Incremental SDN-Enabled Switch Deployment for Hybrid Software-Defined Networks

Meitian Huang and Weifa Liang
Research School of Computer Science
The Australian National University, Canberra, ACT 2601, Australia
Email: u4700480@anu.edu.au, wliang@cs.anu.edu.au

***Abstract*—Software-defined networking (SDN) is a promising technique that has reshaped the landscape of network management. By providing simplified, cost-effective management, SDN has been envisioned as the next-generation network paradigm. However, due to economic, organizational, and technical challenges, replacing all conventional switches in current operational networks by SDN-enabled switches is impractical in the short term. It thus is desirable to deploy SDN-enabled switches into existing networks incrementally, and such a network consisting of SDN-enabled switches and conventional switches is referred to as a hybrid SDN network. The incremental deployment of SDN-enabled switches is challenging because the number of conventional switches that can be replaced is typically limited, due to budget constraints or operational network stability concerns, yet the impact of the deployment should be maximized. In this paper, we deal with the SDN-enabled switch placement problem with the aim to maximize system performance, given K switches to be replaced, for which we first propose heuristics by replacing conventional switches one by one iteratively. We then devise scalable algorithms that replace multiple switches, instead of a single switch, in each iteration. We finally evaluate the performance of the proposed algorithms based on real and synthetic network topologies. Experimental results demonstrate that the proposed algorithms are promising and exhibiting high scalability.**

***Index Terms*—Software-defined networking, hybrid networks, conventional networks, performance optimization.**

I. Introduction

The emerging software-defined networking (SDN) holds great promises to solve long-standing problems in conventional networks which lack automatic reconfiguration and response mechanisms that are critical to handling the dynamics of faults and reacting to load changes. Switches in conventional networks run distributed protocols and perform traffic routing independently. Each link is manually assigned a weight by the network operator, and the shortest paths between sources and destinations are computed in terms of the link weights. However, the link weight setting in conventional networks cannot be easily adjusted. The network resource is consequently often not efficiently utilized, and the performance of traffic engineering on such networks is limited. In contrast, the control plane in a software-defined network is decoupled from the data forwarding plane and located on an external controller that exerts a centralized control of the entire network. The centralized controller can flexibly route traffic flow between any pair of switches and split an arbitrary ratio of the flow to incident links of SDN-enabled switches. As a result, it has been envisioned that the deployment of SDN-enabled switches will provide a convenient and effective way to perform traffic engineering and can result in better network capacity utilization, thereby improving the delay and loss performance of the network [1], [8]. The advantages of SDN provide an incentive to the transition of SDN network. For instance, Microsoft and Google have achieved a high network utilization in a fully SDN-enabled inter-DC WAN by taking advantages of the flexibility of SDN [9], [11].

Although SDN has these advantages, a full deployment of SDN-enabled switches in an existing network remains impractical in the short term due to economic, organizational, and technical challenges. A full SDN deployment requires numerous modifications to current network architectures and error-prone replacements of conventional network equipment, which incurs significant cost and raises network operational stability concerns on enterprises and ISPs, especially in large scale enterprise networks. As a result, a full deployment of SDN, in which all switch nodes support SDN and can be centrally controlled by the controllers, cannot be easily achieved in the short term, and enterprises and network operators must resort to incrementally deploying SDN-enabled switches into their existing networks.

The problem of incremental SDN deployment poses great challenges. Due to the limited budget and operational stability requirement of a network, only a certain number of conventional switches can be replaced by the SDN-enabled switches at each time. Thus, given the number of conventional switches to be replaced by SDN-enabled switches in a network, determining which conventional switches to be replaced while maximizing the network performance is critical. This paper tackles this challenging problem by developing novel strategies that explore the finest tradeoff between the accuracy of solutions and the running time of obtaining the solutions.

The main contributions of this paper are summarized as follows. We consider the SDN-enabled switch placement problem in which SDN-enabled switches are strategically placed into existing conventional networks. We first propose two heuristics based on a novel ranking metric. We then develop two heuristics that scale better in large networks by exploring the non-trivial tradeoff between the solution accuracy and the running time of obtaining the solution. We finally evaluate the performance of the proposed algorithms through simulations, based on real and synthetic network topologies. Experimental

978-1-5090-2991-4/17/$31.00 ©2017 IEEE

results demonstrate that the proposed algorithms are very promising.

