Maximum Concurrent Flow Problem in MPLS-based Software Defined Networks

Tzu-Wen Chang*, Yao-Jen Tang†, Yu-Shu Chen*, Wei-Han Hsu*, and Ming-Jer Tsai*

***Abstract*—Multi-protocol label switching (MPLS) is supported by OpenFlow version 1.2 or higher and widely used in software defined networking (SDN) to achieve higher performance and flexibility. Due to the shortage of the ternary content addressable memory (TCAM), the number of forwarding entries installed in a switch needs to be bounded (node path-degree constraints). Besides, the maximum concurrent flow problem, which asks to maximize the minimum fraction of the flow of a commodity to its demand, is widely studied because of a wide range of applications. In this paper, we address the maximum concurrent flow problem while ensuring the flow routed for a commodity does not exceed its demand (demand constraints) and the node path-degree constraints are imposed, termed the bounded path-degree maximum concurrent flow (BPMCF) problem. We first show the BPMCF problem is NP-hard and intractable to devise any approximation algorithm. Then, we propose an algorithm for the BPMCF problem. Finally, we evaluate the performance of the proposed algorithm through computer simulations and experiments on Global Environment for Network Innovations (GENI) testbed using the real-life traces collected from SNDlib.**

I. Introduction

Recently, the software defined networking (SDN) and OpenFlow have received considerable attention in the networking research community [1]. Unlike traditional networks, SDN separates the control plane from the data plane, and thus, the network operator can manage packet forwarding using one or several centralized controllers to gain better network utilization. For instance, Google has built a SDN with OpenFlow routers to interconnect its data centers (G-Scale) and expects that there is an improvement of 20-30% in the network utilization [2]. A common SDN architecture comprises of the SDN Controller (SDN-C) and SDN Forwarding Elements (SDN-FEs). The SDN-FE stores forwarding entries using local ternary content addressable memory (TCAM), which is extremely expensive and power-hungry. Thus, the number of forwarding entries installed, or the size of the forwarding table, in the SDN-FEs is limited.

Nowadays, with the capabilities of fast packet forwarding and flexible traffic engineering, multi-protocol label switching (MPLS) [3] has been widely used in the core networks and supported by OpenFlow version 1.2 or higher [4]. In MPLS-based SDNs, each forwarding path is identified by a unique label (forwarding entry) in each SDN-FE. According to the analysis of [5], large entry usage will bring heavy burden to SDN-FEs, which will increase the forwarding table look-up latency and decline the network scalability. Moreover, even in small data centers comprising of several dozens of physical servers, the number of different flows can easily grow to several orders of magnitude above the number of SDN-FEs. This implies that the number of forwarding entries in the SDN-FE is very limited. Thus, this motivates the researchers to design a control plane algorithm able to select the right set of forwarding paths, such that the local constraints on the sizes of the forwarding tables in SDN-FEs are satisfied and the global objective (e.g., maximum flow, max-min fairness, maximum concurrent flow) in the SDN-C is reached [6]–[8].

The traffic engineering problem in networks can be analogous to a multi-commodity flow problem and has been addressed by many researchers. In [9], [10], algorithms are proposed for the maximum flow problem, which asks for the maximum total throughput of all commodities, in the telecommunication networks and large IP networks, respectively. In [11], the authors address how to dynamically reallocate the multi-commodity flow such that all commodities are served and the flow reallocation cost is minimized in SDNs. However, none of the above works considers fairness among commodities.

A number of works investigate the issue of fair bandwidth allocation for a multi-commodity problem. In [12], [13], algorithms are proposed for the multi-commodity flow problem with max-min fairness (the minimum flow of a commodity are maximized). Another line of researches consider completion time for the data-parallel applications (e.g., MapReduce jobs). More specifically, a job is finished until all the flows from the sources are completely transferred to the destinations and the completion time is determined by the slowest one among all flows. Thus, several works aim to maximize the minimum fraction of the flow (traffic rate) of a commodity to its demand, which is known as the maximum concurrent flow problem and widely studied [14]–[20]. In [21], the maximum concurrent flow problem where the total flow through a node are bounded is addressed. However, these algorithms for the multi-commodity flow problem do not address the constraints on the node path-degree.[1] Thus, numerous algorithms that address the node path-degree constraints are proposed for the maximum flow problem [6], [7] and the maximum concurrent

*Dept. of Computer Science, National Tsing Hua University, Hsinchu, Taiwan

†Information and Communications Research Laboratories, Industrial Technology Research Institute, Hsinchu, Taiwan

E-mail:{g9962515, mjtsai}@cs.nthu.edu.tw, yjtang@itri.org.tw, lazyeagles@gmail.com and frankhsu523@gapp.nthu.edu.tw

[1]In MPLS-based SDNs, we have to install a label (forwarding entry) to each SDN-FE of each forwarding path. And, the number of forwarding entries installed in the SDN-FEs is limited. Therefore, we limit the maximum number of paths that can go through a node (SDN-FE), and denote this constraint by *path-degree* as the same as in [6], [8].

flow problem [8]. In [8], a randomized algorithm is proposed to select the paths to route flow from a set of pre-determined forwarding paths (e.g., shortest paths) under the circumstance that the flow routed for a commodity is allowed to exceed its demand (i.e., the constraints on the demand are not imposed). However, selecting the forwarding paths without considering the utilization of network resources (e.g. bandwidth and forwarding table) might lead to poor performance. In addition, allowing the flow routed for a commodity to exceed its demand could over-allocate the network resources for some commodities, which may lead to improper link bandwidth allocation for the competing flows and thus deteriorate the max-min goal.

