B.Eng. and M.Eng. degrees in electronic engineering and information science from the University of Science and Technology of China (USTC), in 2002 and 2005, respectively, and the Ph.D. degree in electrical and electronic engineering from The University of Hong Kong (HKU) in 2009. He is currently the Director of the Huawei Montreal Research Centre. Before that, he was a Senior Researcher and the Project Manager of the Huawei Noah's Ark Lab, Hong Kong. His research interests include artificial intelligence, machine learning, big data analytics, SDN (software-defined networks), and data-center networking. He received the IEEE ICC 2010 Best Paper Award.