Incremental Deployment for Traffic Engineering in Hybrid SDN Network

Yingya Guo*‡, Zhiliang Wang†‡, Xia Yin*‡, Xingang Shi†‡, Jianping Wu*‡, and Han Zhang*‡
*Department of Computer Science and Technology, Tsinghua University
†Institute for Network Sciences and Cyberspace, Tsinghua University
‡Tsinghua National Laboratory for Information Science and Technology (TNLIST)
Email:{guoyingya, wzl, zhanghan}@csnet1.cs.tsinghua.edu.cn, yxia@tsinghua.edu.cn,{shixg, jianping}@cernet.edu.cn

***Abstract*—Traffic engineering is a method to balance the flows and optimize the routing in the network. Software defined networking is a new network architecture and we can gain great benefit by migrating the traditional IP network to the SDN-enabled network from the perspective of traffic engineering. However, due to the economical, organizational and technical challenges, migrating to the network with a full deployment of SDN routers is impractical in the short term. It is a desirable choice to deploy SDN incrementally.**

In this paper, we seek to search for an optimal migration sequence of the legacy routers to SDN-enabled routers so that we can decide where and how many routers to migrate firstly. Our main contribution is that we propose a heuristic algorithm, i.e., genetic algorithm, to seek a migration sequence of the routers that obtains the most of the benefit from the perspective of traffic engineering. We evaluate the algorithm by conducting simulation experiments, making comparison to the greedy migration algorithm and static migration algorithms that we propose. The experiments exhibit that the genetic algorithm, outperforms the other migration algorithms in searching for a migration sequence. When properly deployed, about a migration of 40% of routers reaps most of the benefit.

***Index Terms*—traffic engineering; software defined networking; migration sequence; genetic algorithm**

I. Introduction

Traffic engineering has been extensively studied for a few decades. The goal of traffic engineering is to make the most of network resources to balance the flows and optimize the routing for traffic delivery. Generally speaking, minimizing the maximum link utilization and the cost are the two common objective functions to optimize in traffic engineering. In traditional IP network, all the routers run distributed protocols, such as Open Shortest Path First (OSPF) protocol. The traffic is routed along the shortest (least cost) path with each link assigned a weight (or cost). However, the link weight setting of OSPF protocol in the network cannot be changed constantly for stability. Therefore, the network resource is not fully utilized and the performance of traffic engineering is limited under this circumstance.

The emergence of software defined networking provides an efficient method to route the traffic flexibly in the network regardless of the weight setting of the links. Software Defined Networking (SDN) is a new network architecture that exerts a centralized control of the entire network. In this new architecture, the control plane is decoupled from the packet forwarding devices and located on the external controller. We can flexibly split arbitrary ratio of the flow to the outgoing links of the SDN-enabled routers through the centralized controllers. In this method, we can easily improve the performance of traffic engineering to a great extent and achieve the optimal routing with the integration of SDN in the legacy network. Google [1] and Microsoft [2] have already built the inter-connecting data centers which are fully SDN-enabled networks. They improve the network utilization on a large scale by migrating the legacy network to the fully SDN-enabled network.

Unlike data center (DC) networks or Inter-DC WAN (Wide Area Networks) in a large ICP (Internet Content Provider) like Google and Microsoft, migrating to the fully SDN-enabled network is by no means an easy task for Internet Service Provider (ISP) network. There will exist economical, organizational and technical challenges [3]. We should consider the incremental deployment of SDN routers in an ISP network. Hence, it is a wise choice to migrate some legacy routers to SDN-enabled routers firstly. On one hand, we cannot fully utilize the potential of network resources in traffic engineering with too few routers migrated; on the other hand, we have to invest more budgets to migrate more routers, which may make little gain for traffic engineering. Due to the resources and budgets constraints, we should make a trade off between the performance of traffic engineering and the investment. Therefore, it is necessary to solve the optimization problem of determining both the locations and the number of routers to migrate. We get down to solving this problem by searching for an optimal migration sequence, so that we can keep the migrated routers unchanged in the following migration periods and maintain the stability of the network.

In this paper, we focus on the hybrid SDN network scenario and discuss about the migration algorithms from the perspective of traffic engineering in an ISP network. Our work mainly concentrates on searching for an optimized migration sequence of the routers in the traditional IP network to minimize the maximum link utilization. To the best of our knowledge, we are original in searching for a migration sequence that minimizes the maximum link utilization exploiting the genetic algorithm, which outperforms other greedy and static migration algorithms. What's more, we come up with a determined ratio of nodes to be deployed SDN in a traditional network.

