An efficient approximation algorithm for traffic engineering in software defined networks

Gang Wang*, Gang Feng*†, Shuang Qin*, Mu Yan*,
**National Key Laboratory of Science and Technology on Communications,
University of Electronic Science and Technology of China,
Chengdu, Sichuan, 611731, China
E-mail: fenggang@uestc.edu.cn*

Yantao Guo†,
*†Science and Technology on Information Transmission and Dissemination in Communication Networks Laboratory, Shijiazhuang, Hebei,50081, China ,
gyt6400@163.com*

In this paper, we focus on the efficient algorithm for solving the multi-commodity flow problems involved in TE and develop an improved approximation algorithm (i-FPTAS) based on the Fully Polynomial Time Approximation Scheme (FPTAS), with aim of greatly improving computational efficiency without compromising traffic load balancing performance. Numerical results shows that i-FPTAS can achieve close-optimal approximation solution which is much better than that of FPATS with the same approximate parameter, and increasing the approximate parameter of i-FPTAS can lower computational complexity with little loss on close-optimality.

Keywords: Traffic Engineering; multi-commodity flow; FPTAS.

1. Introduction

Traffic engineering (TE) deals with the issue of network performance optimization, involving multiple objectives, including maximizing throughput, balancing link utilization, by dynamically analyzing, predicting and routing traffic[1]. One of TE’s obvious advantages over the traditional shortest path based routing method is that it can exploit the multi-path diversity from source to destination[2]. TE is indeed a traditional open problem which has been studied for decades, while the application scenarios are mainly limited to the Internet [3]. In recent years, the emergence of Software Defined Networking (SDN) has infused new inspiration for TE. SDN centralizes the control and management functions of various network elements, thus network traffic can be processed and forwarded in the granularity of individual flows[4].

Agarwal et al. [5] design a TE scheme for evolutionary hybrid SDN architecture and formulate the SDN-controller's TE as an linear program to achieve network load balancing. Routing decision with middle boxes of a TE scheme is investigated in [6] under an SDN architecture. The proposed policy aware routing applies the service policy to select feasible paths which must pass through some middle boxes, and the authors formulate the offline planning problem to decide how much network capacity is need to meet the demand. Efficient algorithms that solve TE problems are an important part in the TE scheme of SDN-based networks. The logically centralized SDN controller has the global control over the network's data plane which consists of huge number of forwarding nodes and traffic flows between all the node pairs. At the same time, SDN needs to process flows in a finer granularity rather than the aggregated flows as in the traditional backbone network, resulting in very large number of input variables to the TE algorithm [4], [7].

In this paper, we formulate the SDN network TE problem, which is indeed a multi-commodity flow problem, as a linear program. In order to efficiently solve this problem, based on the Fully Polynomial Time Approximation Scheme (FPTAS), we propose an improved approximation algorithm called i-FPTAS. Compared with FPTAS with the same approximation ratio $(1+w)$, i-FPTAS can achieve a solution closer to the optimum solution. Furthermore, i-FPTAS is insensitive to the approximation ratio.

The rest of the paper is organized as follows. In Section 2, we formulate the SDN-based data plane TE problem as a multi-commodity flow problem. We then review the basics of FPTAS and elaborate the proposed approximation algorithm (i-FPTAS) in Section 3. Section 4 presents numerical results as well as discussions. We finally conclude the paper in Section 5.

2. Problem Formulation

We abstract the data plane of the SDN-enabled network as a directed graph $G(\mathcal{N},\mathcal{L})$, where $\mathcal{N}$ is the set of forwarding nodes, $\mathcal{L}$ is the set of links with link capacity $c(e)$. In the network, each service request is a flow $k \in \mathcal{K}$. A feasible path p_{uv} is a loop-free path with the source u and destination v. The set of feasible paths that may serve a flow $k \in \mathcal{K}$, is defined by $\mathcal{P}_k = \{p_{ij} | i = s(k),\ j = t(k)\}$, where $s(k)$ is the source of flow k, and $t(k)$ is the destination. Additionally, the set of feasible paths is defined as $\mathcal{P} = \bigcup_{k\in\mathcal{K}} \mathcal{P}_k$, and the set of paths that use edge e are defined as $\mathcal{P}_e = \{p \in \mathcal{P} | e \in p\}$, $e \in \mathcal{L}$.