In this paper, we are motivated to study the maximum concurrent flow problem while ensuring the node path-degree constraints and the demand constraints are satisfied, under the circumstance that the set of forwarding paths is not pre-determined, termed the bounded path-degree maximum concurrent flow (BPMCF) problem. Our main contributions are summarized as follows:

1) We introduce the BPMCF problem, which has never been studied in the literature, and show the BPMCF problem is NP-hard and intractable to devise any approximation algorithm (Section II).
2) We propose an algorithm for the BPMCF problem (Section III). Precisely, we propose a novel method of computing the reference of a path p, $r(p)$, which is used to evaluate each path obtained by Garg and Könemann's algorithm (a well-known algorithm for the maximum concurrent problem without node path-degree constraints and demand constraints) [18]. In particular, $r(p)$ can well indicate the worth in both keeping the minimum fraction and satisfying the path-degree constraints. Then, with the evaluated $r(p)$, we can remove proper forwarding paths obtained by GK algorithm to satisfy the node path-degree constraints. Finally, we employ a linear program to route flow on each selected path such that the demand constraints are satisfied and the minimum fraction is maximized.
3) We conduct computer simulations and experiments on Global Environment for Network Innovations (GENI) [22] testbed using the real-life traces for Internet Service Provider (ISP) networks collected from SNDlib [23], and the results show our algorithm outperforms the related work in [8] and some naive methods (Section IV).

II. Bounded Path-Degree Maximum Concurrent Flow Problem

We first demonstrate the scenario in Section II-A. Subsequently, we formally define the problem in Section II-B and show its hardness in Section II-C.

A. The Scenario

Our system model follows the OpenFlow model. Consider a network consisting of an SDN-C and a number of SDN-FEs, where the number of forwarding entries is limited in each SDN-FE and the bandwidth of each link between two SDN-FEs is limited. There are several source-destination pairs. A flow is transferred from the source node to the destination node. In addition, each source-destination pair is associated with a flow requirement (demand). Due to the benefits of the multi-path routing for traffic engineering [24], the flow between a source-destination pair may be split among different paths. Since MPLS is employed, for each forwarding path, we install a label (forwarding entry) to each SDN-FE of that path. We also assume the SDN-C has the global information of the network. The control plane algorithm is to decide the forwarding paths and the flow on each chosen path for each source-destination pair (up to its flow requirement) such that the minimum fraction of the total flow between each source-destination pair to its flow requirement is maximized.

B. The Problem

We define the problem based on the studied scenario in Definition 1.

Definition 1. Given a (directed) network $G = (V, E)$ with link capacity $c_e \in \mathbb{R}^+$ associated with each link $e \in E$ and node capacity $b_v \in \mathbb{Z}^+$ associated with each node $v \in V$, and given a set of source-destination pairs K with maximum demand $d_k \in \mathbb{R}^+$ associated with each source-destination pair $k \in K$, the **B**ounded **P**ath-Degree **M**aximum **C**oncurrent **F**low (BPMCF) problem asks for the minimum fraction $\lambda \in [0, 1]$ and the set of forwarding paths such that

1) for each source-destination pair k, the total flow between the source (termed s_k) and the destination (termed t_k), where $t_k \neq s_k$, is at least $\lambda \cdot d_k$ and does not exceed d_k (demand constraints),
2) for each link e, the total flow through e does not exceed c_e (link capacity constraints),
3) for each node v, the number of forwarding paths going through v does not exceed b_v (node path-degree constraints), and
4) λ is maximized.

Let P_k be the set of enumerated forwarding paths of the source-destination pair k, and $\mathcal{P}$ be $P_1 \cup P_2 \cup \cdots \cup P_{|K|}$. Moreover, let c_p be the capacity of a path p in P_k, $k \in K$. More specifically, $c_p = \min\{\min_{e:e\in p}\{c_e\}, d_k\}$. Thus, the flow through a path p does not exceed c_p. In addition, variable $x(p)$ denotes the fraction of the flow on path p to c_p, and binary variable $y(p)$ denotes the path p is chose or not. The BPMCF problem is formulated as a mixed integer program (1a – 1h). Constraints in 1b ensure that, for each source-destination pair, the fraction of the total flow to its demand are at least λ. Constraints in 1c ensure that, for each source-destination pair, the total flow are bounded by its demand. Constraints in 1d (or 1e) ensure that the total flow (or the total number of flow paths) through a link (or node) are bounded by its capacity. Constraints in 1f and 1h ensure a flow is only routed on the chosen path.

$$\text{maximize} \quad \lambda \tag{1a}$$

$$\text{subject to} \quad \sum_{p \in P_k} x(p)c_p \geq \lambda \cdot d_k, \ \forall k \in K \tag{1b}$$

$$\sum_{p \in P_k} x(p)c_p \leq d_k, \ \forall k \in K \tag{1c}$$

$$\sum_{p:e \in p} x(p)c_p \leq c_e, \ \forall e \in E \tag{1d}$$

$$\sum_{p:v \in p} y(p) \leq b_v, \ \forall v \in V \tag{1e}$$

$$x(p) \leq y(p), \ \forall p \in \mathcal{P} \tag{1f}$$

$$x(p) \in [0, 1] \tag{1g}$$

$$y(p) \in \{0, 1\} \tag{1h}$$

C. Hardness

We show the BPMCF problem is NP-hard and cannot be approximated within any "reasonable" factor (unless $P = NP$) by demonstrating a polynomial-time reduction from the node disjoint paths problem (Definition 2), an NP-complete problem, in Theorem 1.