978-1-4673-8590-9/15/$31.00 ©2015 IEEE

Our contributions are three-fold:

- First, we are novel in formulating the deployment problem as a migration sequence searching problem from the perspective of traffic engineering in SDN hybrid networks. Different from the previous work, the weight setting and splitting ratio of the SDN nodes are undetermined in our model.
- Second, we demonstrate that the optimization problem is NP-complete and leverage some heuristic algorithms, i.e., a genetic algorithm, a greedy algorithm, and some static migration algorithms, to seek a migration sequence. We are novel in exploiting a genetic algorithm to seek a migration sequence, which outperforms greedy algorithm and the static migration algorithms.
- Third, we carry on the experiments and make evaluations of various algorithms. We come to a conclusion that the migration sequence that we exploit the genetic algorithm to seek can achieve a maximum link utilization less than the other algorithms during the migration and we can reap the most of benefit with 40% of the nodes migrated.

The rest of the paper is organized as follows. Section II is the related work. Section III depicts the problem formulation of the migration. We propose a genetic algorithm GAS, a greedy algorithm GDS and other static migration algorithms for seeking optimal migration sequence in section IV. In section V, we conduct the experiments and make evaluation of various migration algorithms on different network topologies. Finally, we make conclusion in section VI.

II. Related Work

In this section, we present some related work on traffic engineering and routing optimization .

OSPF protocol is a distributed IP routing protocol and has been proven to be reliable for decades. In traditional IP network, we route the network flows on the shortest paths under OSPF protocol and we can optimize the routing of flows by adjusting OSPF link weights. However, seeking for the optimized weight setting under OSPF protocol is a NP-hard problem. Therefore, various heuristic algorithms ([4] [5] [6]) have been proposed to adjust the link weights and balance the flows so that we can optimize the network performance. However, the performance of traffic engineering is limited when the flows are routed on the shortest path. In [7] and [8], the authors propose novel routing protocols that the flows can route on the longer path with an exponential penalty, which achieves optimal traffic engineering. However, the traditional routers need to support the proposed protocols, which poses a practical limitation.

With the rise of SDN, many more approaches are emerging to facilitate traffic engineering. Google [1] and Microsoft [2] achieve a high network utilization in a fully SDN-enabled inter-DC WAN by taking advantage of the flexibility of the SDN. The traffic engineering problem can be reduced to a multi-commodity problem with the full deployment of SDN and solved in polynomial time. [9] introduces a two-layer architecture and a centralized optimizer DEFO to control the optimization layer. The new architecture combines segment routing with IGP protocol to achieve scalability and flexibility of traffic engineering in the carrier-grade network. In [10], the authors conduct a survey about traffic engineering in SDN-enabled network from the aspects of reliability, scalability and availability. However, the challenges of SDN network lie in that it is not easy nor practical to migrate a traditional network to a fully SDN-enabled network. The routers should be deployed SDN incrementally.

As a transition from the traditional network to the SDN network, it is of great significance to study the hybrid SDN network. In [11] and [12], the authors study the coexistence of centralized routing and distributed routing and propose a new architecture called Fibbing. It combines the flexibility of the centralized routing with the robustness of the distributed routing to balance the load and manage the network more easily. In [13], the authors discuss about the classification of multiple control-planes and provide a framework to study the anomalies under coexistence of multiple control-planes. The hybrid SDN network routing optimization algorithms are shown in [14] and [15]. The SDN-enabled routers and traditional routers coexist in a network and the traffic splitting ratio of the SDN-enabled routers can be arbitrary. Both of them formulate the routing problems as Linear Programming (LP) problems. They are proven to be NP-hard and are solved with polynomial time heuristic algorithms. Our work concentrates on a problem which is different from [15]. In [15], we mainly focus on the traffic engineering algorithm in hybrid SDN network. The greedy algorithm is just exploited to determine the nodes to deploy before evaluating algorithms. However, in our work, we mainly concerned about the migration algorithm of SDN nodes. It is of great significance to know the number and the position of SDN nodes in the network before routing in the hybrid network. The traffic engineering algorithm is determined and the same as the algorithm in [15].

The research on incremental deployment schemes in hybrid SDN network is in a primitive state and deserves our attention. There are a few studies in recent years and we now have a review on their works. As prior work [14] has shown, they exploit the greedy algorithm and select a node that gains the most of the network utilization each time to deploy. In the studies of [16], the authors also develop a greedy algorithm to derive a solution to the deployment schemes of SDN-enabled routers. They choose to deploy the node that provides the maximum number of alternative paths to be used in traffic engineering. However, the prior work do not discuss the number or the ratio of the nodes to deploy that reap the most of the benefit in hybrid network. What's more, they do not make a comparison with other heuristic algorithms and the performance of the migration algorithms is limited in minimizing the maximum link utilization. The migration algorithms are static algorithms and the authors do not consider the weight optimization and the computation of the optimal splitting ratio in determining the performance of migration algorithms.

