Hindawi
Security and Communication Networks
Volume 2017, Article ID 4651395, 13 pages
https://doi.org/10.1155/2017/4651395

Research Article

Scalable Node-Centric Route Mutation for Defense of Large-Scale Software-Defined Networks

Yang Zhou,[1] Wei Ni,[2] Kangfeng Zheng,[1] Ren Ping Liu,[3] and Yixian Yang[1]

[1]*School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2]*CSIRO, Sydney, NSW 2122, Australia*
[3]*Global Big Data Technologies Centre, University of Technology Sydney, Ultimo, NSW 2007, Australia*

Correspondence should be addressed to Yang Zhou; zhouyang@bupt.edu.cn

Received 15 September 2017; Accepted 27 November 2017; Published 25 December 2017

Academic Editor: Qing Yang

Exploiting software-defined networking techniques, randomly and instantly mutating routes can disguise strategically important infrastructure and protect the integrity of data networks. Route mutation has been to date formulated as NP-complete constraint satisfaction problem where feasible sets of routes need to be generated with exponential computational complexities, limiting algorithmic scalability to large-scale networks. In this paper, we propose a novel node-centric route mutation method which interprets route mutation as a signature matching problem. We formulate the route mutation problem as a three-dimensional earth mover's distance (EMD) model and solve it by using a binary branch and bound method. Considering the scalability, we further propose that a heuristic method yields significantly lower computational complexities with marginal loss of robustness against eavesdropping. Simulation results show that our proposed methods can effectively disguise key infrastructure by reducing the difference of historically accumulative traffic among different switches. With significantly reduced complexities, our algorithms are of particular interest to safeguard large-scale networks.

1. Introduction

Instantly mutating route is a promising technique to provide the integrity of large data networks [1]. It can disguise strategically located important network infrastructures and delay or prevent potential reconnaissance attacks [2]. This is of particular interest to many practical networks delivering security-sensitive data, such as military networks [3] and banking systems [4]. On the other hand, software-defined network (SDN) has been increasingly deployed. Centralized control in SDN facilitates mutating routes to disguise the strategically important switches and protect the network.

In general, route mutation is NP-complete constrained path selection in the presence of quality-of-service (QoS) consideration [5]. It has been formulated to be a constraint satisfaction problem and solved by using the satisfiability modulo theory (SMT) [6]. The constraints include the QoS of traffic flows and physical bandwidth of routers [6]. The final routes are randomly chosen from the results of SMT that satisfy constraints. However, the routes can hardly claim optimality in any sense, and the complexity of SMT is prohibitively high [7]. Although some heuristics have been developed to solve this problem within pseudo-polynomial-time [8–11], these algorithms concern network optimality rather than security.

In this paper, we propose new node-centric route mutation methods which are able to effectively change routes of flows at substantially reduced complexities. The key idea is that we propose interpreting route mutation as a signature matching problem and developing a three-dimensional earth mover's distance (EMD) model with network connectivity and QoS constraints to suppress the traffic difference among switches. Another important aspect of our algorithm is that we solve the new node-centric problem as a three-dimensional transportation problem and develop suboptimal algorithms with polynomial time-complexities. Simulation results show that our algorithms are able to suppress the traffic difference among strategically important switches and also achieve fast convergence with substantially reduced complexities.

The rest of paper is organized as follows. Section 2 presents related work. Section 3 describes the system and network model. In Section 4, we discuss the details of designed algorithm. We present the computational efficient heuristics that can apply to large-scale networks in Section 5. Section 6 evaluates the experiment results and performance of our algorithm. In Section 7, we conclude the paper.

2. Related Work

In [12], randomly mutating route or host address was proposed to protect key network infrastructures from exposure to attackers and prevent potential attacks. Only a few structured algorithms have been designed for route or host address mutation, all of which formulated NP-complete constrained satisfaction problems under the constraints of the mutation rate [13], the address range [14], and the range distribution for host address mutation [6, 15]. These algorithms were solved by using SMT [6, 13–15]. However, constraint satisfaction problems are NP-complete with exponential complexities [16]. As far as the routing problem is concerned, there are $N = \sum_{j=1}^{n-1}((n-2)!/(n-j-1)!)$ possible routes with lengths shorter than j hops in an n-node complete network [17]. Choosing k routes out of N yields a complexity of $N!/(N-k)!k!$, limiting the scalability of the designs to large-scale networks.

More efforts have been spent demonstrating the concept of route mutations. Route mutation was first implemented under IPv6, referred to as "Moving Target IPv6 Defense (MT6D)," by continually rotating the IPv6 addresses between the senders and recipients [18]. A new transport layer protocol was designed to enable multiple addresses per network socket and support the rotations of IPv6 addresses [19]. Route mutation was later applied to mobile ad hoc networks [20], by repurposing a classic attack mechanism, the Sybil attack, for effective defense. Most recently, route mutation was demonstrated using Cisco's SDN platform, One Platform Kit [21]. With the flexibility of SDN, a deception system that mutates the topologies virtually was proposed to defend against reconnaissances [22]. A double hopping communication approach was designed to combine the IP address mutation and the route mutation methods in SDN [23], in which routes were selected from all paths within even lengths between sources and destinations. Game theory was adopted for route mutation in [24]. Route mutation was proposed to apply to traditional networks where links were selected using stochastic game routing policies of next-hop probability distribution. In [25], a virtual network embedding framework was proposed to satisfy the capacity, delay, and cycle-free constraints when mutating links by game theoretic routing policies. In [1], a randomized multipath routing method was developed to circumvent black holes and minimize the overall energy consumption in wireless sensor networks.

3. System Model

Figure 1 illustrates the software-defined network under consideration, where there are N number of switches connected

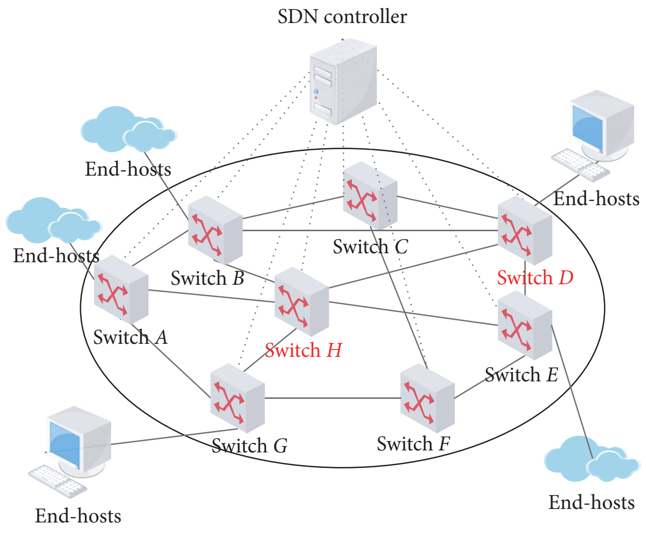


Figure 1: An illustration of SDN topology.

in a network topology and forwarding traffic. We assume that all the switches are connected to a SDN controller in a star topology. The SDN controller is responsible for route discovery and mutation. The switches forward traffic along the routes specified by the SDN controller. The bandwidth of each switch, that is, switch i, is B_i.

Each of the switches is also connected to end-hosts which generate routing requests through the switch to the SDN controller. An end-host can request sending traffic flows to any other end-host. Each traffic flow is assumed to be randomly generated and last for a certain period of time. Without loss of generality, we assume that there are at most K routing requests at any instant. The data rate of the kth flow is denoted by f_k $(k = 1, \ldots, K)$.

Consider a practical network with hundreds of switches and nontrivial topologies (as opposed to fully connected complete graphs). Some of the switches have more connections than the others. These switches are strategically located and handle more traffic than other switches. In the example of Figure 1, switches D and H are such strategically located nodes. The exposure of these switches to adversaries poses critical threats to the integrity of the network.

The topology of the network can be described by an undirected graph $G(\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ collects the vertexes (i.e., switches) and $\mathcal{E}$ collects the edges (i.e., the links connecting the switches). The size of $\mathcal{N}$ is $|\mathcal{N}| = N$, where $|\cdot|$ stands for cardinality. Also define $\mathbf{A}$ to be the adjacent matrix of G. $\mathbf{A}(i, j) = 1$ $(i \neq j)$, if vertices i and j are connected; or $\mathbf{A}(i, j) = 0$, otherwise.