Definition 2. [25] Given an undirected graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ and a set of source-destination pairs $H = \{(s_1, t_1), (s_2, t_2), \cdots, (s_{|H|}, t_{|H|})\} \subseteq \mathbb{V} \times \mathbb{V}$. The node disjoint paths problem asks for a path p_h for each source-destination pair h such that path p_i and path p_j are node disjoint for every $i \neq j$, $i, j \in H$.

Theorem 1. *The BPMCF problem is NP-hard. More specifically, the BPMCF problem cannot be approximated within a factor of α for any $\alpha > 0$.*

Proof. We first show that the BPMCF problem is NP-hard. For every instance of the node disjoint paths problem, we construct a corresponding instance of the BPMCF problem as follows:

1) $V = \mathbb{V}$,
2) $E = \big\{(u, v), (v, u) | \{u, v\} \in \mathbb{E}\big\}$,
3) $c_e = 1$, $\forall e \in E$,
4) $b_v = 1$, $\forall v \in V$,
5) for each source-destination pair h in the instance of node disjoint paths problem, we create a corresponding source-destination pair k, where $(s_k, t_k) = (s_h, t_h)$ and $d_k = 1$.

Clearly, this instance can be constructed in polynomial time. It is then suffices to show that there exists a set of node disjoint paths for the set of source-destination pairs H in an instance of the node disjoint paths problem if and only if the minimum fraction λ of the corresponding instance constructed is 1 (optimal value).

For the "only if" part, suppose that $p_1, p_2, \cdots, p_{|H|}$ are the node-disjoint paths for source-destination pairs $1, 2, \cdots, |H|$, respectively. For each h in H, path $p_h = \{s_h = v_1, v_2, v_3, \cdots, v_{|p_h|} = t_h\}$ has consecutive undirected edges $\{v_1, v_2\}, \{v_2, v_3\}, \cdots, \{v_{|p_h|-1}, v_{|p_h|}\}$. Then, for each corresponding source-destination pair k, let p_k be a path with consecutive directed edges $(v_1, v_2), (v_2, v_3), \cdots, (v_{|p_h|-1}, v_{|p_h|})$ and the flow on p_k be 1. It is easy to verify that the set of paths $\{p_1, p_2, \cdots, p_{|K|}\}$ is a feasible solution with $\lambda = 1$ to the instance of the BPMCF problem. We omit the proof of the "if" part due to its similarity with that for the "only if" part.

Then we show that the BPMCF problem cannot be approximated within a factor of α for any $\alpha > 0$ unless $P = NP$. We use the same way to construct the instance of our problem for an instance of the node disjoint paths problem. It is suffices to show an α-approximation algorithm $\mathcal{A}$ can be used to solve the node disjoint paths problem, where $\alpha > 0$. More specifically, if the minimum fraction λ obtained by $\mathcal{A}$ in the instance constructed is greater than 0, there exists a set of node disjoint paths for H in the corresponding instance of node disjoint paths problem, and otherwise there does not. We omit the proof due to the similarity with that for the "if" part in the proof of the hardness, and thus, complete the proof. □

III. The Proposed Algorithm

Since the BPMCF problem is intractable to devise any approximation algorithm by Theorem 1, we propose a heuristic algorithm for the BPMCF problem. Our idea is to extend Garg and Könemann's algorithm (GK algorithm) for the maximum concurrent flow (MCF) problem in [18]. The MCF problem is the same as the BPMCF problem except that the demand constraints (i.e., the constraints in 1c) and the node path-degree constraints (i.e., the constraints in 1e) are not imposed. We notice that the set of paths $\mathcal{P}^{GK}$ with (x, y, λ) obtained by GK algorithm has a performance guarantee but may violate the path-degree constraints and the demand constraints. Hence, our goal is to make the obtained solution feasible while keeping λ (the minimum fraction) as large as possible.

Clearly, we could make the obtained solution satisfy the path-degree constraints by removing some paths from $\mathcal{P}^{GK}$ So, the basic idea is to select a subset, $\mathcal{P}^{ours}$, of $\mathcal{P}^{GK}$ to route the flow such that the path-degree constraints are satisfied and λ is not significantly affected. Recall that $x(p)$ denotes the fraction of the flow on path p to c_p. Intuitively, selecting the paths p with great $x(p)$ to $\mathcal{P}^{ours}$ seems to be able to keep λ. However, since the GK algorithm does not address the path-degree constraints, selecting the path with great $x(p)$ to $\mathcal{P}_{ours}$ has not much help for satisfying the path-degree constraints. Thus, we need a better reference of each path. Obviously, a proper reference of each path p, $r(p)$, has to embody in both keeping λ and satisfying the path-degree constraints. In the following, we present our approach consisting of three steps. First, for each path p in $\mathcal{P}^{GK}$, we compute a reference $r(p)$. Then, we select a subset, $\mathcal{P}^{ours}$, of $\mathcal{P}^{GK}$ based on the path reference. Finally, we determine the flow on each path in $\mathcal{P}^{ours}$ such that the demand constraints are satisfied and the minimum fraction is maximized.