There are also some related work on facility location

problems. For instance, the problem of the placement of SDN controllers, is also an interesting research topic in SDN network and gives us some hints on the incremental deployment schemes in hybrid SDN network. The authors discuss about the number and the location of the controllers in [17] and [18], which can be reduced to a minimum k-median problem. They discuss the placement of the multi-controllers from the perspective of minimizing the average latency and the worst-case latency, which is also a NP-hard problem.

III. Problem Formulation

In this section, we firstly depict the hybrid scenario, and then we formulate the problem as an optimization problem. Finally, we analyze the complexity of the problem.

A. Hybrid Scenario

A SDN/OSPF hybrid network scenario is depicted in Fig.1. In this topology, nodes 3,4,9,11 are SDN routers in hybrid mode and are controlled by an external controller. The routers in hybrid mode are both SDN-enabled and OSPF-enabled routers, so that these routers support OSPF protocol and at the same time we can arbitrarily split the outgoing flows of the SDN routers through the external centralized controllers. The others are legacy routers that in OSPF mode, which support OSPF routing protocol only.

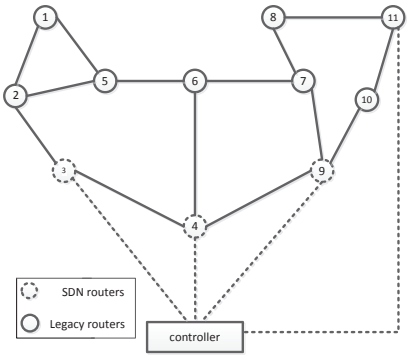


Fig. 1. A hybrid network scenario

B. Optimization Problem Formulation

We are given an undirected network graph $G = (V, A)$ (V is the vertexes set, A is the arcs set), capacity matrix C (C_{ij} denotes the capacity of link $(i, j) \in A$,$i \in V$,$j \in V$) and traffic matrix TM (TM_{ij} denotes the estimated traffic demands from node $i \in V$ to node $j \in V$). ω denotes the link weight setting matrix under OSPF protocol (ω_{ij} denotes the weight of link $(i, j) \in A$,$i \in V$,$j \in V$). $In(v)$, $v \in V$ denotes the edge sets on which there are traffic flowing into the node v. $Out(v)$, $v \in V$ represents the edge sets on which there are traffic flowing out of the node v. $\mathbf{Z}$ is the integer set. $x_e(\omega)$ denotes the splitting flow on the edge $e \in Out(c)$ with the weight setting ω. $f_e(\omega)$ denotes the flow on the edge e with the weight setting ω. $g_e^{vt}(\omega)$ denotes the flow on the edge e from v to t with the weight setting ω. We use $N = |V|$ to represent the number of the nodes in the network. We intend to obtain a migrating sequence $S = (v_1, v_2, ...v_N)$ of the nodes so that the total maximum link utilization is minimized. We deploy each node as the sequence in S. We refer ϕ to be the objective function. CN is the present SDN nodes set. U_{CN} is the maximum link utilization with the nodes in CN migrated. The migrating sequence, weight setting and the splitting ratio are all variables in our formulation.

To measure the performance of the migrating sequence, we add up the maximum link utilization with the migration of each node. Our goal is to find the migration sequence S with the least total maximum link utilization. We can formulate the objective function as follows, which is subjected to the constraints from (2)-(5).

$$\phi = \textbf{minimize} \sum_{s=1}^{N} U_{\{v_1,v_2,.....v_s\}} \tag{1}$$

$$\sum_{e \in In(c)} f_e(\omega) + TM_{ct} = \sum_{e \in Out(c)} x_e(\omega) \quad c \in CN, t \in V \tag{2}$$

$$\sum_{v,t \in V} g_e^{vt}(\omega) \leq U_{\{v_1,v_2,.....v_s\}} C(e) \quad e \in A \tag{3}$$

$$\omega_{ij} \in \mathbf{Z} \quad i \in V, j \in V \tag{4}$$

$$x_e(\omega) \geq 0 \quad e \in A \tag{5}$$

(2) denotes that the flows flow into the node plus the flows originate from this node equals the flows that flow out of this node, which is the flow conservation. At the same time, the equation implies the demand is meet. (3) denotes that the total flows on an edge cannot exceed its capacity, which is the capacity constraint. (4)-(5) denote that the weights are integers and the flow on a link must be non-negative.

In the problem formulation, we should firstly determine a migration sequence and then we can exploit the proposed traffic engineering algorithm in [15] to evaluate the migration sequence.

C. Problem Complexity

Searching for an optimal migration sequence is harder than a Travelling Salesman Problem (TSP) problem, which traverses each city once and returns to the origin city with the least distance. In TSP problem, the cost of every edge is fixed and independent on the other edges. While in our problem, the maximum link utilization is dependent on the nodes migrated before. If the maximum link utilization after the migration of each node is determined and independent on the other nodes, then our problem can be reduced to a TSP problem, which is a NP-complete problem. Our problem is proved to be harder than a NP-complete problem. So the complexity of optimal migration sequence searching is a NP-complete problem.