Each commodity's demand $d(k)$ is known in advance by the network controller. Using the path-flow model, in order to achieve the objective of load balancing, i.e. minimizing the maximum link utilization, the TE problem can be formulated as

$$\begin{aligned} & min\ \theta, && (1)\\ & s.t.\ \sum\nolimits_{p\in\mathcal{P}_e} y(p) \le \theta c(e),\ \ \forall e \in \mathcal{L}, && (1.1)\\ & \quad \sum\nolimits_{p\in\mathcal{P}_k} y(p) \ge d(k),\ \ 1 \le k \le |\mathcal{K}|, && (1.2)\\ & \quad y(p) \ge 0,\ \ \forall p \in \mathcal{P}, && (1.3)\\ & \quad 0 \le \theta \le 1, && (1.4) \end{aligned}$$

where θ is the maximum link utilization ratio and $y(p)$ is the bandwidth allocated to path p. Note that (1.1) ensures no edge is allowed to be overloaded, and (1.2) guarantees each commodity's flow demand. The path-flow model for problems (1) has an exponential number of path variables, resulting in very high computational complexity. An alternative linear program using *edge-flow* model could be established, which can be solved by polynomial time algorithms. However, the problem scale of LP can be substantially large for a network with moderate or large size. In this case, even the polynomial time algorithm is infeasible. Besides, the solution derived from the edge-flow model may exist loops[8], which is undesirable for link resource allocation.

3. Efficient Approximation Algorithm for Traffic Engineering

To solve problem (1), it is converted to the maximum concurrent flow problem [9] by substitution of $\pi = 1/\theta\,, x(p) = y(p)/\theta$. Then FPTAS can be used to solve problem (1) to achieve an $(1+w)$-optimal solution, since FPTAS's complexity is the polynomial function of the number of edges. To achieve better approximation performance and computational efficiency, we design an improved algorithm of FPTAS (i-FPTAS), which is summarized as Algorithm 1.

By carefully analyzing the procedure and the output of FPTAS [9], we find that the optimal solution is located at the boundary of the feasible set, while there is a high probability that the solution of FPTAS is in the interior. This observation inspires us to develop i-FPTAS by fine-tuning the feasible solution to find another solution at the boundary of the feasible set, which is much closer to the optimal solution. We elaborate i-FPTAS by looking into the solution $x(p)$ of FPTAS. Since $x(p)$ is feasible, the constraints of problem (1) are satisfied, i.e.

$$\sum\nolimits_{p\in\mathcal{P}_e} x(p) \le c(e),\ \ \forall e \in \mathcal{L}, \tag{2}$$

$$\sum\nolimits_{p\in\mathcal{P}_k} x(p) \ge \pi d(k),\ \ \forall k \in \mathcal{K}. \tag{3}$$

where $\pi = \min_k \sum_{p\in\mathcal{P}_k} x(p)\,/d(k)$. Since some commodities are assigned excessive bandwidth on the path, which does not contribute to the increase of value π, we can get $\hat{x}(p)$ first by decreasing the value of some component $x(p)$ so as to make the constraint (3) become

$$\sum\nolimits_{p\in\mathcal{P}_k} \hat{x}(p) = \pi d(k), \ \ \forall k \in \mathcal{K}.$$

Because $\hat{x}(p) \le x(p)$, $\sum_{p\in\mathcal{P}_e} \hat{x}(p) \le c(e), \ \ \forall e \in \mathcal{E}$ still holds. Defining that

$$g = \min_{e\in\mathcal{L}} c(e) / \sum\nolimits_{p\in\mathcal{P}_e} \hat{x}(p),$$

then $g \ge 1$ always holds, and when $g > 1$, and there exists improvement for the solution. We let $\hat{x}(p) \leftarrow g\hat{x}(p)$. Thus, the objective value π is increased to $g\pi$.

Algorithm 1: (i-FPTAS) solve max-concurrent flow problem.

Input: network topology $G(\mathcal{N},\mathcal{L})$, link capacity $c(e)$,
commodity flow set $(\mathcal{K}, d(k))$, approximation parameter w.