The rest of the paper is organized as follows. Section II will review prior related studies. Section III will introduce the notations and notions, and formally define the problem. Section IV will devise scalable strategies for incremental switch deployment, while Section V will propose improved heuristics for the problem. Section VI will present the experimental results of performance evaluation of the proposed algorithms. Section VII will conclude the paper.

II. Related work

A closely related study on SDN-enabled switch placement is due to Agarwal *et al.* [1], they were the first to investigate the problem in SDN. They adopted a greedy approach by replacing switches one by one iteratively. At each iteration, a switch that results in the greatest increase in throughput is replaced by an SDN-enabled switch. This process is repeated until a given number of SDN-enabled switches has been replaced. However, later experimental results will demonstrate that their approach has poor scalability. Similarly, Levin *et al.* [13] provided an architecture that implements an SDN control plane in a network that consists of both conventional switches and SDN-enabled switches that can be incrementally deployed. However, their work focused on the system implementation of the architecture and did not address the problem of determining which switches should be replaced. Furthermore, several studies tackled a variety of traffic engineering aspects of network performance and reaped the benefit of partial SDN deployment. For instance, Guo *et al.* [8] devised a novel algorithm that can minimize the maximum link utilization of a hybrid network, where the OSPF weights and flow splitting ratio of the SDN nodes can both be changed, and the flows coming into SDN-enabled switches can be arbitrarily split. It is however assumed that the locations and number of the SDN nodes are given as *a priori*. Hu *et al.* [10] also studied traffic engineering in hybrid SDN networks by finding a maximum flow by only tuning the forwarding behaviors of the SDN devices, through formulating the maximum flow problem in networks with partial SDN deployments and developing a fast Fully Polynomial Time Approximation Scheme (FPTAS) for it. Wang *et al.* [16] investigated how to reduce energy consumption in hybrid SDNs and minimize energy cost by finding the minimum-power network subsets and shutting down other unnecessary switches and links. All mentioned studies assumed that SDN-enabled switches have already been placed in the network.

On the other hand, the other studies all determine which conventional switches should be replaced by SDN-enabled switches. Guo *et al.* [8] explored traffic engineering in a hybrid network. Specifically, they proposed a novel algorithm to minimize the maximum link utilization. Caria *et al.* [4] uses a divide and conquer method to partition the OSPF domain to sub-domains by selecting satiable SDN nodes for traffic engineering. Furthermore, several metrics, e.g., [5], could be used to evaluate the rank of a node in a graph. However, they were typically based on the topological structure of the graph and often did not take into account the traffic matrix. Thus they are not suitable for the problem of concern.

III. Preliminaries

In this section, we first introduce the system model as well as notations and notions. We finally define the problem.

A. System model

We consider a network represented by an undirected graph $G = (V, E)$, where V is the set of conventional switches and E is the set of links between switches. Each conventional switch $v \in V$ can be potentially replaced by an SDN-enabled switch, and each link $e \in E$ has a workload of u_e, which is the ratio of actual bandwidth usage to its capacity. We further assume that a traffic matrix T in G is given, where each entry $T_{i,j}$ is the traffic rate between $v_i \in V$ and $v_j \in V$.

B. Deployment of SDN-enabled switches

The operator of a network G may be interested in deploying some SDN-enabled switches into G to experiment the SDN technology. However, in order to meet the budget constraint and ensure a stable operation of the network, the number of conventional switches to be replaced is often limited. It has been shown that hybrid SDNs can achieve fast recovery if flows pass through at least one SDN-enabled switch [6], and it is possible to operate the entire network if the path between a source and a destination includes at least one SDN-enabled switch [13]. Consequently, the goal of incremental SDN-enabled switch deployments is to maximize the fraction of flow routed through SDN-enabled switches by replacing some conventional switches with SDN-enabled switches subject to a budget constraint. Formally, we assume that the number of conventional switches to be replaced is K. After K SDN-enabled switches replace K conventional switches in G, a fraction of traffic in traffic matrix T will be routed through SDN-enabled switches. We denote such fraction as λ, clearly $0 \leq \lambda \leq 1$.