Reconnaissance attacks [26] are considered, where an adversary keeps monitoring (or eavesdropping) the switches until strategically located switches are identified and attacks (such as DDoS) are initiated. The difference of historically accumulative traffic among switches is typically used by the adversary to identify strategically located switches, for example, D and H in Figure 1. The exposure of the strategically located switches would facilitate the potential attacks. Consider the worst-case scenario that the adversary is able

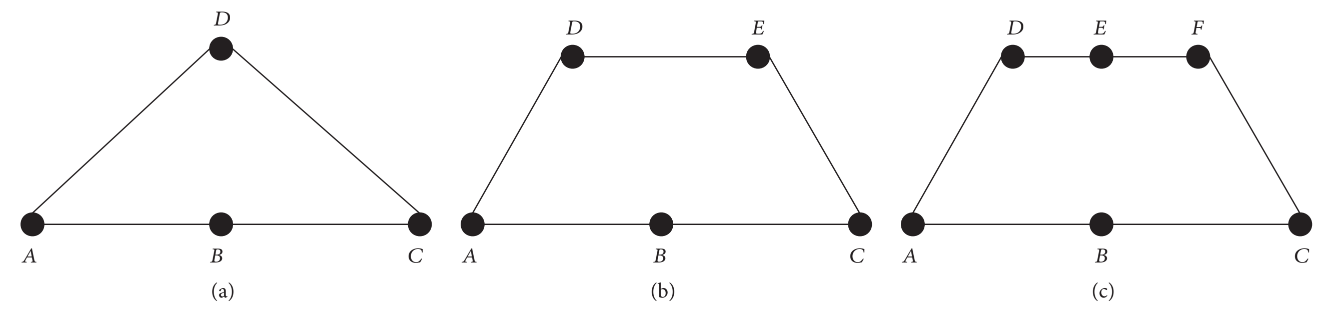


Figure 2: Illustrations of multihop detours around a strategically located node, say node B in this example, where the lengths of the detours are 2, 3, and 4, respectively.

to access each individual switch, install invasive software, accumulate the traffic volume handled at the switch, and collect the traffic volumes of all the switches. In practice, this can be implemented by remotely installing monitoring software, such as Cacti [27], to get the switches to collect and report the SNMP information to the adversary.

4. The Proposed Route Mutation for Large-Scale Networks

In this section, we propose mutating routes on a node basis to leverage between the robustness against reconnaissance attacks and the computational complexity. Particularly, routing requests from end-hosts are responded to independently from one another using classical routing techniques, such as OSPF [28]. The exponentially increasing complexity of joint routing, such as the one proposed in [6], can be avoided.

Given the routes, we propose that the SDN controller identifies historically heavily loaded nodes with high exposure risks and detours their traffic flows via other historically lightly loaded nodes. A node is identified to be historically heavily or lightly loaded, based on the accumulative traffic of the node. By this means, the strategically located nodes can be disguised and protected, delaying or even preventing potential attacks.

The detours bypassing a historically heavily loaded node can be multihop. Figure 2 gives the examples of two-, three-, and four-hop detours that are considered, where node B is a strategically located node. The possible detours for a traffic flow currently passing node B must connect the nodes one hop away from node B along the traffic flow, that is, nodes A and C. Such detours can be readily obtained using depth-first search (i.e., search for the closed loops passing node B) [29]. The lengths of the detours are 2, 3, and 4 (in numbers of hops) in Figures 2(a), 2(b), and 2(c), respectively.

We also propose interpreting the route mutation as a signature matching problem. The accumulative traffic loads of the nodes are visualized as a signature, and routes are mutated to conform the signature to the one with even loads across all the nodes. Widely used to measure the difference of two images [30], EMD [31] is extended to measure the difference between the two signatures. The difference, or more specifically, the EMD, is reduced recursively by detouring flows around historically heavily loaded nodes, as described earlier.

The proposed route mutation is able to substantially reduce the computationally complexity. Assume that the detours are limited to two hops. The worst-case complexity is $\mathcal{O}(n^2 2^n)$, consisting of detour discovery for every switch and redirecting at most K traffic flows from every switch, where n represents for the number of variables. In this case, $n \leqslant KN(N-3)$. In contrast, the state-of-the-art joint mutation of all routing paths, using SMT [6], would result in prohibitive exponential complexity and limited scalability to networks with hundreds of nodes.

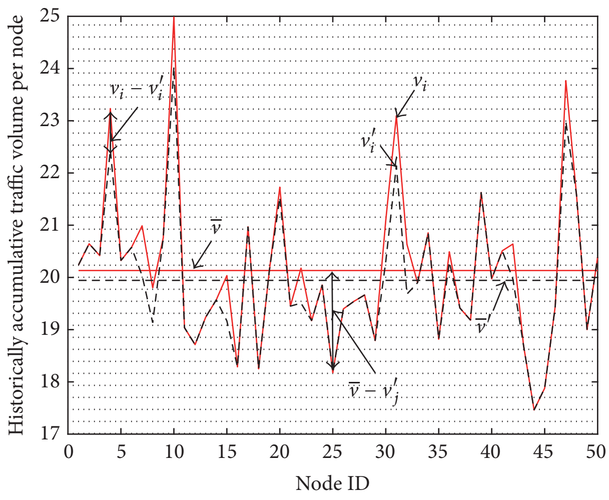
4.1. Nodewise Route Mutation for Delay-Tolerant Traffic. First consider delay-tolerant traffic. The total number of current traffic flows is K, consisting of existing ongoing flows collected in the set $\mathcal{K}^e$ and new flows collected in the set $\mathcal{K}^n$. $\mathcal{K} = \mathcal{K}^e \cup \mathcal{K}^n$. $K = |\mathcal{K}|$. Denote v_i', $i \in \mathcal{N}$, and $\bar{v}'$ (in bytes) to be the accumulative traffic of each individual node i and their average, respectively, before a round of route mutation. $\bar{v}' = (1/N)\sum_{i=1}^{N} v_i'$. v_i and $\bar{v}$ are the accumulative traffic of node i, $i \in \mathcal{N}$, and the average, respectively, after the round of route mutation. $\bar{v} = (1/N)\sum_{i=1}^{N} v_i$.

The proposed route mutation redirects the K traffic flows, attempting to conform the pictorial signature of the accumulative traffic of the nodes to a uniform signature with even traffic of $\bar{v}$ at every node. To do this, we construct two signatures, that is, two one-dimensional distributions: $\{v_i - v_i',\ \forall i \in \mathcal{N}\}$ and $\{\bar{v} - v_j',\ \forall j \in \mathcal{N}\}$. The first signature $\{v_i - v_i',\ \forall i \in \mathcal{N}\}$ provides the traffic volumes that can detour. The second signature $\{\bar{v} - v_j',\ \forall j \in \mathcal{N}\}$ denotes the demand of each node to achieve traffic uniformity among the nodes, as illustrated in Figure 3.

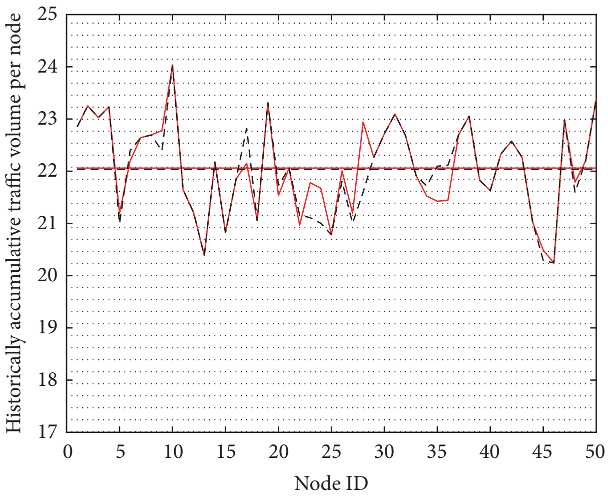
The EMD can be defined to quantify the difference between the two signatures, as given by

$$\epsilon^* = \arg\min_{\epsilon(\tau)} \tau, \tag{1a}$$

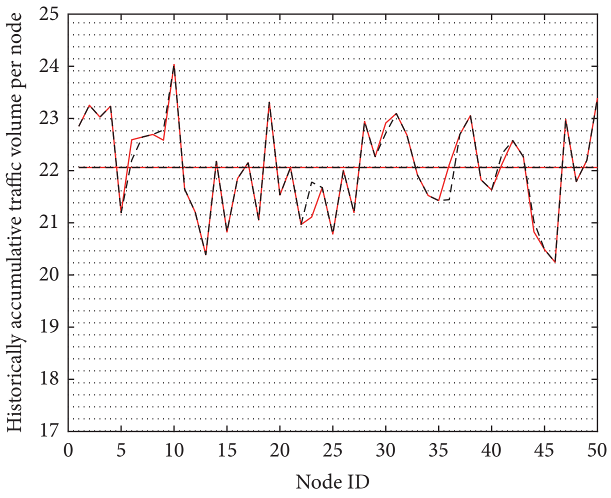
$$\epsilon(\tau) = \left(\frac{\sum_{i\in\mathcal{N}} \sum_{k=1}^{K} \sum_{r=1}^{R} \sum_{\mathbf{j}\in\mathcal{L}_r(i,k)} c_{i\mathbf{j}} \theta_{ik} f_k x_{i\mathbf{j}k}}{\min\left(R\sum_{i\in\mathcal{N}} \left(v_i - v_i'\right), \sum_{j\in\mathcal{N}} \left[\bar{v} - v_j'\right]^+\right) - \tau} \right), \tag{1b}$$