IV. Migration Algorithm

It is computation intensive to search an optimized migration sequence, for the space of the solution is $N!$. The search is exhaustive if we exploit the brute force algorithm, which is unacceptable. Given the NP-completeness of the problem, we resort to heuristic algorithms for optimized migration sequence. In this section, we firstly propose two heuristic algorithms, i.e., genetic algorithm searching (GAS) and greedy algorithm searching (GDS), to solve the problem in section III and analyze the complexity of the algorithm. Then, we propose some simple and novel static migration algorithms for seeking migration sequences, which are used to conduct comparison experiments in the next section.

Algorithm 1: GAS

Input: $G = (V, A)$, CN , C , TM , ω
Output: S

1 $i = j = k = 0$;
2 **for** $i < RS$ **do**
3 $chroms[i].s$ =Rand_initial();
4 $chroms[i].\phi$ =compute_util();
5 $i++$;
6 **for** $j < I$ **do**
7 (top_chrom,mid_chrom,low_chrom)=sort_chrom();
8 **for** $k <$ *number of chromosomes in* mid_chroms **do**
9 par_ch_1 = one parent from top_chroms;
10 par_ch_2 = one parent from mid_chroms;
11 $new\_chrom.s$ =cross_over (par_ch_1, par_ch_2);
12 t = mutation($new\_chrom.s$);
13 $new\_chrom.s$ = t;
14 $new\_chrom.\phi$ =compute_util();
15 $k++$;
16 $chroms$ =generation_updates(top_chrom,new_chrom);
17 $j++$;
18 $S = top\_chroms[0].s$;

A. Genetic Algorithm

The genetic algorithm belongs to the evolutionary algorithms (EA). It generates solutions to the optimization problems exploiting techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. Algorithm 1 depicts the genetic algorithm that we propose in searching for the optimized migration sequence. We now show the algorithm in details. RS denotes maximum number of random solution and I denotes maximum number of iteration. Both of them are constant numbers.

1) Initialize population: We firstly initialize the population by randomly generating some arrays with a permutation of N nodes and also exploiting the solutions generated by the static algorithms in the IV-C(line 3). For example, a random sequence of four nodes can be $[3, 2, 0, 1]$. Given the migration sequence, we can obtain the value of ϕ by computing the maximum link utilization with each node deployed (line 4). Each chromosome has two attributes, with $chroms.s$ denoting the migration sequence and $chroms.\phi$ representing the value of ϕ corresponding to $chroms.s$. Then we can begin the process of propagation with the initial population.

2) Sort chromosomes: We exploit bubble sort method to sort the chromosomes in increasing order of value ϕ (line 7). According to its value of ϕ, we divide the solutions into three classes: top class, middle class and low class (line 7). Through extensive experiments, we choose the percentage of three classes to be $10\%, 80\%, 10\%$ of the total population, respectively.

3) Parent selection: We then carry on parent selection. We select one parent from the top class (top_chrom), and the other parent from the middle class (mid_chrom) (line 9-10). The size of the parents we select is set to be the same as the middle class population. Then we can begin the crossover and mutation.

4) Crossover and mutation: Then we crossover the two parents to generate a new solution and mutate the new solution according to a pre-set probability (line 11-13). The method we employ to crossover is to inherit first half of array from the sorted nodes of father and then the remaining half from the sorted nodes of mother which do not appear in the first half of the father. For example, if the sequence of one parent chromosome is $[3, 2, 1, 0]$ and the other parent chromosome is $[2, 0, 1, 3]$, then the generated chromosome is $[3, 2, 0, 1]$. In mutation, we randomly generate two integer numbers ranging from 0 to N as the indexes and exchange the nodes in the corresponding array positions according to the given possibility.

Algorithm 2: Compute_util

Input: $G = (V, A)$, CN , C , TM
Output: ϕ

1 Initialize $\phi\, CN = \emptyset$; **for** *choose each node* s *as the sequence in* S **do**
2 $util$ =SOTE(V, A, CN, C, TM);
3 $\phi = +util$;
4 return ϕ;

5) Generation update: Finally, we update the present population (line 16). The top_chrom are put in the next generation. Then, the new solutions through crossover and mutation are inherited to the next generation. Finally, some randomly generated chromosomes also join in the next generation to escape local optimum. We exploit the new generation to continue the propagation until the terminal conditions.

At the last of the iteration, the sequence at the top of the sorted population is the best solution that we are seeking (line 18).