C. Problem definition

Given a network $G = (V, E)$, the number of conventional switches to be replaced K, and a traffic matrix T, *the SDN-enabled switch placement problem in* G is to replace K conventional switches by K SDN-enabled switches such that λ is maximized.

IV. Efficient Algorithms

Since the SDN-enabled switch placement problem is NP-hard [1], we here propose fast yet scalable algorithms for it. The basic idea of the proposed algorithms is to replace conventional switches through which the traffic is most likely to pass with SDN-enabled switches so that the impact of switch replacement on λ is maximized. We thus first introduce a metric that accurately captures the impact of replacing a conventional switch. We then propose a heuristic and an improved algorithm based on it.

A. Ranking conventional switches

The rank of a conventional switch should accurately capture its contribution to λ when it is replaced by an SDN-enabled switch. We have the following important observations. Firstly, the degree of a switch should exhibit high flexibility so that the switch can direct flows to its incident links. In other words, if a switch node has more incident links than others, then it will contribute to λ better than others, because an incoming flow can be allocated to its incident links. Secondly, the workload of a link is also an important consideration, because if the link has already been overloaded, it can accommodate less flow from other switch nodes. Accordingly, each conventional switch $v \in V$ is assigned a rank $rank(v)$ as follows.

$$rank(v) = \alpha \cdot \frac{1}{\sum_{e \in E_v} u_e} + \beta \cdot |P_v| \tag{1}$$

where $|P_v|$ is the number of shortest paths between distinct pairs of switches in the traffic matrix T passing through v, and E_v is the set of incident links of v, u_e is the workload of an incident link from v, and α and β are tuning parameters with $\alpha + \beta = 1$. Intuitively, this ranking metric accounts for both the traffic entering and leaving node v through its incident links.

B. A heuristic

Having defined the ranking metric, a greedy algorithm then proceeds iteratively. Within each iteration, a switch with the highest rank will be replaced. As switch nodes with higher ranks are more likely to have greater impacts on the fraction of flows passing through SDN-enabled switches, the algorithm selects the top K switches to be replaced by SDN-enabled switches. The detailed algorithm is given in Algorithm 1.

Algorithm 1 A basic heuristic for the SDN-enabled switch placement problem

Input: an SDN $G = (V, E)$, the number of conventional switches to be replaced K, and a traffic matrix T;
Output: K conventional switches that should be replaced by SDN-enabled switches;

for each pair of switch nodes u and v with traffic rate $T_{u,v}$ **do**
  Compute the shortest path $P_{u,v}$ in G between u and v, and label links and switch nodes on $P_{u,v}$;
end for
for each switch $v \in V$ **do**
  Compute its rank $rank(v)$ according to Eq. (1);
end for
return K highest ranked switch nodes by sorting the switch nodes in non-decreasing order by their ranks and taking the first K switch nodes.

We now analyze the time complexity of the algorithm as follows.

Theorem 1: Given a network $G = (V, E)$ and the number of conventional switches to be replaced K, there exists an algorithm, Algorithm 1, that delivers a feasible solution in $O(|V|^3)$ time.

Proof: Clearly, the calculation of ranks of switches in G depends on the information about all-pairs shortest paths in G, which takes $O(|V|^3)$ time. Thus, Algorithm 1 takes $O(|V|^3)$ time as its running time is dominated by the calculation of all pairs shortest paths in G. ■

C. Improved algorithm

Algorithm 1 has poor scalability, and it is painstaking for large-scale networks. To develop a fast yet scalable algorithm, we propose another algorithm that delivers comparable solutions in much less running time, by approximately estimating the ranks of switches. That is, instead of finding a shortest path between every pair of switches, a subset of shortest paths between switches will be randomly chosen, which will be used as an approximate estimation of the ranks of switches.

Specifically, we randomly choose $|U|$ switches from V with $|U| \ll |V|$, and construct a shortest path tree rooted at each switch $v \in U$ by Dijkstra's algorithm. Thus $|U||V|$ pairs shortest paths can be identified. In the end, each switch has an approximate rank. The improved algorithm proceeds to replace the K switches with the top-K ranks. The detailed algorithm is given in Algorithm 2.