Path Reference Computation: To obtain a proper reference $r(p)$ of each path p in $\mathcal{P}^{GK}$, we employ the following linear program, LP^{PRC}. Let $f_p = x(p) \cdot c_p$ denote the flow on a path p in $\mathcal{P}^{GK}$. In LP^{PRC}, the objective function (2a) and constraints in (2b) account for the goal to maximize

$\min_{k\in K}\{\frac{\sum_{p\in P_k\cap\mathcal{P}^{GK}} r(p)f_p}{d_k}\}$ and are to assign proper $r(p)$ to maximize $\tilde{\lambda}$ (in order to keep the minimum fraction λ obtained by GK algorithm). Constraints in (2c) ensure that the total $r(p)$ of the paths p through a node v is bounded by b_v and are to assign proper $r(p)$ such that the node path-degree constraints are satisfied. Thus, $r(p)$ can well indicate in both keeping λ and satisfying the path-degree constraints.

$$\text{maximize} \quad \tilde{\lambda} \tag{2a}$$

$$\text{subject to} \quad \sum_{p\in P_k\cap\mathcal{P}^{GK}} r(p)f_p \geq \tilde{\lambda}\cdot d_k,\ \forall k\in K \tag{2b}$$

$$\sum_{p:v\in p} r(p) \leq b_v,\ \forall v\in V \tag{2c}$$

$$r(p)\in[0,1] \tag{2d}$$

Path Selection: After we have $r(p)$ for each p in $\mathcal{P}^{GK}$, we can just select the paths p with great $r(p)$ to $\mathcal{P}^{ours}$. However, recall that our goal is to maximize the minimum fraction. So, we need to take the goal into account. We say that a path p is feasible if the node path-degree constraints are not violated when p is selected into $\mathcal{P}^{ours}$. The path selection proceeds in iterations. In each iteration, we select a feasible path p with the greatest $r(p)$ into $\mathcal{P}^{ours}$ for the source-destination pair k which has the minimum $\frac{\sum_{p\in P_k\cap\mathcal{P}^{ours}} r(p)f_p}{d_k}$ (ties are broken arbitrarily). The source-destination pair k becomes frozen if there is no feasible path between them. The path selection ends when all source-destination pairs are frozen.

Flow Optimization: When the path selection is done, it is observed that the capacities of the links on the unselected paths (i.e., $\mathcal{P}^{GK}\setminus\mathcal{P}^{ours}$) can be utilized by the selected paths going through that links. Thus, we aim to route the flows on paths in $\mathcal{P}^{ours}$ such that the demand constraints are satisfied and the minimum fraction is maximized. Note that the path-degree constraint of each node is not violated even if we route the flow on all paths in $\mathcal{P}^{ours}$. Thus, our current problem can be formulated as the following linear program, LP^{FO}.

$$\text{maximize} \quad \hat{\lambda} \tag{3a}$$

$$\text{subject to} \quad \sum_{p\in P_k\cap\mathcal{P}^{ours}} \hat{x}(p)c_p \geq \hat{\lambda}\cdot d_k,\ \forall k\in K \tag{3b}$$

$$\sum_{p:e\in p} \hat{x}(p)c_p \leq c(e),\ \forall e\in E \tag{3c}$$

$$\sum_{p\in P_k\cap\mathcal{P}^{ours}} \hat{x}(p)c_p \leq d_k,\ \forall k\in K \tag{3d}$$

$$\hat{x}(p)\in[0,1] \tag{3e}$$

Note that both the number of variables and the number of constraints are polynomial in LP^{PRC} and LP^{FO}. Thus, both LP^{PRC} and LP^{FO} can be solved in polynomial time using a linear program solver. The proposed algorithm for the BPMCF problem is described in Algorithm 1. An instance of the BPMCF problem and a solution generated by Algorithm 1 is shown in Fig. 1. The path selection proceeds in iterations until each source-destination pair is frozen (line 4). In iteration 1, k_{min} is set to (s_1,t_1) by random since $\frac{\sum_{p\in P_1\cap\mathcal{P}^{ours}} r(p)f_p}{d_1} = \frac{\sum_{p\in P_2\cap\mathcal{P}^{ours}} r(p)f_p}{d_2} = 0$ (line 5), and p_{max} is set to p_1 since path p_1 has the greatest $r(p)$ among all feasible paths between (s_1,t_1) (line 8). Thus, p_1 is added to $\mathcal{P}^{ours}$ in phase 1 (line 9). Similarly, paths p_2, p_3, and p_4 are added to $\mathcal{P}^{ours}$ in phases 2, 3, and 4, respectively. After $\mathcal{P}^{ours}$ is obtained, $\hat{x}(p_1)=\hat{x}(p_3)=1$, $\hat{x}(p_2)=0.286$, $\hat{x}(p_4)=0$, and $\hat{\lambda}=0.286$ are obtained by solving LP^{FO} (line 11).

Algorithm 1: The Algorithm for BPMCF Problem

Input : An infeasible solution $\mathcal{P}^{GK}, x, y, \lambda$ obtained by GK algorithm

1 Obtain $r(p)$ of each path p in $\mathcal{P}^{GK}$ by solving LP^{PRC} using a linear program solver
2 $\mathcal{P}^{ours}\leftarrow\emptyset$
3 $K^{frozen}\leftarrow\emptyset$
4 **while** $|K|>|K^{frozen}|$ **do**
5 $\quad k_{min}\leftarrow$ the source-destination pair with the minimum $\frac{\sum_{p\in P_k\cap\mathcal{P}^{ours}} r(p)f_p}{d_k}$
6 $\quad$ **if** *All paths of* k_{min} *are infeasible* **then**
7 $\quad\quad K^{frozen}\leftarrow K^{frozen}\cup\{k_{min}\}$ and **continue**
8 $\quad p_{max}\leftarrow$ the feasible path in $\mathcal{P}^{GK}$ with the greatest $r(p)$ between k_{min}
9 $\quad\mathcal{P}^{ours}\leftarrow\mathcal{P}^{ours}\cup\{p_{max}\}$
10 $\quad\mathcal{P}^{GK}\leftarrow\mathcal{P}^{GK}\setminus\{p_{max}\}$
11 Obtain $\hat{x}(p)$ of each path p in $\mathcal{P}^{ours}$ and $\hat{\lambda}$ by solving LP^{FO} using a linear program solver
12 **return** $\hat{x}, \mathcal{P}^{ours}, \hat{\lambda}$

IV. Numerical Results

In this section, we conduct computer simulations to evaluate the performance of Algorithm 1. In addition, we also conduct experiments on Global Environment for Network Innovations (GENI) [22] testbed. In both simulations and experiments, we use the real-life traces for Internet Service Provider (ISP) networks collected from SNDlib [23].