(a) Historically accumulative traffic volumes of nodes when Algorithm 1 starts



(b) Historically accumulative traffic volumes of nodes before Algorithm 1 terminates



(c) Historically accumulative traffic volumes of nodes after Algorithm 1 terminates

Figure 3: An illustration of the proposed application of signature matching to route mutation, where v_j and v_j' are the historically accumulative traffic volumes of node i at the current and the previous instants, respectively, and $\overline{v}$ and $\overline{v}'$ are the averages of the historically accumulative traffic volumes at the current and the previous instants, respectively. v_i' and $\overline{v}'$ are the results of the route mutation at the previous instant. v_i and $\overline{v}$ are the values before the current route mutation is executed.

where $[\cdot]^+ = \max(\cdot, 0)$; R is the maximum allowed length of a detour (in numbers of hops); $x_{i\mathbf{j}k} = 1$ if the current flow k is to be offloaded from node i to detour $\mathbf{j}$, or $x_{i\mathbf{j}k} = 0$, otherwise; $\theta_{ik} = 1$, if flow k currently goes through node i, or $\theta_{ik} = 0$, otherwise; $\delta_{j\mathbf{j}}$ is given by

$$\delta_{j\mathbf{j}} = \begin{cases} 1, & \text{if } j \in \mathbf{j}; \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

$\mathscr{L}_r(i,k)$ is the set of detours with length of $r \leq R$, and each of the detours connects to the two nodes one hop away from node i along the current route of flow k. For any $\mathbf{j} \in \mathscr{L}_r(i,k)$, $\mathbf{j} \in \mathbb{N}^{r\times 1}$; that is, $|\mathbf{j}| = r$. Let $\mathbf{j}_l$ $(l = 1, \dots, r)$ denote the lth element of $\mathbf{j}$. We have $\mathbf{A}(\mathbf{j}_1, \mathbf{j}_2) = \cdots = \mathbf{A}(\mathbf{j}_{r-1}, \mathbf{j}_r) = 1$.

Here, $c_{i\mathbf{j}}$ is the cost of offloading traffic flows from node i to detour $\mathbf{j} \in \mathscr{L}_r(i,k)$. Keep in mind that our goal is to minimize the difference of accumulative traffic among the nodes. Meanwhile, the EMD is inherently defined to minimize the difference. For this reason, $c_{i\mathbf{j}}$ is defined as the normalized total difference of node i and detour $\mathbf{j}$ from $\overline{v}$, as specified by

$$c_{i\mathbf{j}} = \sqrt{\frac{1}{r+1}\left(\left(v_i - \overline{v}\right)^2 + \sum_{j\in\mathbf{j}}\left(v_j - \overline{v}\right)^2\right)} \quad \text{for } \mathbf{j} \in \mathbb{N}^{r\times 1}. \tag{3}$$

Our EMD has a different form to the conventional definition for image processing applications [31]. An auxiliary variable τ is defined to indicate the gap between the total traffic load that is expected to detour and the total traffic that can detour, whereas there is no such gap for image applications. This is due to the fact that route mutation (or, in other words, route reselection) is discrete and these two loads do not equate in most cases. (1a) and (1b) provide the average cost to minimize the difference between the total traffic load expected to be detoured and the total traffic load that can be detoured.

Predicted before a round of mutation, $\overline{v}$ is unnecessarily equal to the one actually achieved after the mutation. This is due to the dependence of traffic between the nodes imposed by traffic flows. Particularly, a detour needs to be taken across multiple nodes and increases the traffic of the nodes evenly. Therefore, the average traffic load is expected to increase as a result of mutation. However, the difference of $\overline{v}$ before and after a round of mutation can be iteratively reduced by increasing the rounds and diminishes as the routes stabilize. Once stabilized, the routes are implemented into the network. The convergence of the iterations can be guaranteed, since $\overline{v}$ does not decrease during the iterations while it is also obviously bounded. Figure 3 gives a detailed illustration on our signature matching method.

Given v'_j, $\overline{v}'$, v_j, and $\overline{v}$, we can formulate a binary linear programming problem to minimize the EMD of the two signatures, as given by

$$\min_{\substack{x_{\mathbf{i}jk},\tau,\forall(i,k)\in\mathcal{S}\\ \mathbf{j}\in\mathcal{L}_r(i,k)}} \left(\sum_{i\in\mathcal{N}}\sum_{k=1}^{K}\sum_{r=1}^{R}\sum_{\mathbf{j}\in\mathcal{L}_r(i,k)} c_{i\mathbf{j}}\theta_{ik}f_k x_{i\mathbf{j}k}\right), \tag{4a}$$

$$\text{s.t.} \quad \sum_{r=1}^{R}\sum_{\mathbf{j}\in\mathcal{L}_r(i,k)} \theta_{ik}x_{i\mathbf{j}k} \le 1, \quad \text{for any } k; \tag{4b}$$

$$\sum_{k=1}^{K}\sum_{r=1}^{R}\sum_{\mathbf{j}\in\mathcal{L}_r(i,k)} \theta_{ik}f_k x_{i\mathbf{j}k} \le v_i - v'_i, \quad \text{for any } i\in\mathcal{N}; \tag{4c}$$

$$\sum_{i\in\mathcal{N}}\sum_{k=1}^{K}\sum_{r=1}^{R}\sum_{\mathbf{j}\in\mathcal{L}_r(i,k)} \delta_{j\mathbf{j}}\theta_{ik}f_k x_{i\mathbf{j}k} \le \left[\overline{v} - v'_j\right]^+, \quad \text{for any } j\in\mathcal{N}; \tag{4d}$$

$$\min\left(R\sum_{i\in\mathcal{N}}\left(v_i - v'_i\right), \sum_{j\in\mathcal{N}}\left[\overline{v} - v'_j\right]^+\right) - \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k=1}^{K}\sum_{r=1}^{R}\sum_{\mathbf{j}\in\mathcal{L}_r(i,k)} \delta_{j\mathbf{j}}\theta_{ik}f_k x_{i\mathbf{j}k} \le \tau; \tag{4e}$$

$$x_{i\mathbf{j}k} \in \{0,1\}, \tag{4f}$$

where $\mathcal{S}$ is the set of nodes that can be detoured. $\mathcal{S}$ needs to be preselected so that at most one of two consecutive nodes along a traffic flow, for example, nodes B and C in Figure 2, detours. This is to prevent $\mathcal{L}_r(i,k)$ from changing during solving (4a), (4b), (4c), (4d), (4e), and (4f). We can label the nodes along a traffic flow before running (4a), (4b), (4c), (4d), (4e), and (4f). $\mathcal{S}$ can be first set to collect the nodes with even labels and input to (4a), (4b), (4c), (4d), (4e), and (4f). Then $\mathcal{S}$ is reset to collect the remaining nodes with odd labels to run (4a), (4b), (4c), (4d), (4e), and (4f) again.

Here, constraint (4b) restricts a flow that can only be offloaded from a node to one detour; (4c) and (4d) restrict the amounts of traffic that can be supplied and demanded by the first and the second signatures, respectively; (4e) specifies the total maximum traffic amount that can detour; and (4f) specifies the variables to be binary.