Algorithm 2 A faster heuristic for the SDN-enabled switches placement problem based on approximate ranks

Input: a network $G = (V, E)$, the number of conventional switches to be placed K, the number of iterations L, and $|U|$ the number of switches to be chosen;
Output: K conventional switches that should be replaced by SDN-enabled switches;

$i \leftarrow 0$; /* the number of iterations */
$V' \leftarrow V$;
for each $v \in V$ **do**
  $rank(v) \leftarrow 0$;
end for
while $i \neq L$ and $V' \neq \emptyset$ **do**
  Randomly choose $|U|$ switches from V' and put them in V'_i;
  for each switch $v \in V'_i$ **do**
    Find a shortest tree in G rooted at v using Dijkstra's algorithm;
    Increase the ranks of the switches in these shortest paths starting from v;
    $V' \leftarrow V' \setminus V'_i$;
  end for
  $i \leftarrow i + 1$;
end while
Sort all switches in non-decreasing order of their ranks;
return the top K switches with highest ranks.

Theorem 2: Given a network $G = (V, E)$, the number of conventional switches to be replaced K, and the number of iterations L, the improved algorithm, Algorithm 2, takes $O(L|U||E|+|U||V|\log|V|)$ time to deliver a feasible solution, where $|U|$ is the number of switches to be chosen within each iteration and $|U| = \lfloor |V|/L \rfloor \ll |V|$.

Proof: The running time of Algorithm 2 is analyzed as follows. There are L iterations and each iteration involves running Dijkstra's algorithm $|U|$ times. Therefore, the overall running time is $O(L|U||E| + L|U||V|\log|V|) = O(|U||E| + |U||V|\log|V|)$. In particular, the running time is $O(|U||E| + |U||V|\log|V|) = O(|V|^{1/2}|E| + |V|^{3/2}\log|V|)$ when $|U| = \sqrt{|V|}$. In the rest, we assume that $|U| = \sqrt{|V|}$. We then analyze the probability of a switch node being chosen. If it is chosen in the first iteration, then the probability of it

being chosen is $|U|/|V| = 1/|V|^{1/2}$, otherwise, if it is chosen in the second iteration, the probability of a switch node being chosen is $|U|/(|V|-|U|) \approx 1/|V|^{1/2}$. If it is chosen in the Lth iteration, the probability of it being chosen is approximately $\frac{1}{\sqrt{|V|}-L+1}$. Accordingly, we can calculate the probability of an approximate rank of each switch, compared with the exact rank of the node. In a network with $|V|$ nodes, the total number of shortest paths between any pair of these nodes is $|V|(|V|-1)$. Algorithm 2 only identifies $L|U||V|$ pairs of shortest paths. Thus, the probability of a shortest path between two switches being selected by Algorithm 2 is $\frac{L|U||V|}{|V|^2} = \frac{L}{\sqrt{|V|}}$. When L is sufficiently large, this probability is very high, resulting in accurate approximations of node ranks. ■

V. Batch-deployment Heuristics

A. Basic idea

Compared with the proposed algorithms in the previous section that place SDN-enabled switches one by one, we here place multiple switches instead of a single one within each iteration. This method takes more time yet delivers a better solution than the algorithms devised in the previous section.

B. Heuristic with fixed numbers of switch placement

Specifically, the proposed algorithm proceeds iteratively. Within each iteration, $\lfloor\sqrt{K}\rfloor$ SDN-enabled switches will be placed. The key then is to determine which $\lfloor\sqrt{K}\rfloor$ locations to be placed. To this end, we adopt the proposed algorithms in the previous section to determine these $\lfloor\sqrt{K}\rfloor$ locations. One important issue is that the fractional demands of each pair of nodes that has been routed using SDN-enabled switches will be removed from consideration, associated with the workload of each link will also be considered. The detailed algorithm is given in Algorithm 3.