A. The Settings

Network Instances: The network instances were obtained from SNDlib. We used eight traces, including four small-scale ones (*abilene*, *atlanta*, *newyork*, *polska*) and four large-scale ones (*india35*, *cost266*, *pioro40*, *germany50*). Due to the computation limit, we only use four small-scale traces on GENI. Each trace contains the topology of the network, the capacity of each link, and the source-destination pairs with maximum demands. Since the capacity of each switch (b_v) is unavailable, b_v was set to $m\cdot|K|$ where $m\in[0.5,1]$.

Comparison Methods: We compared our algorithm with GK algorithm followed by one of three greedy strategies (denoted by GREEDY-1, GREEDY-2, and GREEDY-3) to remove the violation of the node path-degree constraints. GREEDY-1 iteratively removes the forwarding path p with minimum $\frac{\triangle\lambda(p)}{v(p)}$ from $\mathcal{P}^{GK}$ until the node path-degree constraints are satisfied, where $\triangle\lambda(p)$ denotes the decrease on the value of λ as p is removed, and $v(p)$ denotes the number of nodes on p which violates the node path-degree constraints. Similar to GREEDY-1, GREEDY-2 and GREEDY-3 remove the flow paths p with minimum $\triangle\lambda(p)$ and maximum $v(p)$, respectively. For

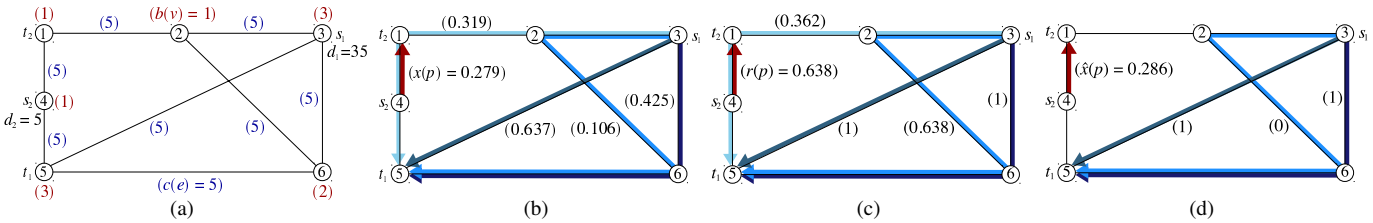

Fig. 1. Example of generating a solution by Algorithm 1. (a) The network G with two source-destination pairs, where $d_1 = 35$, $d_2 = 5$ and the capacity of a link (or node) is shown in the parenthesis. (b) The solution obtained by GK algorithm for G, where $p_1 = (3, 5)$, $p_3 = (3, 6, 5)$, $p_4 = (3, 2, 6, 5)$, and $p_5 = (3, 2, 1, 4, 5)$ are selected to route the flows between (s_1, t_1) and $p_2 = (4, 1)$ is selected to route the flow between (s_2, t_2). For each path p in $\mathcal{P}^{GK}$, $x(p)$ is shown in the parenthesis. (c) (or (d)) shows $\mathcal{P}^{GK}$ (or $\mathcal{P}^{ours}$), where $r(p)$ (or $\hat{x}(p)$) obtained by solving LP^{PRC} (or LP^{FO}) is shown in the parenthesis.

GREEDY-1, GREEDY-2, GREEDY-3 and Algorithm 1, the parameter ε in GK algorithm is set to 0.5. In addition, we also consider the randomized algorithm (denoted by RAN) proposed in [8]. Since the set of forwarding paths needs to be pre-determined, we used Yen's algorithm [26], as suggested by the authors, to generate k-shortest paths ($k = 3, 5$) for each source-destination pair. For RAN, we generated 100 solutions and selected the solution with the greatest minimum fraction. For the solution obtained by GREEDY-1, GREEDY-2, GREEDY-3 and RAN, we scaled down the flow on each path between a source-destination pair k by the violation factor if the flow between the source-destination pair k exceeded its maximum demand.

GENI testbed: We employ resources including a controller and OpenFlow-enabled switches to emulate SDN environment on GENI. In addition, we used Iperf [27] tool to generate UDP traffic between each source-destination pair and each experiment lasted for five minutes.

Metrics: We evaluated the minimum fraction and the total throughput of the solution obtained by Algorithm 1, GREEDY-1, GREEDY-2, GREEDY-3 and RAN. In addition, we studied the switch load distribution, where the switch load is the total number of the forwarding entries installed on a switch divided by its switch capability.