Output: flow demand scaling factor π, flow variables on each edge $f_k(e)$.

```
1   Initialize: $\forall e \in \mathcal{L}, l(e) \leftarrow \delta/c(e), \forall e, k, f_k(e) \leftarrow 0$
2   while $D(l) < 1$, in phases $i = 1,2,\ldots$ do
3     for iterations $j = 1,2,\ldots,|S|$, if $D(l) < 1$, do
4       $\mathcal{K}(u_j) \leftarrow \{k | s(k) = u_j\}$
5       $d_{i,j}^k \leftarrow d(k), \ \forall k \in \mathcal{K}(u_j)$
6       for steps $s = 1,2,\ldots s_j$, if $D(l) < 1$, do
8         $\mathcal{k}(u_j) \leftarrow \{k | k \in \mathcal{K}(u_j), d_{i,j}^k > 0\}$
9         if $\mathcal{k}(u_j) = \Phi$   end steps.
10        $t_{i,j,s} \leftarrow$ shortest path tree serving flows $\mathcal{k}(u_j)$
11        $p_{i,j,s}^k \leftarrow$ shortest path serving flow $k \in \mathcal{k}(u_j)$
12        $\sigma \leftarrow \max\left\{1, \max_{e\in t_{i,j,s}} \left\{\sum_{k:e\in p_{i,j,s}^k} d_{i,j}^k / c(e)\right\}\right\}$
13        $\begin{cases} f_{i,j,s}^k \leftarrow d_{i,j}^k/\sigma \\ d_{i,j}^k \leftarrow d_{i,j}^k - f_{i,j,s}^k \end{cases}, \ \forall k \in \mathcal{k}(u_j)$
14        $f_k(e) \leftarrow f_k(e) + f_{i,j,s}^k, \ \forall e \in p_{i,j,s}^k, \forall k \in \mathcal{k}(u_j)$
15        $l(e) \leftarrow l(e)\left(1 + \varepsilon \cdot \sum_{k:e\in p_{i,j,s}^k} f_{i,j,s}^k / c(e)\right), \ \forall e \in t_{i,j,s}$
17      end for (steps)
18    end for (iterations)
19  end while (phases)
20  $f_k(e) \leftarrow f_k(e) / \log_{1+\varepsilon}\frac{1+\varepsilon}{\delta}, \ \forall e \in \mathcal{L}, \forall k \in \mathcal{K}$
21  $\pi_k \leftarrow \sum_{head(e)=s(k)} f_k(e)/d(k)$
22  $\pi \leftarrow \min_k \pi_k$
23  $f_k(e) \leftarrow f_k(e)/(\pi_k/\pi), \ \forall e \in \mathcal{E}, \forall k \in \mathcal{K}$
24  $g \leftarrow \min_e \{c(e)/\sum_{k\in\mathcal{K}} f_k(e)\}$
25  $f_k(e) \leftarrow g \cdot f_k(e), \pi \leftarrow g\pi$
```

4. Numerical Results

We comapre the performance of i-FPTAS, the originial FPTAS [14]. Since the objective of problem (1) is minimizing the maximum link utilization, to demonstrate the improvement on approximation performance of i-FPTAS, we define the *relative maximum link utilization* ($RMLU$) as

$$RMLU = \theta_{approx}/\theta_{opt},$$

where θ_{approx} and θ_{opt} represent the objective value of approximation and the optimal solution. The former is achieved by i-FPTAS, FPTAS. The latter is obtained by LP which could incur very high computational complexity for large size network. Obviously, $RMLU$ is always greater than 1, and the closer it is to 1, the better the approximation performance for problem (1).

The numerical experiment is carried out with random network with average degree of 3, with number of nodes ranging from 10 to 120 and each link is provisioned with adequate capacity. For each instance, 10 random traffic matrices are randomly generated, where the flows with randomly selected source/sink have demands between 1Mbps~10Mbps, and the number of flows originated from one node is proportional to the network size (the proportion to is set to 0.5). We first run linear program of problem (1) to determine the capacity of each link that is needed to accommodate all the tested traffic matrices.

Fig. 1 compares the $RMLU$ of FPTAS, i-FPTAS with two approximation parameters: $w = 5$ and 10. We can observe that the objective value of i-FPTAS is very close to the optimal value. In comparison, the objective value of FPTAS has a notable gap to the optimal value. Hence our proposed i-FPTAS has a significant improvement on approximation performance, especially when w is large(i.e. $w = 10$ comparing with $w = 5$). In addition, FPTAS with different approximation parameters achieves very different $RMLU$, while the difference is not apparent for i-FPTAS. Thus i-FPTAS is not sensitive to the approximation ratio, which implies that a nearly optimal solution can be achieved with a low computational complexity (*i.e.*, using a large w in the algorithm).

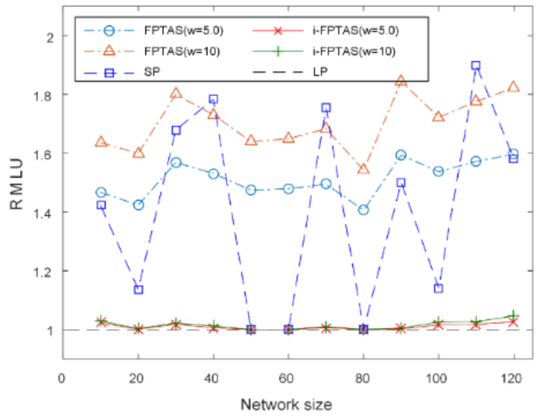

Fig. 1 Comparison of RMLU between basic i-FPTAS , FPTAS

depicts the running time of i-FPTAS with different approximation ratios, the optimal linear programming (LP) and the heuristic shortest path method (SP), as a function of the network size. Note that the running time of FPTAS is not presented here, since the difference of running time between FPTAS and i-FPTAS is negligible, which is partly shown in Table 1. As expected, a smaller approximation ratio (closer to optimal solution) of i-FPTAS results in longer running time. For a network of small size, say smaller than 30 nodes, the running time of i-FPTAS could be larger than that of LP. However, the running time of LP grows very rapidly with the network size and becomes larger than that of i-FPTAS soon. For example, for the network of 120 nodes, the running time of LP is larger than that of i-FPTAS(both $w = 5$ and 10) by approximately two orders of magnitude.