Algorithm 3 A heuristic for the SDN-enabled switch placement problem based on replacing a fixed number of switches

Input: a network $G = (V, E)$, the number of conventional switches to be replaced K, and the number of iterations L;
Output: K conventional switches that should be replaced by SDN-enabled switches;
1: $i \leftarrow 0$; /* the number of iterations */
2: $G_0 \leftarrow G$;
3: $SV \leftarrow \emptyset$ /* the placed SDN-enabled switches and their locations */
4: **while** $i \neq \lceil\sqrt{K}\rceil$ **do**
5: Choose the top-$\lfloor\sqrt{K}\rfloor$ nodes in G_i, by invoking `Algorithm 1` $(G_i, \sqrt{K}, V \setminus SV)$, and denote the result by U_i;
6: $SV \leftarrow SV \cup U_i$;
7: $i \leftarrow i + 1$;
8: Recalculate the demands of each switch and link with the set SV of SDN-enabled switches by invoking the multi-commodity flow algorithm due to [7];
9: Let G_i be the resulting graph after removing the fractional demands of each pair that has been routed using SDN-enabled switches;
10: **end while**
11: **return** SV.

Theorem 3: Given a network $G = (V, E)$ and the number of conventional switches to be replaced K, there exists an algorithm, Algorithm 3, that delivers a feasible solution to the SDN-enabled switch replacement problem in $O(K^{5/4}\sqrt{|V|}|E| + K^{5/4}|V|\log|V|)$ time.

Proof: The time complexity of Algorithm 3 is as follows. There are $\lceil\sqrt{K}\rceil$ iterations. In the i-th iteration, Algorithm 3 constructs an auxiliary graph $G_i = (V_i, E_i)$, where V_i is the set of switches that have not been replaced and E_i is the set of links that connect the switches in V_i. The number of conventional switches in G_i is $|V| - (i-1)\lfloor\sqrt{K}\rfloor$ while and the number of links in G_i is $O(|E|)$. The dominant running time of each iteration lies in the computation of the multi-commodity flow. Accordingly, the running time of the i-th iteration is $O(\sqrt{|V_i|}|E| + |V_i|^{3/2}\log|V_i|)$, i.e., $O(\sqrt{|V| - (i-1)\lfloor\sqrt{K}\rfloor}|E| + (|V| - (i-1)\lfloor\sqrt{K}\rfloor)^{3/2}\log(|V| - (i-1)\lfloor\sqrt{K}\rfloor))$. Consequently, the overall running time of Algorithm 3 is $O(K^{5/4}\sqrt{|V|}|E| + K^{5/4}|V|\log|V|)$. ■

C. Heuristic with geometric numbers of switch placement

Instead of placing the same number of switches in each iteration of Algorithm 3, we place a geometric number of switches. The reason behind this is that the very first several switch placements are crucial, we replace fewer switches in the very first few iterations. On the other hand, the impact of later replacement on the performance is less significant. Thus, more switches can be replaced. As a result, the number of iterations required to place K SDN-enabled switches in the logarithm of K. The detailed algorithm is given in Algorithm 4.

Algorithm 4 A heuristic for the SDN-enabled switch placement problem based on geometric sequences

Input: a network $G = (V, E)$, the number of conventional switches to be replaced K, and the number of iterations L;
Output: K conventional switches that should be replaced by SDN-enabled switches;
1: $i \leftarrow 0$; /* the number of iterations */
2: $G_0 \leftarrow G$;
3: $SV \leftarrow \emptyset$ /* the placed SDN-enabled switches and their locations */
4: **while** $|SV| \neq K$ **do**
5: $k_i \leftarrow 2^i$ /* the number of conventional switches to be placed */;
6: call the procedure to choose the top-k_i nodes in G_i, by invoking `Algorithm 1`$(G_i, k_i, V \setminus SV)$.
7: Let $V_i \subset V \setminus SV$ be the set with $|V_i| = k_i$;
8: $SV \leftarrow SV \cup V_i$;
9: $i \leftarrow i + 1$;
10: Recalculate the demands of each switch and link with the set SV of SDN-enabled switches by invoking the multi-commodity flow algorithm due to [7];
11: **end while**
12: **return** SV.

Theorem 4: Given a network G, the number of conventional switches to be replaced K, there exists an algorithm, Algorithm 4, that delivers a feasible solution to the SDN-enabled switch placement problem in $O(\sqrt{|V|}|E|\log K + |V|^{3/2}\log|V|\log K)$ time.

Proof: Following Algorithm 4, the number of conventional switches placed in the first i iterations is $2^0+2^1+\cdots+2^i = O(2^i)$ with $0 \le i \le \lceil \log K \rceil$. The number of iterations is thus $\lceil \log K \rceil$. Accordingly, the overall running time of Algorithm 3 is $O(\sqrt{|V|}|E| \log K + |V|^{3/2} \log |V| \log K)$. ■

VI. Performance Evaluation

In this section, we evaluate the performance of the proposed algorithms in both real and synthetic networks.