Given τ, (4a), (4b), (4c), (4d), (4e), and (4f) are integer linear program and can be optimally solved using a branch and bound/cut algorithm [32]. Specifically, we first relax (4f) and evaluate the upper and lower bounds of (4a) by increasingly setting the variables to "0" or "1" and using the Simplex method [33] to optimize the remaining variables. At any instant, the settings whose lower bounds are higher than the upper bound of another setting are discarded. The final remaining integer results are the solution for (4a), (4b), (4c), (4d), (4e), and (4f).

A bisection method can be taken to recursively search for the minimum feasible value of τ. When τ is too small, (4a), (4b), (4c), (4d), (4e), and (4f) can become infeasible. When τ is large, (4e) becomes inactive and (4a), (4b), (4c), (4d), (4e), and (4f) becomes always feasible. To this end, (4a), (4b), (4c), (4d), (4e), and (4f) is an on-off function of τ preserving monotonicity and can be readily solved bisectionally.

Algorithm 1 summarizes the proposed route mutation, where $\epsilon \ll 1$ is a predetermined positive threshold for the termination of the algorithm. This algorithm resides in the SDN controller and runs periodically or on-demand. As described, the kernel of the algorithm is the EMD problem (4a), (4b), (4c), (4d), (4e), and (4f) solved using the binary branch and bound/cut method in Step (11). The optimal solution for (4a), (4b), (4c), (4d), (4e), and (4f) is used to update θ_{ik}, or, in other words, to find the detours of the routes, as depicted in Step (12). By repeating these steps, the existing routes gradually move away and increase lengths, until the uniformity of all the nodes as regards accumulative traffic cannot be further improved; see the loop from Steps (2) to (13).

The majority of the complexity in Algorithm 1 lies in Step (11), that is, binary branch and bound/cut. Given a network $G(\mathcal{N},\mathcal{E})$, we can precompute all possible loops using depth-first search. Consider the worst-case scenario where the network is a complete graph. Let L denote the maximum number of detours that a node can take, and $L = N - 3 + \sum_{r=2}^{R}((N-3)!/2(N-3-r)!)$, where r is the length of detours in node numbers. As a result, the complexity of Steps (3) to (9) is $\mathcal{O}(KNL)$. In the best-case and worst-case of branch and bound/cut, the members of branches are $\sum_{k=1}^{n} k$ and $\sum_{k=1}^{n} k2^k$, respectively, where $n = KNL$. As per each branch, the complexity is $\mathcal{O}(\min(m^2, n^2))$, where m is the number of constraints [34, 35]. $m \geqslant n$, as (4f) limits each variable to be smaller than 1. Thus, the complexity is $\mathcal{O}(n^2)$. As a result, the

```
Input: $G(\mathcal{N}, \mathcal{E})$; $\mathbf{A}$; $R$; the accumulative traffic of each node
    $v_i'$; the current route of each flow $k \in \mathcal{K}$; $f_k$; and the
    initial $\theta_{ik}$ $(i \in \mathcal{N};\ k \in \mathcal{K})$.
Output: $\theta_{ik}$, $\forall i \in \mathcal{N},\ k \in \mathcal{K}$
(1) Let $v_i = v_i' + \sum_{k\in\mathcal{K}} \theta_{ik} f_k$.
(2) repeat
(3)   for $i = 1, \ldots, N$ do
(4)     for $k = 1, \ldots, K$ do
(5)       if $\theta_{ik} = 1$ then
(6)         construct $\mathcal{L}_r(i,k)$, $r = 1, \ldots, R$, using depth-first search, and calculate $\delta_{j\mathbf{j}}$.
(7)       end if
(8)     end for
(9)   end for
(10)  calculate $\bar{v} = (1/N)\sum_{i=1}^{N} v_i$, and $c_{i\mathbf{j}}$ using (3);
(11)  substitute $v_i$, $\bar{v}$, $v_i'$ $(i \in \mathcal{N})$, $\bar{v}'$, $c_{i\mathbf{j}}$ and $\theta_{ik}$ into the EMD problem (4a), (4b), (4c), (4d), (4e), and (4f),
      and solve the problem optimally using the binary branch and bound/cut method;
(12)  update
          $\theta_{ik} \longleftarrow \sum_{j\in\mathcal{N}} \sum_{r=1}^{R} \sum_{\mathbf{j}\in\mathcal{L}_r(j,k)} \theta_{jk}\delta_{i\mathbf{j}} x_{j\mathbf{j}k}$;
          $v_i \longleftarrow v_i' + \sum_{k\in\mathcal{K}} \theta_{ik} f_k$;
          $(i \in \mathcal{N};\ k \in \mathcal{K})$
(13) until $\theta_{ik}$ stops changing; or in other words, $|\Delta\bar{v}| \le \epsilon$.
```

Algorithm 1: Route mutation for delay-tolerant traffic.

lower bound complexity of Algorithm 1 is $\mathcal{O}(n^2)\mathcal{O}(\sum_{k=1}^{n} k) = \mathcal{O}(K^4N^4L^4)$, and the upper bound is $\mathcal{O}(K^2N^2L^2 \sum_{k=1}^{n} k2^k) = \mathcal{O}(K^2N^2L^2 2^{KNL})$.

4.2. Route Mutation for Delay-Bounded Traffic. Algorithm 1 can be extended to the case with QoS constraints on traffic flows. For illustration convenience, we assume that the QoS requirement of a traffic flow, say flow, is characterized by the maximum allowed route length of the flow, denoted by L_k (in numbers of hops). To this end, the following constraint is added to (4a), (4b), (4c), (4d), (4e), and (4f):

$$\sum_{i\in\mathcal{N}} \sum_{r=1}^{R} \sum_{\mathbf{j}\in\mathcal{L}_r(i,k)} r x_{i\mathbf{j}k} \le \Delta l_k, \quad \text{for any } k = 1, \ldots, K, \quad (5)$$

where $\Delta l_k = L_k - l_k$. l_k is the route length of traffic flow $k \in \mathcal{K}^e$ or is the length of the route discovered using OPSF for a new flow $k \in \mathcal{K}^n$. Equation (5) limits the total length of the detours that each flow k can take.

The finite bandwidth of switches can also be considered in the proposed algorithm. We can incorporate a new constraint of the bandwidth of each link into a node-centric optimization problem, as shown as follows:

$$B_i = \min \quad \{B(i,u),\ \text{for any } u \in \mathcal{N},\ \mathbf{A}(i,u) = 1\}; \quad (6a)$$

$$\sum_{k=1}^{K} \theta_{ik} f_k - \sum_{\mathbf{j}\in\mathcal{L}_r(i,k)} \sum_{k=1}^{K} \theta_{ik} f_k x_{i\mathbf{j}k} + \sum_{\mathbf{j}\in\mathcal{L}_r(i,k)} \sum_{j=1}^{N} \sum_{k=1}^{K} \delta_{i\mathbf{j}} \theta_{jk} f_k x_{j\mathbf{j}k} \le B_i, \quad \text{for any } i \in \mathcal{N}, \quad (6b)$$

where $B(i,u)$ denotes the maximum bandwidth of link (i,u), and u is connected to node i in one hop; that is, $\mathbf{A}(i,u) = 1$. For any node i in the network, B_i denotes the link bandwidth, which is computed as the minimum one among all bandwidths of the links connecting node i. $\sum_{k=1}^{K} \theta_{ik} f_k$ accounts for the traffic load of node i before mutation; after we mutate the flows out of node i, that is, $\sum_{\mathbf{j}\in\mathcal{L}_r(i,k)} \sum_{k=1}^{K} \theta_{ik} f_k x_{i\mathbf{j}k}$, and move flows to it, that is, $\sum_{\mathbf{j}\in\mathcal{L}_r(i,k)} \sum_{j=1}^{N} \sum_{k=1}^{K} \delta_{i\mathbf{j}} \theta_{jk} f_k x_{j\mathbf{j}k}$, its traffic load is within the range of b_i.