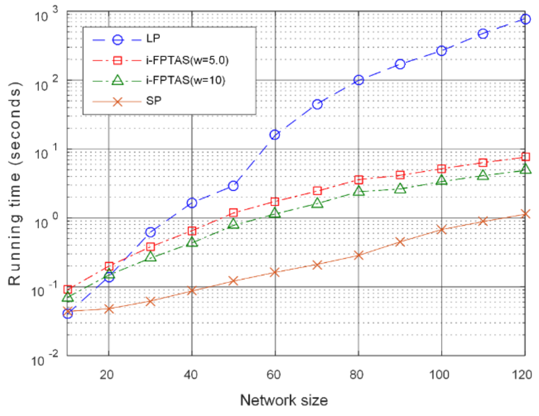

Fig. 2 Running time of i-FPTAS($w = 5$, 10), LP (the axis of running time is log-based)

Table 1. Comparison of Running Time (seconds) between FPTAS and i-FPTAS

# of nodes	20	40	60	80	100	120
FPTAS(w=5.0)	0.199	0.652	1.735	3.584	5.160	7.572
i-FPTAS(w=5.0)	0.199	0.652	1.737	3.590	5.170	7.590

From the results in Fig.1, we can see that i-FPTAS($w = 5$) and i-FPTAS($w = 10$) have almost the same close-optimal objective value, while the running time of i-FPTAS($w = 5$) is a bit larger than i-FPTAS($w = 10$). Therefore, a relatively large approximation ratio is adequate for i-FPTAS to obtain a very close-optimal solution, while the computational complexity remains almost the same with that of FPTAS.

5. Conclusions

In this paper, we have proposed an efficient approximation algorithm (i-FPTAS) for traffic engineering. Under the SDN-based network architecture, traffic engineering scheme on the controller can control the individual traffic flows to be flexibly routed through multiple path to improve the system throughput or reduce the load on the bottleneck link. The traffic engineering problem is formulated as a maximum concurrent flow problem, while solving it via linear programming is difficulty in terms of computational complexity. Through analysis and numerical results, we show that i-FPTAS can essentially provide improvement to the solution of FPTAS, and thus it can improve the scalability of SDN controller's TE scheme.

Acknowledgement

This work is supported by the open project grant of Science and Technology on Information Transmission and Dissemination in Communication Networks Laboratory (Grant number: KX152600017/ITD-U15008).

References

1. 1.D. Awduche, A. Chiu, A. Elwalid, et al., "Overview and Principles of Internet Traffic Engineering," RFC 3272, 2002.
2. 2.S. Singh, T. Das, and A. Jukan, "A Survey on Internet Multipath Routing and Provisioning," IEEE Commun. Surv. Tuts., vol. PP, no. 99, pp. 1–19, 2015.
3. 3.N. Wang, K. Hon Ho, G. Pavlou, et al., "An overview of routing optimization for internet traffic engineering," IEEE Commun. Surv. Tuts., vol. 10, no. 1, pp. 36–56, 2008.
4. 4.B. Astuto, A. Nunes, M. Mendonca, et al., "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," IEEE Commun. Surv. Tuts., vol. 16, no. 3, pp. 1617–1634, 2014.
5. 5.S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic Engineering in Software Defined Networks," in Proceedings of the IEEE INFOCOM, 2013, pp. 2211–2219.
6. 6.Z. Cao, M. Kodialam, and T. V. Lakshman, "Traffic Steering in Software Defined Networks: Planning and Online Routing," in Proceedings of the 2014 ACM SIGCOMM workshop on Distributed cloud computing (DCC'14), 2014, pp. 65–70.
7. 7.I. F. Akyildiz, A. Lee, P. Wang, et al., "A roadmap for traffic engineering in SDN-OpenFlow networks," Comput. Netw., vol. 71, pp. 1–30, 2014.
8. 8.R. G. Gallager, "Loops in multicommodity flows," in IEEE Conference on Decision and Control including the 16th Symposium on Adaptive Processes and A Special Symposium on Fuzzy Set Theory and Applications, 1977, vol. 16, pp. 819–825.
9. G. Karakostas, "Faster Approximation Schemes for Fractional Multicommodity Flow Problems," ACM Trans. Algorithms, vol. 4, no. 1, p. 17, 2008.