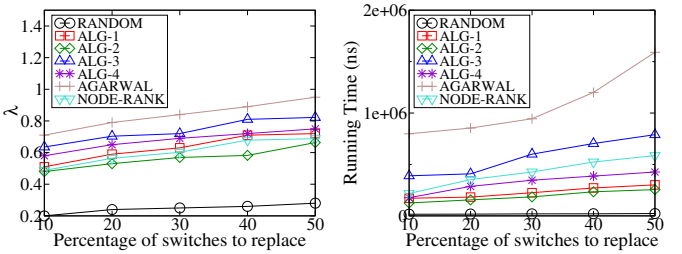
A. Experimental settings

We adopt commonly used, real network topologies including GÉANT [14] and ISP networks [15] in our simulations, where GÉANT is a European network consisting of 40 nodes and 122 links. These network topologies are chosen as representatives of real networks that institutions and businesses own and are interested in upgrading to SDN-enabled networks. We also use synthetic network topologies to study the scalability of the proposed algorithms. The bandwidth of each Internet link varies from 1,000 Mbps to 10,000 Mbps [12]. The probability that there is a flow between two switch nodes in the network is 0.2, and the demand of such a flow is randomly drawn from 10 to 120 Mbps [2]. We compare the performance of the proposed algorithms against a baseline algorithm `RANDOM` that randomly replaces K conventional switches in V and an algorithm `NODE-RANK` that replaces K switches with the top-K highest ranks according to the metric due to Cheng *et al.* [5]. We also implement the algorithm due to Agarwal *et al.* [1], referred to as `AGARWAL`. The proposed algorithms are referred to as `ALG-1`, `ALG-2`, `ALG-3`, and `ALG-4`, respectively. The running time is obtained based on a machine with a 3.40GHz Intel i7 Quad-core CPU and 16 GiB RAM. The accuracy parameter for the multi-commodity flow algorithm ϵ is 0.01. α and β in Equation (1) are both set to 0.5. The number of iterations L in the proposed algorithms is set to the square root of the number of conventional switches in the network. Unless otherwise specified, these parameters will be adopted in the default setting. Each value in figures is the mean of the results of 30 trials.

B. Performance evaluation

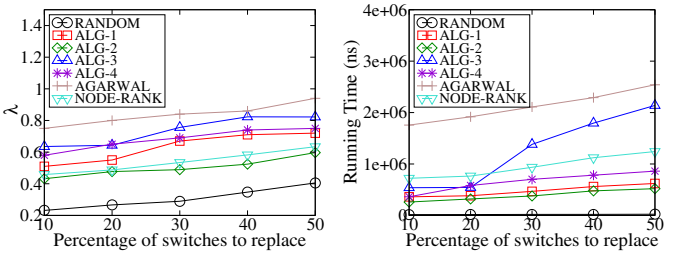
We compare the proposed algorithms against the baseline algorithm in GÉANT and several ISP networks by varying the number of conventional switches to be replaced K from 10 percent to 50 percent of the switches.

We first evaluate the performance of different algorithms in GÉANT in Fig. 1. We can see from Fig. 1 (a) that the greater the number of conventional switches to be replaced, the larger the value of λ. Although `AGARWAL` can achieve the highest λ among all algorithms, its running time is much slower than the proposed algorithms, which can achieve comparable results. When replacing 10 percent of the conventional switches by SDN-enabled switches, the λ values achieved by all proposed heuristics are 90% of that achieved by `AGARWAL` (0.68 versus 0.72), while the running time of the proposed heuristics is a half of that of `AGARWAL` (0.4×10^6ns versus 0.8×10^6ns). `RANDOM` is the worst among all algorithms. It is only able to achieve a λ value less than 0.3. Meanwhile, algorithm `NODE-RANK` outperforms than `RANDOM` yet is worse than the other heuristics, because it only takes the topological structure of the network into account and does not consider the given traffic matrix. It is worth noting that the running time of the proposed algorithm is significantly less than that of `AGARWAL`, while achieving comparable performance as `AGARWAL` does.