B. The Results

Minimum fraction (our goal) and total throughput: Fig. 2 shows the results for the impact of the switch capacity on the minimum fraction through computer simulations. In general, Algorithm 1 outperforms GREEDY-1, GREEDY-2, GREEDY-3 and RAN in all traces. Compared to GREEDY-1, GREEDY-2, and GREEDY-3, Algorithm 1 has a greater minimum fraction since our algorithm removes proper paths from $\mathcal{P}^{GK}$ to satisfy the path-degree constraints based on the reference of path p, $r(p)$, which indicates the worth of path p in both keeping the minimum fraction and satisfying the path-degree constraints. For RAN, the minimum fraction is 0 in all traces except for newyork and india35. This is because there are some source-destination pairs with mice demands. More specifically, for these source-destination pairs, the probability of selecting a pre-determined path as the forwarding path is very small, resulting in a high probability that the fraction of the flow to the demand is 0. It is observed that all

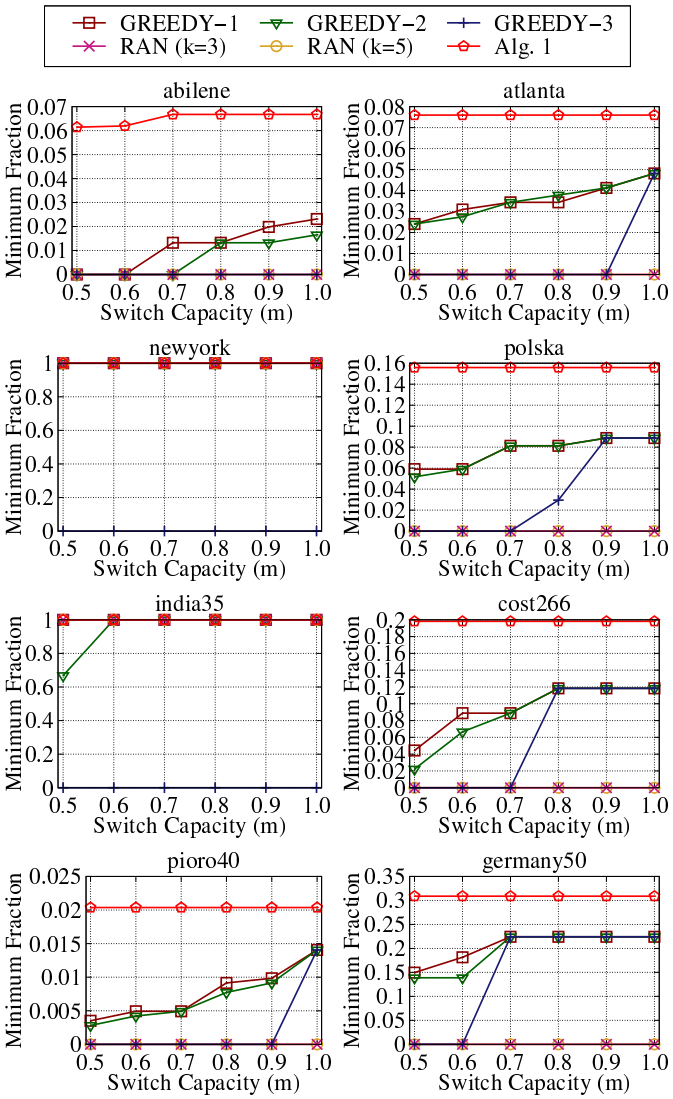

Fig. 2. The impact of the switch capacity on the minimum fraction through computer simulations, where the switch capacity is set to $m \cdot |K|$.

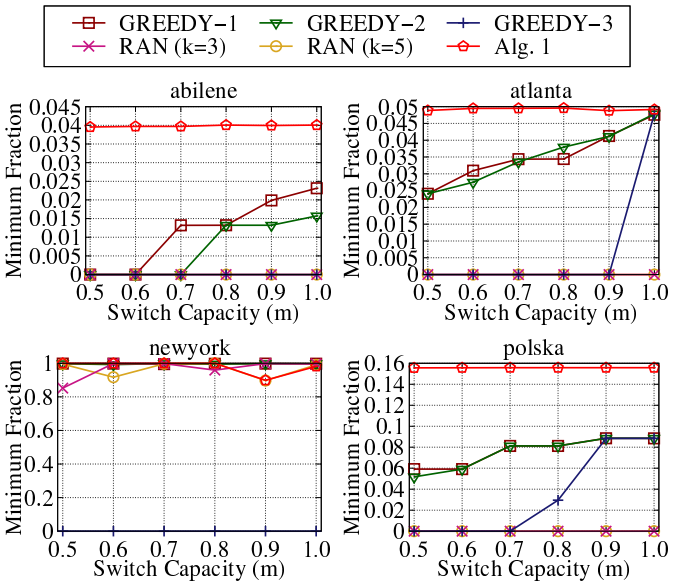

Fig. 3. The impact of the switch capacity on the minimum fraction on GENI, where the switch capacity is set to $m \cdot |K|$.

methods except for GREEDY-3 generate optimal solutions ($\lambda = 1$) in newyork and india35. This observation results from that all source-destination pairs are fully served due to the abundant network resources in newyork and india35. Fig. 3 shows the results for the impact of the switch capacity on the minimum fraction through experiments on GENI. As anticipated, the results through experiments on GENI are similar to those through computer simulations, and the minimum fraction of each algorithm on GENI is slightly smaller than that in computer simulations due to the packet loss in the real machines. Fig. 4 and Fig. 5 show that our algorithm outperforms or ties the compared methods in terms of the total throughput. Compared to GREEDY-1, GREEDY-2, and GREEDY-3, Algorithm 1 has a greater total throughput since our algorithm employs the step of flow optimization to re-utilize the link capacity occupied by the forwarding paths in $\mathcal{P}^{GK}$ unselected by our algorithm. It can be seen that for RAN, the total throughput is 0 in all traces except for newyork and india35. This is because RAN employs a linear program solver to perform flow optimization while the linear program solver tends to assign zero flow to all selected forwarding paths as the object value (i.e., the value of the minimum fraction) is 0 due to that no forwarding path is selected for some source-destination pairs with mice demands.