In practice, the maximum number of flow entries on an SDN switch is limited by the physical resources of the switch, such as CAM and SRAM. We can add constraint (7) into Algorithm 1 to restrain the maximum number of flow rules that the switches can install. Without loss of generality, we denote the average size of a flow rule as a. The number of flow rules at each switch depends on the total number of flows passing through it, as given by

$$\sum_{k=1}^{K} \theta_{ik} - \sum_{\mathbf{j}\in\mathcal{L}_r(i,k)} \sum_{k=1}^{K} \theta_{ik} x_{i\mathbf{j}k} + \sum_{\mathbf{j}\in\mathcal{L}_r(i,k)} \sum_{j=1}^{N} \sum_{k=1}^{K} \delta_{i\mathbf{j}} \theta_{jk} x_{j\mathbf{j}k} \le \frac{t_i}{a}, \quad \text{for any } i \in \mathcal{N}, \quad (7)$$

```
**Input:** $G(\mathcal{N}, \mathcal{E})$; $\mathbf{A}$; $R$; the accumulative traffic of each node
    $v'_i$; the current route of each flow $k \in \mathcal{K}$; $f_k$; the initial
    $\theta_{ik}$; $L_k$; and the initial $l_k$ $(i \in \mathcal{N};\ k \in \mathcal{K})$.
**Output:** $\theta_{ik}$, $\forall i \in \mathcal{N},\ k \in \mathcal{K}$
(1) Let $v_i = v'_i + \sum_{k\in\mathcal{K}} \theta_{ik} f_k$ and $\Delta l_k = L_k - l_k$.
(2) **repeat**
(3)     Run Steps (3) to (10) of Algorithm 1.
(4)     substitute $v_i$, $\bar{v}$, $v'_i$ $(i \in \mathcal{N})$, $\bar{v}'$, $c_{i\mathbf{j}}$, $\theta_{ik}$ and $\Delta l_k$ into the EMD problem (4a), (4b), (4c), (4d), (4e), and (4f)
        incorporating (5), and solve the problem optimally using the binary branch and bound/cut method;
(5)     update
            $\theta_{ik} \longleftarrow \sum_{j\in\mathcal{N}} \sum_{r=1}^{R} \sum_{\mathbf{j}\in\mathcal{L}_r(j,k)} \theta_{jk} \delta_{i\mathbf{j}} x_{j\mathbf{j}k}$;
            $v_i \longleftarrow v'_i + \sum_{k\in\mathcal{K}} \theta_{ik} f_k$;
            $\Delta l_k \longleftarrow \Delta l_k - \sum_{i\in\mathcal{N}} \sum_{r=1}^{R} \sum_{\mathbf{j}\in\mathcal{L}_r(i,k)} r x_{i\mathbf{j}k}$;
            $(i \in \mathcal{N};\ k \in \mathcal{K})$
(6) **until** $\theta_{ik}$ stops changing; or in other words, $|\Delta \bar{v}| \le \epsilon$.
```

ALGORITHM 2: Route mutation for delay-bounded traffic.

where, for any switch i in the network, the size of its flow table is bounded by t_i. Thus, the maximum number of flow entries that a switch can install is t_i/a. The left-hand side of inequality (7) shows the number of flow rules of switch i after mutation, which is bounded by t_i/a.

Given Δl_k and the constraints (5), (6a), (6b), and (7), (4a), (4b), (4c), (4d), (4e), and (4f) can be optimally solved using binary branch and bound/cut in the same way, as described in Algorithm 1. After that, both $\bar{v}$ and l_k need to be updated until l_k cannot be further increased. Algorithm 2 summarizes the proposed route mutation approach in the presence of QoS constraints. The key difference from Algorithm 1 is Step (4), implementing (5), (6a), (6b), and (7). The only other changes are in Steps (1) and (5), where Δl_k is initialized and updated, respectively.

5. Computationally Efficient Heuristics

As described in Section 4, binary linear programming is formulated and the binary branch and bound/cut method is required. This has the worst-case complexity of $\mathcal{O}(\min(m^2, n^2) \sum_{k=1}^{n} k2^k)$.

In this section, we propose a simplified version of Algorithms 1 and 2, which decouples traffic allocation and flow selection into two concatenated subproblems. The traffic allocation can be formulated as a linear programming transportation problem. The flow selection can be achieved by using the aforementioned binary branch and bound/cut algorithm, but with a substantially smaller number of variables, that is, less than K. As a result, the complexity can be significantly reduced to the complexity of solving (8a), (8b), (8c), and (8d).

First consider delay-tolerant traffic. We begin with optimizing the total traffic $y_{i\mathbf{j}}$, $i \in \mathcal{N}$, $\mathbf{j} \in \bigcup_{r=1}^{R}(\bigcup_{k=1}^{K} \mathcal{L}_r(i,k))$, which needs to move from one of the nodes, that is, node i, to a detour, that is, detour $\mathbf{j}$ $(i \notin \mathbf{j})$. Following the EMD criterion, a linear programming problem can be formulated to determine the total traffic that needs to detour from the heavily loaded nodes, as given by

$$\min_{\substack{y_{i\mathbf{j}},\\ \forall i\in\mathcal{S},\mathbf{j}}} \left(\sum_{i\in\mathcal{N}} \sum_{r=1}^{R} \sum_{\mathbf{j}\in\bigcup_{k=1}^{K} \mathcal{L}_r(i,k)} c_{i\mathbf{j}} y_{i\mathbf{j}} \right), \tag{8a}$$

$$\text{s.t.} \quad \sum_{r=1}^{R} \sum_{\mathbf{j}\in\bigcup_{k=1}^{K} \mathcal{L}_r(i,k)} y_{i\mathbf{j}} \le v_i - v'_i, \quad \text{for any } i \in \mathcal{N}; \tag{8b}$$

$$\sum_{i\in\mathcal{N}} \sum_{r=1}^{R} \sum_{\mathbf{j}\in\bigcup_{k=1}^{K} \mathcal{L}_r(i,k)} \delta_{j\mathbf{j}} y_{i\mathbf{j}} \le \left[\bar{v} - v'_j\right]^+, \quad \text{for any } j \in \mathcal{N}; \tag{8c}$$

$$\min \left(R \sum_{i\in\mathcal{N}} \left(v_i - v'_i\right), \sum_{j\in\mathcal{N}} \left[\bar{v} - v'_j\right]^+ \right) - \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}} \sum_{r=1}^{R} \sum_{\mathbf{j}\in\bigcup_{k=1}^{K} \mathcal{L}_r(i,k)} \delta_{j\mathbf{j}} y_{i\mathbf{j}} \le \tau; \tag{8d}$$

which can be solved using the Simplex method [36]. The optimal solution for (8a), (8b), (8c), and (8d) is denoted by $\{y^*_{i\mathbf{j}},\ \forall i \in \mathcal{N};\ \mathbf{j} \in \bigcup_{r=1}^{R}(\bigcup_{k=1}^{K} \mathcal{L}_r(i,k))\}$, or $\{y^*_{i\mathbf{j}}\}$ for short. A bisection search can also be taken to identify the minimum value of τ to preserve the feasibility of (8a), (8b), (8c), and (8d).

```
Input: $G(\mathcal{N},\mathcal{E})$; $\mathbf{A}$; $R$; the accumulative traffic of each node
      $v_i'$; the current route of each flow $k \in \mathcal{K}$; $f_k$; and the
      initial $\theta_{ik}$ $(i \in \mathcal{N};\ k \in \mathcal{K})$.
Output: $\theta_{ik}$, $\forall i \in \mathcal{N},\ k \in \mathcal{K}$
(1) Let $v_i = v_i' + \sum_{k\in\mathcal{K}} \theta_{ik} f_k$.
(2) repeat
(3)   Run Steps (3) to (10) of Algorithm 1.
(4)   substitute $v_i$, $\bar{v}$, $v_i'$ $(i \in \mathcal{N})$, $\bar{v}'$, and $c_{\mathbf{ij}}$ into (8a), (8b), (8c), and (8d), and solve the optimal
      solution $y_{i\mathbf{j}}^*$ using the Simplex method;
(5)   Arrange $\{v_i, \forall i \in \mathcal{N}\}$ in decreasing order so that $v_{\pi(1)} \geq v_{\pi(2)} \geq \cdots \geq v_{\pi(N)}$.
(6)   for $i = 1,\ldots,N$ do
(7)     Let $\mathcal{Q} = \bigcup_{r=1}^{R}(\bigcup_{k=1}^{K} \mathcal{L}_r(\pi(i),k))$.
(8)     repeat
(9)         Let $\mathbf{j}^* = \min_{\mathbf{j}\in\mathcal{Q}}((1/|\mathbf{j}|)\sum_{j\in\mathbf{j}} v_j)$;
(10)        Offload traffic flows from node $\pi(i)$ to detour $\mathbf{j}^*$ by solving (9) for $\phi_{\pi(i)\mathbf{j}^*k}$, given $y_{i\mathbf{j}}^*$;
(11)        Update
                $\mathcal{Q} \longleftarrow \mathcal{Q} \setminus \mathbf{j}^*$;
                $v_{\pi(i)} \longleftarrow v_{\pi(i)} - \sum_{k=1}^{K}\sum_{j\in\mathbf{j}^*} \phi_{\pi(i)\mathbf{j}^*k} f_k$;
                $v_j \longleftarrow v_j + \sum_{k=1}^{K} \phi_{\pi(i)\mathbf{j}^*k} f_k, \quad \forall j \in \mathbf{j}^*$.
(12)    until no traffic flow can be further offloaded from node $\pi(i)$, or $\mathcal{Q} = \emptyset$.
(13)  end for
(14) until $|\Delta\bar{v}| \leq \epsilon$ or $y_{i\mathbf{j}}^* = 0\ \forall i, \mathbf{j}$.
```

Algorithm 3: Heuristic route mutation for delay-tolerant traffic.