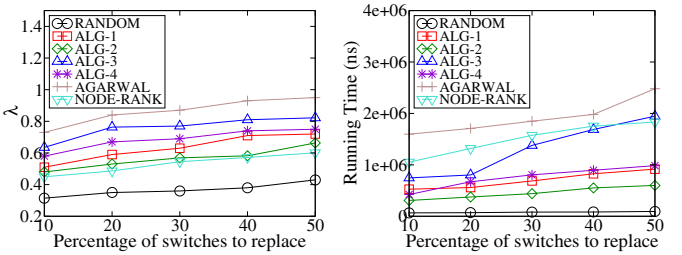
(a) The values of λ realized by different algorithms (b) Running time of different algorithms in nanoseconds

Fig. 1. Performance of different algorithms in the GÉANT network



(a) The values of λ realized by different algorithms (b) Running time of different algorithms in nanoseconds

Fig. 2. Performance of different algorithms in AS-4755

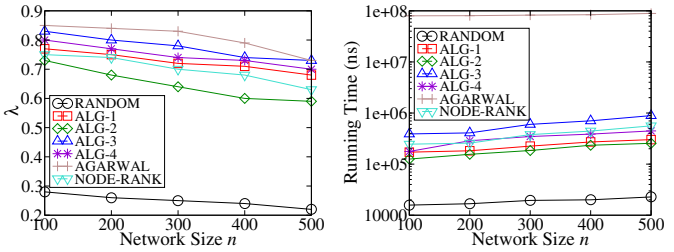


(a) The values of λ realized by different algorithms (b) Running time of different algorithms in nanoseconds

Fig. 3. Performance of different algorithms in AS-1755

We then compare different algorithms in larger, real networks AS-4755 and AS-1755, and the results for these two topologies are shown in Fig. 2 and Fig. 3, respectively. It must be reiterated that although `AGARWAL` achieves the highest value of λ among all algorithms, the gap between `AGARWAL` and the proposed algorithms is narrow and `AGARWAL` runs much slower than the proposed heuristics. The difference between `AGARWAL` and `ALG-3` is less than 10% as we observe in both Fig. 2 (a) and Fig. 3 (a), yet `AGARWAL` usually takes twice as much time as `ALG-3` does to deliver solutions. Specifically, when the percentage of switches to be replaced is 10%, the performance of `ALG-3` is 90% of that of `AGARWAL` (0.65 versus 0.7), while delivering results in less than a half of the time compared to `AGARWAL` (0.8×10^6ns

versus 1.8×10^6ns). In both networks, the result delivered by NODE-RANK is on a par with that delivered by ALG-4. However, NODE-RANK has much higher running time due to more time consuming metric calculations. Thus, the inferior performance and higher running time of NODE-RANK compared to the proposed heuristics illustrate the fact that a metric that focuses on the topology of a network is insufficient for the problem of concern, and a new metric is needed to solve the problem. Moreover, because both AS-4755 and AS-1755 have more switches, randomly replacing a given number of conventional switches by SDN-enabled switches is more likely to cause more flows passing through the newly placed SDN-enabled switches. This is the reason why the performance of RANDOM in these two AS topologies is better than that in GÉANT. Fig. 2 (b) and Fig. 3 (b) the running time of different algorithms in these topologies, respectively. In both topologies, AGARWAL is the slowest while the proposed algorithms run much faster. The running time of RANDOM is low, regardless of the network size because it simply returns K switches.