Switch load distribution: Fig. 6 shows the cumulative percentage of switches of switch load from 0 to 100% as $b_v = |K|$. Our algorithm outperforms GREEDY-1, GREEDY-2, and GREEDY-3 in all traces, and outperforms RAN in newyork and india35. This is because our algorithm employs less forwarding paths to route the flow for a source-destination pair due to that our algorithm selects the forwarding path based on the evaluated $r(p)$ which can well indicate the worth of path p in both keeping the minimum fraction and satisfying the path-degree constraints. For RAN, in all traces except for newyork and india35, the load of each switch is 0 since the total throughput is 0 (as can be seen in Fig. 4).

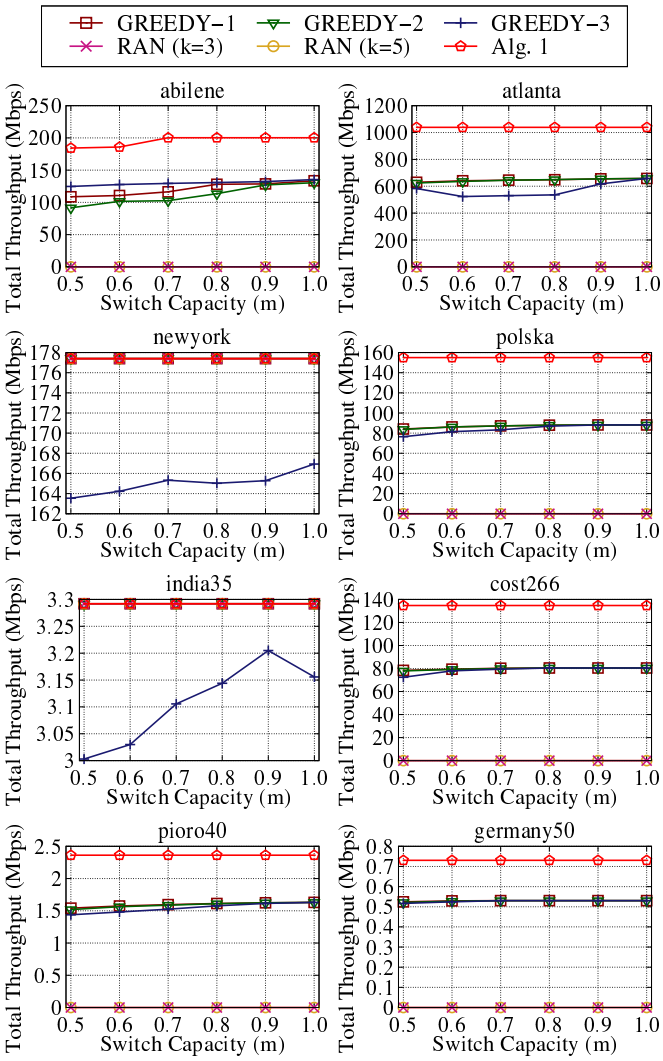

Fig. 4. The impact of the switch capacity on the total throughput through computer simulations, where the switch capacity is set to $m \cdot |K|$.

V. Conclusion

In this paper, we studied the bounded path-degree maximum concurrent flow problem, under the circumstance that the routing path set is not pre-determined. The problem asks for the paths and the flow on each chosen path for each source-destination pair (up to its demand) such that the minimum fraction of the total flow between each source-destination pair to its demand is maximized, while ensuring the link capacity and node path-degree are satisfied. To tackle this problem, through formulating and solving linear programs, we first compute the reference of each forwarding path obtained by GK algorithm (a well-known algorithm for the maximum concurrent problem without node path-degree and demand constraints) to decide whether to remove the forwarding path, and then optimize the flow of each selected path, where the reference of a forwarding path shows the worth in both keeping the minimum fraction and satisfying the path-degree constraints. Computer simulations and experiments results on GENI testbed using the real-life traces collected from SNDlib

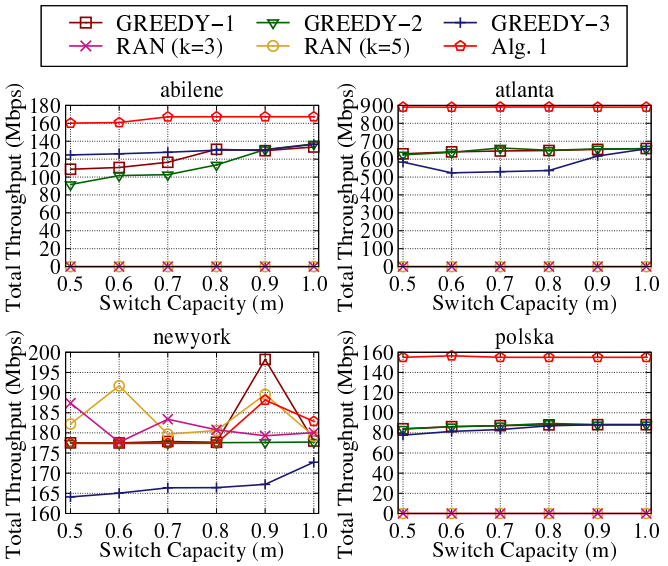

Fig. 5. The impact of the switch capacity on the total throughput on GENI, where the switch capacity is set to $m \cdot |K|$.