Algorithm 2 shows the process of computing ϕ. We resort to the algorithms (Algorithm 3 and 4) in [15] to compute the utilization after the migration of one node. It illustrates how

Algorithm 3: SOTE

Input: $G = (V, A)$, CN , C , TM
Output: U

Initial weight setting matrix W , U ;
$currutil =$ floydwarshall (G, W, TM) ;
$bestutil = currutil$;
$currweights = W$;
$bestweights = W$;
foreach *iteration times increase by one* **do**
    $currweights =$neighbour_search (G, W);
    $currutil =$Splitting_ratio $(currweights, CN, TM, C, V, A)$;
    if $currutil < bestutil$ **then**
        $bestutil = currutil$;
        $bestweights = currweights$;
$U = bestutil$;

Algorithm 4: Splitting_ratio

Input: local optimal weight setting $currweights$, CN, TM, C, V, A
Output: U

foreach *vertex* $d \in V$ **do**
    $DAG =$Dijkstra_Shortest_Path$(currweights, d)$;
    foreach *arc* (i, j) *in outgoing_links*(SDN_i) **do**
        $DAG(i, j) = 1$;
        if check_loop$(DAG) == 1$ $DAG(i, j) = 0$;
    foreach *vertex* v *in Topological_Sort*(V) **do**
        Route_Flow (v);
$expr =$ Multi_Commodity (A, V);
cplex_solve $(expr)$;

to exploit the local search to determine the weight setting (Algorithm 3) and how to compute the splitting ratio of SDN nodes in SDN/OSPF hybrid network (Algorithm 4). The details are shown in [15]. We then add the utilization and obtain the total ϕ.

Now we analyze the complexity of the algorithm. The complexity of Rand_initial and sort_chrom are both $O(n^2)$. The complexity of cross_over is $O(n^3)$ and the complexity of compute_util is $O(n^4)$. Therefore, the complexity of the entire algorithm is $O(n^5)$.

B. Greedy Algorithm

We also propose a simple greedy algorithm GDS to search for the optimized migrating sequence. The details are shown in Algorithm 5.

In each iteration, we choose the node that we can get minimum maximum link utilization after its migration and add it to the end of the present migration sequence. In this method, we obtain the complete migration sequence eventually. The complexity of the GDS is $O(n^5)$.

Algorithm 5: GDS

$i = 0$;
$bestutil = 1$;
$D = \varnothing$;
while $i < N$ **do**
    for *every node* $a \in V$,$a \notin D$ **do**
        $util =$ compute_util();
        if $util < bestutil$ **then**
            $b = a$;
            $bestutil = util$;
    $D = D \bigcup b$;
    $i++$;
return D;

C. Other Static Migration Algorithms

We define the **alternative links** of a router are the links that can be used to split the traffic in traffic engineering when the router has migrated from OSPF mode to hybrid mode.

For example, in Fig.1, the alternative links of node 2 are the links (2,1), (2,5), (2,3).

1) Alternative links (AL): In this static algorithm, we deploy the router according to its number of alternative links. The routers with more alternative links have higher priority to be deployed in this strategy. The intuitive idea behind this strategy is that, the more alternative links the router has, the more capability that the router has to split the flows once deployed and the less likelihood that the alternative links get congested in traffic engineering. A_{ij} denotes the link from node i to node j. $A_{ij} = 0$ if there is no link between node i and node j, otherwise $A_{ij} = 1$. Then the number of alternative links of node i can be calculated as follows:

$$N_i = \sum_{j \in V} A_{ij} \tag{2}$$

2) Traffic demands (TD): In this static algorithm, we sort the routers in descending order of the traffic demands. We choose to deploy the router with more traffic demands originated from it so that we can deal with "big" flows first and perhaps obtain a better result in traffic engineering. The traffic demands originated from node i can be computed as follows:

$$T_i = \sum_{j \in V} TM_{ij} \tag{3}$$

3) Maximum link utilization (MLU): In this static algorithm, in order to decrease the maximum link utilization, we migrate the routers, whose outgoing link has the maximum link utilization, to hybrid mode firstly. We first exploit the Floyd algorithm to find the shortest path between each pair of nodes and route the traffic demands on the links of the path. Then we can get the utilization matrix of the network and sort the nodes according to the maximum link utilization of

its outgoing links. The maximum link utilization is computed as follows:

$$M_i = max_{j \in V} U_{ij} \quad (4)$$

The aforementioned static migration algorithms are all considered from the perspective of minimizing the maximum link utilization. We will evaluate these algorithms in section V.

V. Experiments And Evaluation

In this section, we carry on the simulation experiments to evaluate various migration algorithms. We first show the network topologies that we exploit to conduct experiments. Then, we plot the curves of the maximum link utilization varying with different number of migrated nodes. Finally, we analyze the performance of different migration algorithms. All of the simulation experiments are done on a personal computer with 2.6GHz Intel Core 4 CPU and 4GB memory.