(a) The values of λ realized by different algorithms

(b) Running time of different algorithms in nanoseconds on a logarithmic scale

Fig. 4. Performance of different algorithms in synthetic networks

As these topologies have limited sizes, we adopt the widely used Barabási-Albert model [3] to generate networks of different sizes. Specifically, we vary the number of switches in a network from 100 to 500, and the results are depicted in Fig. 4. We can see from Fig. 4 (a) that AGARWAL achieves the highest value of λ among all algorithms in the experiments, yet the proposed algorithms, especially ALG-3, achieve similar performance as AGARWAL, but runs several orders of magnitudes faster. Namely, when the number of switches is 500, the value of λ realized by ALG-3 is approximately identical to that by AGARWAL (both at 0.72), yet the running time of ALG-3 is a fraction of that of AGARWAL, it is only 0.1% of the running time of AGARWAL (0.4×10^6ns versus 0.8×10^8ns). In addition, we can see that with the increase in the network size, the value of λ realized by all algorithms decreases. The reason behind is that the number of conventional switches to be replaced, i.e., the value of K, is set to the square root of the number of conventional switches. Thus, when the network size is 100, the value of K is 10, and when the network size is 500, the value of K is approximately 22. Even though the network size increases by five times, the number of conventional switches to be replaced only doubles. Thus, the growth in the number of conventional switches in a network outpaces the number of conventional switches to be replaced. This necessitates the need for a strategy capable of replacing existing conventional switches in a large scale network in a way such that the impact of the placed SDN-enabled switches is maximized. The running time of different algorithms is shown in a logarithmic scale in Fig. 4 (b). Due to the larger network sizes, the performance of AGARWAL worsens significantly and is an order of magnitude slower than other algorithms. Specifically, when the network size is 500, AGARWAL is hundreds of times slower than the proposed algorithm. Thus, although AGARWAL has slightly superior performance, it does not scale well to large networks. Furthermore, both RANDOM and NODE-RANK suffer from poor performance because they fail to take either the topological structure of the network or the traffic matrix into account, thereby delivering inferior solutions.

VII. Conclusion

In this paper, we considered the SDN-enabled switch placement problem to form hybrid SDN networks. We explored the solution accuracy and the running time to obtain the solutions through striving for a non-trivial tradeoff. We first proposed two algorithms based on a novel ranking metric and estimating the rank of each switch. We also developed two heuristics by exploring the finest tradeoff between the solution accuracy and the running time of obtaining the solution. We finally evaluated the performance of the proposed algorithms through simulations. Experimental results demonstrated that the proposed algorithms are scalable and very promising.

References

[1] S. Agarwal, M. Kodialam, and T.V. Lakshman. Traffic engineering in software defined networks. *Proc. of INFOCOM*, IEEE, 2013.

[2] Amazon Web Services, Inc. Amazon ec2 instance configuration. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-ec2-config.html.

[3] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, Vol. 286, pp. 509–512, 1999.

[4] M. Caria, T. Das, and A. Jukan. Divide and conquer: partitioning OSPF networks with sdn. *International Integrated Network Management*, IFIP/IEEE, 2015.

[5] X. Cheng, S. Su, Z. Zhang, *et al.* Virtual network embedding through topology-aware node ranking. *SIGCOMM Computer Communication Review*, Vol. 41, pp. 38–47, ACM, 2011.

[6] C.-Y. Chu, *et al.* Congestion-aware single link failure recovery in hybrid sdn networks. *Proc. of INFOCOM*, IEEE, 2015.

[7] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *Journal on Computing*, Vol. 37, pp. 630-652, SIAM, 2007.

[8] Y. Guo, Z. Wang, X. Yin, X. Shi, and J. Wu. Traffic engineering in sdn/ospf hybrid network. *Proc. of ICNP*, IEEE, 2014.

[9] C. Hong *et al.* Achieving high utilization with software-driven WAN. *Proc. of SIGCOMM*, ACM, 2013.

[10] Y. Hu *et al.* Maximizing network utilization in hybrid software-defined networks. *Proc. of GLOBECOM*, IEEE, 2015.

[11] S. Jain *et al.* B4: experience with a globally-deployed software defined WAN. *Proc. of SIGCOMM*, ACM, 2013.

[12] S. Knight *et al.* The internet topology zoo. *J. Selected Areas in Communications*, Volume 29, pp. 1765–1775, IEEE, 2011.

[13] D. Levin, M. Canini, S. Schmid, F. Schaffert, and A. Feldmann. Panopticon: reaping the benefits of incremental sdn deployment in enterprise networks. *Proc. of ATC*, USENIX, 2014.

[14] J. Martins *et al.* ClickOS and the art of network function virtualization. *Proc. of NSDI*, 2014.

[15] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with rocketfuel. *Proc. of SIGCOMM*, ACM, 2002.

[16] H. Wang *et al.* Saving energy in partially deployed software defined networks. *Trans. on Computers*, Vol. 65, no. 5, pp 1578–1592, IEEE, 2016.