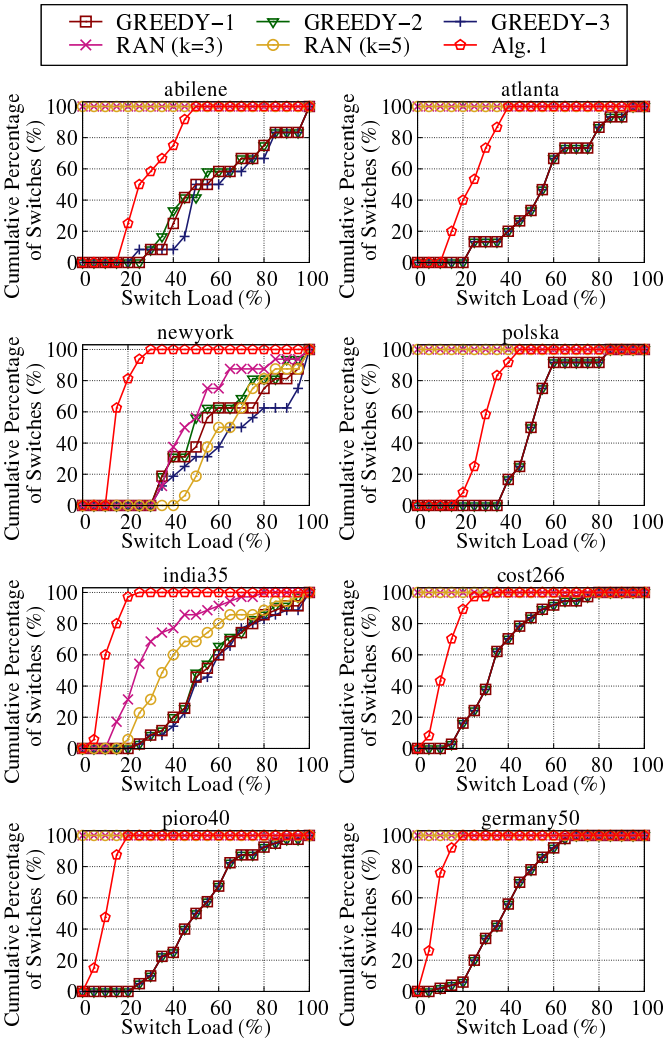

Fig. 6. The cumulative percentage of switches of switch load.

showed that our algorithm outperforms naive methods and an existing algorithm that assumes the forwarding path set is predetermined and does not address the demand constraints.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *ACM SIGCOMM*, 2008.

[2] U. Hoelzle, "Opening address: 2012 open network summit," *Date Retrieved*, 2012. [Online]. Available: http://www.opennetsummit.org/archives/apr12/hoelzle-tue-openflow.pdf

[3] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," Tech. Rep., 2000.

[4] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, 2015.

[5] S. Yasukawa, O. Komolafe, and A. Farrel, "An analysis of scaling issues in mpls-te core networks," 2009.

[6] R. Cohen, L. Lewin-Eytan, J. S. Naor *et al.*, "On the effect of forwarding table size on sdn network utilization," in *IEEE INFOCOM*, 2014.

[7] X.-N. Nguyen, D. Saucez, C. Barakat, and T. Turletti, "Officer: A general optimization framework for openflow rule allocation and endpoint policy enforcement," in *IEEE INFOCOM*, 2015.

[8] A. Gushchin, A. Walid, and A. Tang, "Enabling service function chaining through routing optimization in software defined networks," in *IEEE Annu. Allerton Conf. Commun. Control Comput.*, 2015.

[9] J. Wang, C. Qiao, and H. Yu, "On progressive network recovery after a major disruption," in *IEEE INFOCOM*, 2011.

[10] M. Chiesa, G. Kindler, and M. Schapira, "Traffic engineering with equal-cost-multipath: An algorithmic perspective," in *IEEE INFOCOM*, 2014.

[11] S. Paris, A. Destounis, L. Maggi, G. Paschos, and J. Leguay, "Controlling flow reconfigurations in sdn," in *IEEE INFOCOM*, 2016.

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

[13] E. Danna, A. Hassidim, H. Kaplan, A. Kumar, Y. Mansour, D. Raz, and M. Segalov, "Upward max min fairness," in *IEEE INFOCOM*, 2012.

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

[15] L. Liu, X. Cao, Y. Cheng, L. Du, W. Song, and Y. Wang, "Energy-efficient capacity optimization in wireless networks," in *IEEE INFOCOM*, 2014.

[16] J. He, X. Zhao, and B. Zhao, "Joint request routing and video adaptation in collaborative vod systems," in *IEEE WCNC*, 2013.

[17] Z. Cao, M. Kodialam, and T. Lakshman, "Traffic steering in software defined networks: planning and online routing," in *ACM SIGCOMM workshop on Distributed cloud computing*, 2014.

[18] N. Garg and J. Könemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," *SICOMP*, 2007.

[19] L. K. Fleischer, "Approximating fractional multicommodity flow independent of the number of commodities," *SIAM Journal on Discrete Mathematics*, 2000.

[20] G. Karakostas, "Faster approximation schemes for fractional multicommodity flow problems," *ACM TALG*, 2008.

[21] H. Farmanbar and H. Zhang, "Traffic engineering for software-defined radio access networks," in *IEEE NOMS*, 2014.

[22] "Geni: Global environment for network innovations." [Online]. Available: https://www.geni.net/

[23] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "Sndlib 1.0-survivable network design library," *Networks*, 2010. [Online]. Available: http://sndlib.zib.de

[24] M. R. Abbasi, A. Guleria, and M. S. Devi, "Traffic engineering in software defined networks: A survey," *Journal of Telecommunications and Information Technology*, 2016.

[25] R. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*. Plenum Press, 1972.

[26] J. Y. Yen, "An algorithm for finding shortest routes from all source nodes to a given destination in general networks," *Quarterly of Applied Mathematics*, 1970.

[27] "Iperf." [Online]. Available: https://iperf.fr