A. Topologies

The real topologies that we refer to are shown in TABLE I. The topologies are inferred by Rocketfuel [19]. We also employ the synthetic topology generated by BRITE [20] to conduct the experiments. The parameters are shown in Table II. The three synthetic network generated by BRITE is shown in TABLE III.

TABLE I
Topologies from ROCKETFUEL

Name	Nodes	Links
Abovenet	17	74
Ebone	18	66
Exodus	21	72

TABLE II
Parameters for BRITE

Model	N	HS	LS
Waxman	10/15/20	1000	100
m	NodePlacement	GrowthTypem	$alpha$
6	Random	Incremental	0.15
$beta$	BWDist	BwMin	BwMax
0.2	Heavy Tail	10.0	1024.0

TABLE III
Synthetic topologies

Name	Nodes	Links
Synthetic topology1	10	26
Synthetic topology2	15	62
Synthetic topology3	20	118

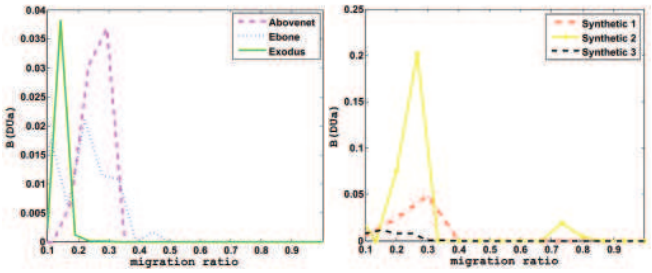


Fig. 2. $B(D \bigcup a)$ under six different topologies

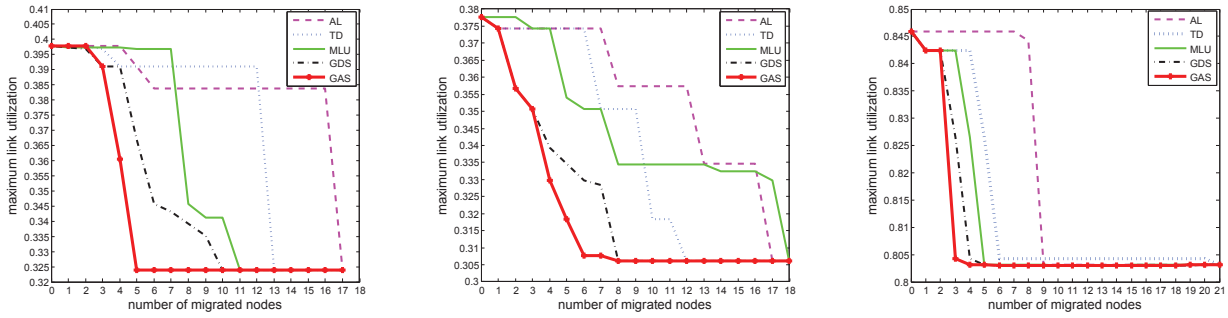
B. Simulation Experiments

We compare the GAS to GDS and other static algorithms, which are AL, TD, MLU, respectively. Through extensive experiments, we choose to set the iteration time to be 50. The traffic demands matrices are generated randomly. We plot the curves that the maximum link utilization varies with the increasing of the migrated nodes under different topologies. The results are shown in Fig.3 and Fig.4.

From the curves above, we can find that the maximum link utilization decreases with the increasing of the number of the migrated nodes. The curves of GAS decrease drastically at beginning and become flat in the following. GAS algorithm can seek a better migration sequence compared with other migration algorithms in minimizing the maximum link utilization.

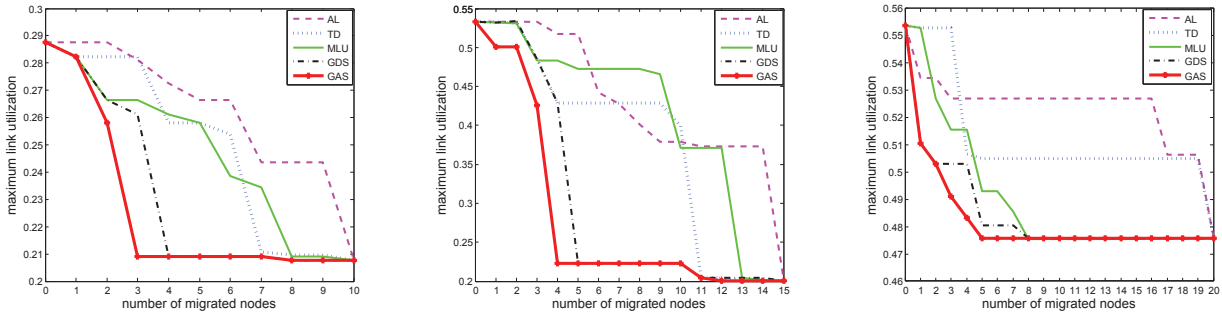
We define $B(D \bigcup a) = U_D - U_{\{D \bigcup a\}}$ (D is the set of nodes already migrated and a is a new node to be migrated). $B(D \bigcup a)$ denotes the benefit of migration of each node a. In Fig.2, we plot the curves that $B(D \bigcup a)$ varies with the migration ratio. We can find that $B(D \bigcup a)$ varies drastically when the migration ratio is less than 40% $B(D \bigcup a)$ and becomes steady when the migration ratio is more than 40%. We can draw the conclusion that when the migration ratio is less than 40%, we can reap a great benefit with the migration of one more node; when the migration ratio is more than 40%, there is not much meaning in migrating the nodes. 40% of migration can be seen as a turning point in the curves. As a result, we can choose to migrate 40% of the nodes firstly.