Given $\{y_{i\mathbf{j}}^*\}$, we proceed to select the traffic flows for redirecting from node i to detour $\mathbf{j}$ and formulate a binary linear program, as given by

$$\begin{aligned} &\max_{\phi_{i\mathbf{j}k}, \forall k\in\mathcal{K}} \quad \left(\sum_{k=1}^{K} \phi_{i\mathbf{j}k} f_k\right), \\ &\text{s.t.} \quad \sum_{k=1}^{K} \phi_{i\mathbf{j}k} f_k \leq y_{i\mathbf{j}}^*; \\ &\qquad \phi_{i\mathbf{j}k} \in \{0,1\},\ \forall k \in \mathcal{K}, \end{aligned} \tag{9}$$

which can be readily solved using binary branch and bound/cut.

We start by offloading the traffic flows of the node with $\max_{i\in\mathcal{N}}(v_i)$. And (9) is used to offload the flows first to the detour with $\min_{\mathbf{j}\in\bigcup_{k=1}^{K}\mathcal{L}_{|\mathbf{j}|}(i,k)}((1/|\mathbf{j}|)\sum_{j\in\mathbf{j}} v_j)$ and then to the other detours in increasing order of $(1/|\mathbf{j}|)\sum_{j\in\mathbf{j}} v_j$. This repeats for the other nodes in decreasing order of v_i.

Algorithm 3 summarizes the proposed computationally efficient heuristic approach for route mutation under delay-tolerant traffic. Steps (3) and (4) execute the traffic allocation, following the EMD criterion and using (8a), (8b), (8c), and (8d). Steps (5) to (13) conduct the flow selection, using (9). These two parts concatenate to reduce the complexity.

The major complexity of Algorithm 3 lies in solving (8a), (8b), (8c), (8d), and (9). Solving (8a), (8b), (8c), and (8d) requires the complexity of $\mathcal{O}(\min(m^2, d^2))$, as discussed in Section 4.1, where $m = 2N + 1$ here. In the worst-case scenario with a complete graph, $d = NL$, where $L = N - 3 + \sum_{r=2}^{R}((N-3)!/2(N-3-r)!)$, as given earlier, N is large, and the number of variables is larger than the number of constraints. Therefore, the complexity of (8a), (8b), (8c), and (8d) is $\mathcal{O}(N^2)$. Given each nonzero solution $y_{i\mathbf{j}}^*$, (9) implements branch and bound/cut. The best-case complexity of solving (9) is $\mathcal{O}(k^4)$, where k is the number of variables in (9), and k is substantially smaller than K. The worst-case complexity is $\mathcal{O}(k^2 2^k)$. Suppose all of the solutions from (8a), (8b), (8c), and (8d) are nonzero. The best-case and worst-case complexities of Algorithm 3 are $\mathcal{O}(k^4N^2)$ and $\mathcal{O}(k^2 2^k N^2)$, respectively.

Proceed with delay sensitive traffic, where the route lengths of traffic flows are strictly bounded. Algorithm 3 can be readily extended to this case by evaluating the route lengths on-the-go. The constraints of bandwidth and switch capacity can also be considered in the same way in Algorithm 2, as discussed in Section 4.2.

To be specific, we can attach (10) to (8a), (8b), (8c), and (8d) to capture the bandwidth constraint.

$$v_i - v_i' - \sum_{\mathbf{j}\in\bigcup_{k=1}^{K}\mathcal{L}_r(i,k)} y_{i\mathbf{j}} + \sum_{j\in\mathcal{N}}\sum_{\mathbf{j}\in\bigcup_{k=1}^{K}\mathcal{L}_r(i,k)} \delta_{i\mathbf{j}} y_{j\mathbf{j}} \leq B_i, \tag{10}$$

for any $i \in \mathcal{N}$;

```
Input: $G(\mathscr{N}, \mathscr{E})$; $\mathbf{A}$; $R$; the accumulative traffic of each node
    $v'_i$; the current route of each flow $k \in \mathscr{K}$; $f_k$; the initial
    $\theta_{ik}$; $L_k$; and the initial $l_k$ $(i \in \mathscr{N};\ k \in \mathscr{K})$.
Output: $\theta_{ik},\ \forall i \in \mathscr{N},\ k \in \mathscr{K}$
(1) Let $v_i = v'_i + \sum_{k\in\mathscr{K}} \theta_{ik} f_k$ and $\Delta l_k = L_k - l_k$.
(2) repeat
(3)   Run Steps (3) to (5) of Algorithm 3
(4)   for $i = 1, \ldots, N$ do
(5)     Let $\mathscr{Q} = \bigcup_{k=1}^{K}(\bigcup_{r=1}^{\min(R,\Delta l_k)} \mathscr{L}_r(\pi(i), k))$.
(6)     repeat
(7)       Run Steps (9) and (10) of Algorithm 3
(8)       Update
              $\mathscr{Q} \longleftarrow \mathscr{Q} \setminus \mathbf{j}^*$;
              $v_{\pi(i)} \longleftarrow v_{\pi(i)} - \sum_{k=1}^{K} \sum_{j\in\mathbf{j}^*} \phi_{\pi(i)\mathbf{j}^*k} f_k$;
              $v_j \longleftarrow v_j + \sum_{k=1}^{K} \phi_{\pi(i)\mathbf{j}^*k} f_k, \quad \forall j \in \mathbf{j}^*$;
              $\Delta l_k \longleftarrow \Delta l_k - \phi_{\pi(i)\mathbf{j}^*k} |\mathbf{j}^*|, \quad \forall k \in \mathscr{K}$.
(9)     until no traffic flow can be further offloaded from
        node $\pi(i)$, or $\mathscr{Q} = \emptyset$.
(10)  end for
(11) until $|\Delta \overline{v}| \le \epsilon$ or $y^*_{i\mathbf{j}} = 0\ \forall i, \mathbf{j}$.
```

Algorithm 4: Heuristic route mutation for delay-bounded traffic.

we can also add (11) to (9) to restrain the number of flow entries per switch i.

$$\sum_{k\in\mathscr{K}} \theta_{ik} - \sum_{k\in\mathscr{K}} \sum_{\mathbf{j}\in\bigcup_{k=1}^{K}\mathscr{L}_r(i,k)} \theta_{ik}\phi_{i\mathbf{j}k} + \sum_{j\in\mathscr{N}} \sum_{\mathbf{j}\in\bigcup_{k=1}^{K}\mathscr{L}_r(i,k)} \sum_{k\in\mathscr{K}} \delta_{i\mathbf{j}}\theta_{jk}\phi_{j\mathbf{j}k} \le \frac{t_i}{a}, \quad \text{for any } i \in \mathscr{N}. \tag{11}$$

The extension is summarized in Algorithm 4, where only the differences between the two algorithms are highlighted.

6. Simulation and Evaluation

In this section, simulations are carried out to evaluate the proposed algorithms. We generate different random k-regular graphs by using Python package NetworkX (NX) [37], where the degrees of all the nodes are k. Nodes have the same probabilities of requesting new routes at any instant. The routes between these nodes are initially generated by the Dijkstra algorithm [38] before our algorithms are carried out. The traffic loads of the routes are randomly and uniformly distributed within $[0, 1]$. The simulations run in a computer with 32 GB RAM and 6 cores of Intel Xeon CPU. We use IBM-CPLEX to solve linear programming problems.

We assume that adversaries can identify strategically important nodes by monitoring historically accumulative traffic of the nodes. Dynamic eavesdropping or interception is considered where, after every interval of monitoring, the eavesdropper acquires network information and updates the identification of targets accordingly. Under this eavesdropping model, we consider different numbers of nodes that the eavesdroppers can identify and the upper bound of traffic difference that eavesdroppers use to identify strategically important nodes. The upper bound reflects the eavesdropping capability and therefore is referred to as "node interception probability." We also consider different time intervals for the eavesdroppings.

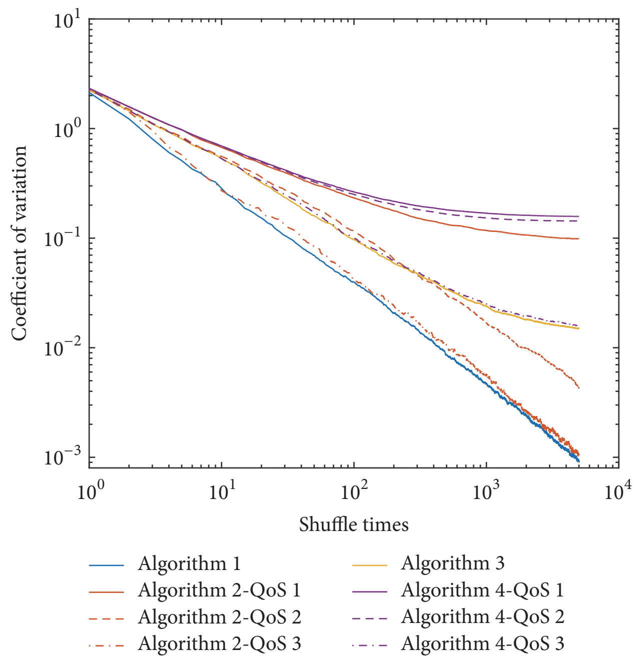


Figure 4: Results of four algorithms that we proposed, with and without QoS constraints. The networks are all generated as 6-regular graph with 50 nodes. The probability of initiating requests is 10%. There are 5000 instants and 50 rounds of iterations.

The performance of the algorithms are measured by two metrics.

(1) Coefficient of Variation (CV). A more disperse distribution of traffic can facilitate eavesdroppers recognizing heavily loaded nodes in network. Coefficient of variation is a metric to evaluate the dispersion of distribution, which is defined as the ratio of standard deviation σ to the mean μ,

$$c_v = \frac{\sigma}{\mu}. \tag{12}$$

(2) The Percentage of Traffic That Bypasses Any Node Being Eavesdropped or Monitored. Figure 4 compares the CV of accumulated traffic between the proposed Algorithms 1, 2, 3, and 4, in a 50-node network with topology of 6-regular graph. The CV changes as the time elapses. In Algorithms 2 and 4, different upper bounds of QoS constraints are considered, that is, 1, 2, and 3 in number of nodes. We see that after routes mutations, the network and variations exhibit significantly reduced variance of accumulative traffic. In other words, the difference of traffic loads among the nodes

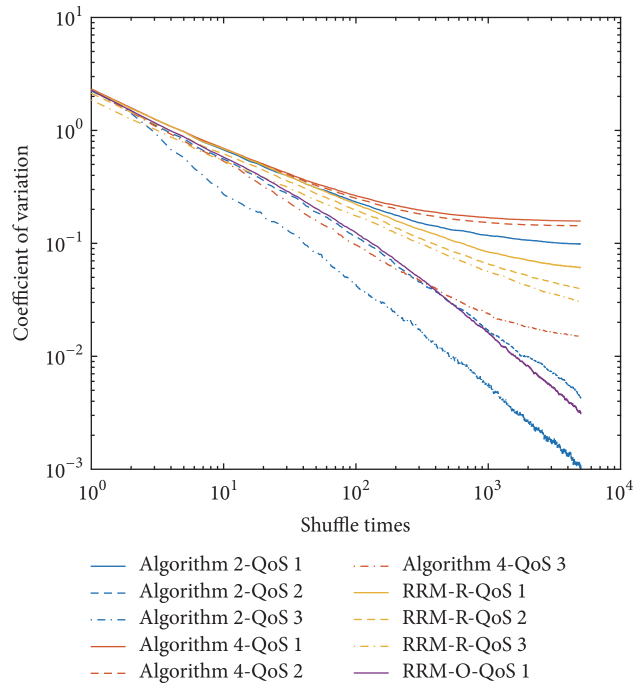


Figure 5: Results of Algorithms 2 and 4; random route mutation methods proposed in [6]. The configurations of this simulation are the same as Figure 4.

fast diminishes. Moreover, with the increasing upper bounds of QoS constraints, Algorithm 2 approaches Algorithm 1.

Figure 5 compares Algorithms 2 and 4. For comparison purpose, we also simulate the random route mutation proposed in [6]. Two cases are considered. The first case is to choose the optimal combination of routes to minimize the CV of traffic at any instant, referred to as "RRM-O." The second case is to randomly choose routes as long as the QoS requirements are met and therefore referred to as "RRM-R." Clearly, RRM-O is optimal in the sense of minimized CV of accumulative traffic. Unfortunately, RRM-O requires exhaustive search of all possible combinations of routes, with a prohibitive complexity. Figure 5 also demonstrates that Algorithm 2 outperforms Algorithm 4 and RRM-R. This is due to the fact that Algorithm 2 carefully designs the routes in a structured way to reduce the CV, where RRM-R does not.

Table 1 shows the average running time for each instant of the proposed algorithms with 50-node 6-regular graph, and the probability to request new routes is 10%. In this table, $|\mathbf{j}|$ represents the length of detour $\mathbf{j}$, that is, the upper bound of QoS. As Algorithms 1 and 3 are designed without QoS constraints, we marked their $|\mathbf{j}|$ as \. Comparing the proposed heuristic methods with the method of [6], Table 1 shows that the increase of the upper bounds of QoS constraints has less influence on our algorithms. RRM-R degrades, when the length of a detour increases from a single hop to two hops. Here, we only plot the tightest QoS constraints of RRM-O in Table 1. This is because the complexity of RRM-O becomes too high to be practical, when the maximum allowed detour length is longer than $(L+1)$ hops.

Table 1: Average computational time for different algorithms.

Algorithms	$\lvert\mathbf{j}\rvert$	Computational time (s)
Algorithm 1	\	0.4870
Algorithm 2	3	0.5370
Algorithm 2	2	0.2826
Algorithm 2	1	0.1132
Algorithm 3	\	0.0775
Algorithm 4	3	0.1048
Algorithm 4	2	0.0218
Algorithm 4	1	0.0190
RRM-O	1	0.0363
RRM-R	3	0.2379
RRM-R	2	0.0518
RRM-R	1	0.0131

As mentioned in Section 3, the eavesdroppers can identify nodes by the differences of accumulative traffic. Figure 6 plots the distribution of "safe" traffic, that is, the traffic that does not pass through any compromised node, under the aforementioned dynamic interception model. In this figure, Algorithm 4 with the constraints of QoS of one-hop increment performs worst. This is because the first step of computing the traffic value to move, as given by (8a), (8b), (8c), and (8d), overlooks flows. As a result, a scenario that, for any $i \in y^*_{i\mathbf{j}}$, there is no detour, that is, $\mathbf{j}$, to move, may occur. When the number of flows increases, for example, the curve of "Algorithm 4-QoS 3" in Figure 6, the probability of flows to move also increases. Therefore it can outperform RRM-R with the upper bound of QoS constraint is three hops, that is, the line of "RRM-R-QoS 3." We can also conclude from this figure that the results of this metric are consistent with Figure 5.

Figure 7 shows the convergent variance for networks with increasing numbers of nodes. The topologies of the networks are 5-regular random graphs. Each line in Figure 7 corresponds to a different probability that nodes are assigned as sources or destinations for some route. We see that, with the increasing network size and the increasing number of routes to mutate, the difference of nodes traffic becomes smaller. Algorithm 3 increasingly approaches Algorithm 1. In other words, the low-complexity heuristic, Algorithm 4, becomes increasingly robust against eavesdroppings.

Figure 8 shows the convergent CV in the networks with different connectivities. The probability of nodes initiating or receiving routing requests is all set to be 10%. Each line corresponds to the nodes having the same degree. As the network size gets larger, the difference of nodes traffic reduces. Figure 9 shows the CV of 40 nodes with connectivities of 3, 4, 5, and 6, respectively. With the increasing of network connectivities, there are more available detours to move to for sources, and the variations are getting smaller.