In Fig.5 and Fig.6, we draw the Cumulative Distribution Function (CDF) graphs of the maximum link utilization with 40% of nodes migrated under different algorithms and compare it to the maximum link utilization with 100% of nodes migrated under GAS algorithm. We conduct twenty experiments with randomly generated traffic matrices for each algorithm. We compare our algorithm with other static algorithms and GDS algorithm with a migration of 40%. 100%-GAS denotes that the migration ratio is 100%. As the figures illustrate, when 40% of nodes migrated, GAS performs better than the other static migration algorithms and GDS algorithm. When strategically deployed, 40% of nodes of migration can reap the most of benefit and is close to the performance with 100% of nodes migrated.



(a) Abovenet (b) Ebone (c) Exodus

Fig. 3. The migrating curves of three real topologies



(a) Synthetic topology 1 (b) Synthetic topology 2 (c) Synthetic topology 3

Fig. 4. The migrating curves of three synthetic topologies

VI. Conclusion

The SDN/OSPF hybrid network scenario has attracted people's attention recently, while it is still in a primitive period. Therefore, it is a practical and significant problem to migrate the traditional IP network to the SDN/OSPF hybrid network from the perspective of traffic engineering and we should determine which router to migrate firstly. By studying the migration sequence, we can better analyze the problem of migration and determine which node to deploy first. The goal of traffic engineering is to minimize the maximum link utilization. In this paper, we consider the migration to the SDN/OSPF hybrid network in minimizing the maximum link utilization. We have exploited a genetic algorithm, i.e., GAS, to seek a sequence that can obtain a better performance compared to GDS and static algorithms during migration. We also come to a conclusion that with a migration of 40% of the nodes, we can reap the most of the benefit.

In the future, we will conduct extensive experiments on real SDN devices and consider other traffic engineering problems in SDN/OSPF hybrid network from the perspective of network operators.

Acknowledgment

This work is partially supported by the National High Technology Research and Development Program of China (863 Program) No. 2015AA016105 and 2015AA015603, the National Natural Science Foundation of China (Grant No. 61202357), and the Project for 2012 Next Generation Internet technology research and development, industrialization, and large scale commercial application of China (No. 2012 1763).

References

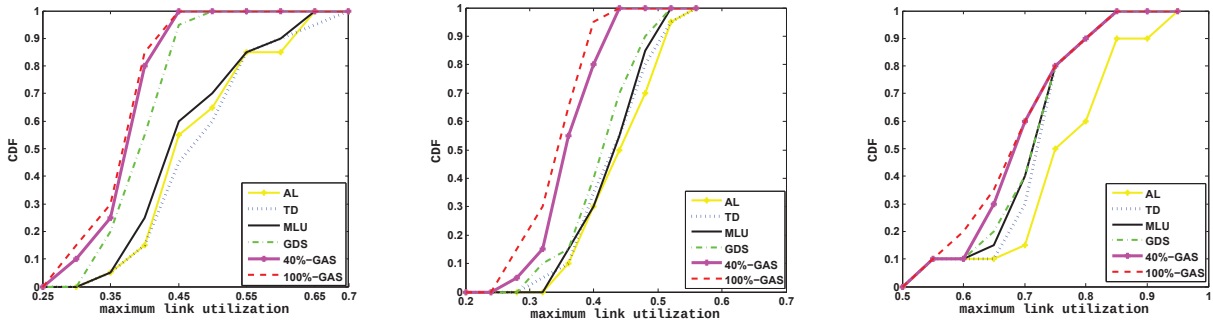
[1] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined wan," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*. ACM, 2013, pp. 3–14.

[2] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*. ACM, 2013, pp. 15–26.

[3] S. Vissicchio, L. Vanbever, and O. Bonaventure, "Opportunities and research challenges of hybrid software defined networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 70–75, 2014.

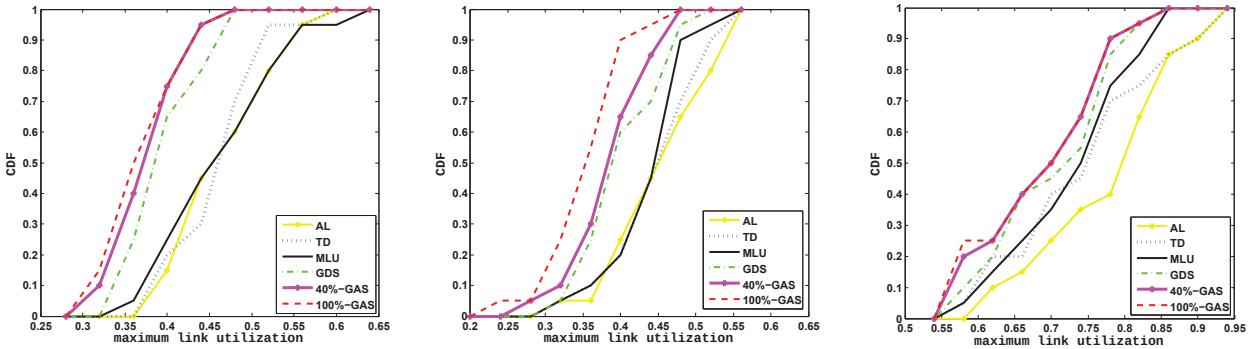
[4] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing ospf weights," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2. IEEE, 2000, pp. 519–528.

[5] M. Ericsson, M. G. C. Resende, and P. M. Pardalos, "A genetic algorithm for the weight setting problem in ospf routing," *Journal of combinatorial optimization*, vol. 6, no. 3, pp. 299–333, 2002.



(a) Abovenet (b) Ebone (c) Exodus

Fig. 5. CDF curves of three real topologies



(a) Synthetic topology 1 (b) Synthetic topology 2 (c) Synthetic topology 3

Fig. 6. CDF curves of three synthetic topologies

[6] A. Altın, B. Fortz, M. Thorup, and H. Ümit, "Intra-domain traffic engineering with shortest path routing protocols," *Annals of Operations Research*, vol. 204, no. 1, pp. 65–95, 2013.

[7] D. Xu, M. Chiang, and J. Rexford, "Deft: Distributed exponentially-weighted flow splitting," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*. IEEE, 2007, pp. 71–79.

[8] ——, "Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering," *IEEE/ACM Transactions on Networking (TON)*, vol. 19, no. 6, pp. 1717–1730, 2011.

[9] R. Hartert, S. Vissicchio, P. Schaus, O. Bonaventure, C. Filsfils, T. Telkamp, and P. Francois, "A declarative and expressive approach to control forwarding paths in carrier-grade networks," in *ACM SIGCOMM*, 2015.

[10] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in sdn-openflow networks," *Computer Networks*, vol. 71, pp. 1–30, 2014.

[11] S. Vissicchio, L. Vanbever, and J. Rexford, "Sweet little lies: Fake topologies for flexible routing," in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*. ACM, 2014, p. 3.

[12] S. Vissicchio, O. Tilmans, L. Vanbever, and J. Rexford, "Central control over distributed routing," in *ACM SIGCOMM*, 2015.

[13] S. Vissicchio, L. Cittadini, O. Bonaventure, G. G. Xie, L. Vanbever *et al.*, "On the co-existence of distributed and centralized routing control-planes," in *IEEE INFOCOM, April*, 2015.

[14] S. Agarwal, M. Kodialam, and T. Lakshman, "Traffic engineering in software defined networks," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 2211–2219.

[15] Y. Guo, Z. Wang, X. Yin, X. Shi, and J. Wu, "Traffic engineering in sdn/ospf hybrid network," in *Network Protocols (ICNP), 2014 IEEE 22nd International Conference on*. IEEE, 2014, pp. 563–568.

[16] M. Caria, A. Jukan, and M. Hoffmann, "A performance study of network migration to sdn-enabled traffic engineering," in *Global Communications Conference (GLOBECOM), 2013 IEEE*. IEEE, 2013, pp. 1391–1396.

[17] Y.-n. HU, W.-d. WANG, X.-y. GONG, X.-r. QUE, and S.-d. CHENG, "On the placement of controllers in software-defined networks," *The Journal of China Universities of Posts and Telecommunications*, vol. 19, pp. 92–171, 2012.

[18] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 7–12.

[19] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "Inferring link weights using end-to-end measurements," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. ACM, 2002, pp. 231–236.

[20] A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: An approach to universal topology generation," in *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001. Proceedings. Ninth International Symposium on*. IEEE, 2001, pp. 346–353.