Figure 10 shows the average running time of the algorithms corresponding to Figure 7. As compared with RRM, our proposed algorithms are far more tolerant against the growth of the network size. As a matter of fact, the convergence time of the proposed algorithms remains almost

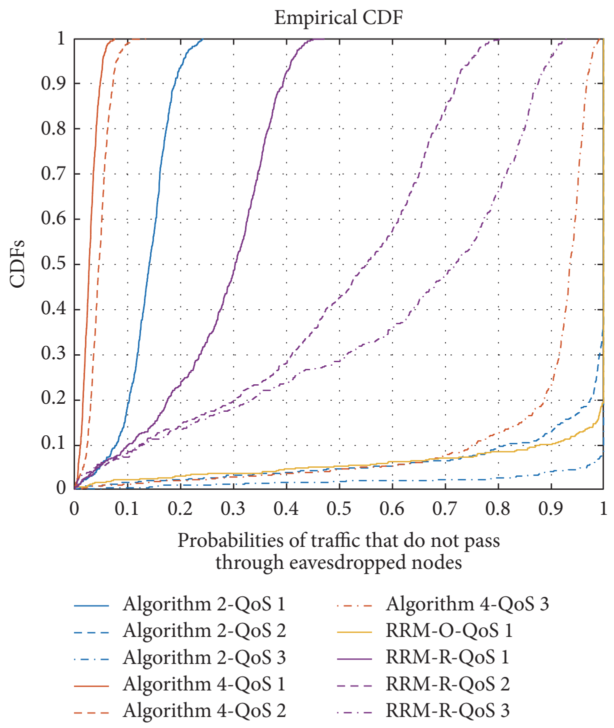


Figure 6: The CDFs of the probabilities of traffic that do not pass through any node which is monitored or eavesdropped. The difference of traffic to identify strategically located nodes is 0.01. The interval of monitoring or eavesdropping is 10.

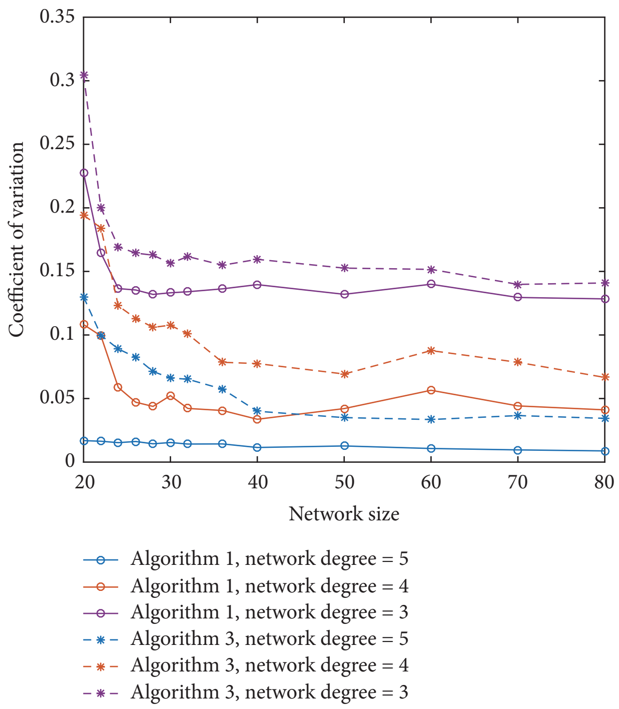


Figure 8: Results of Algorithms 1 and 3 with different sizes of networks and different connectivities.

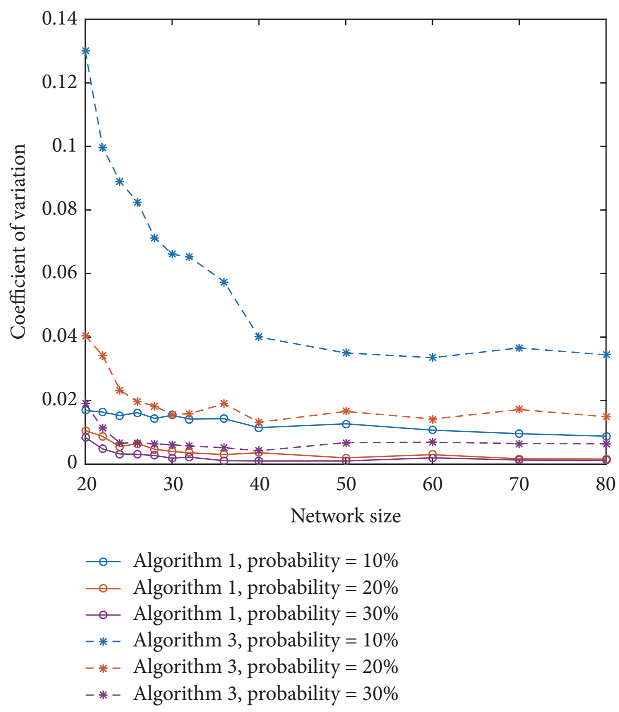


Figure 7: Results of Algorithms 1 and 3 with different networks and different probabilities to choose nodes, where the connectivities of networks are all five.

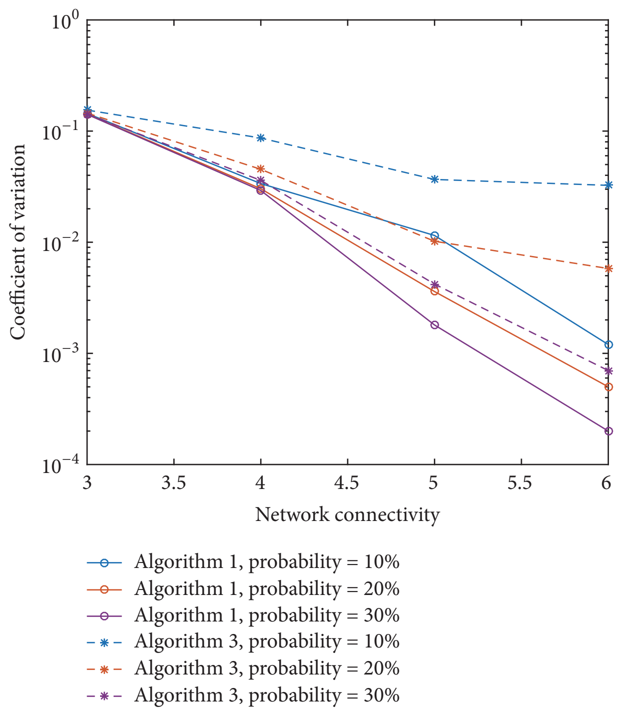


Figure 9: Results of Algorithms 1 and 3 with 40 nodes and different connectivities.

unchanged, as the network grows. In contrast, the RRM algorithms suffer from exponentially increasing complexity and convergence delay. Our methods are time efficient and suitable to larger-scale networks.

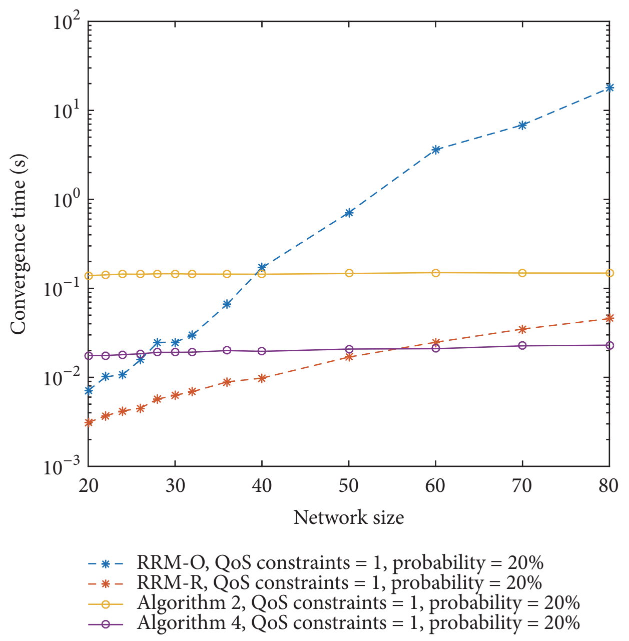


FIGURE 10: Computational time of algorithms that have QoS constraints limitation. The upper bound of QoS constraints is one hop. The probability of nodes to request routes is 20%.

7. Conclusions

In this paper, we propose a node-centric route mutation method which interprets route mutation as a signature matching problem. A three-dimensional EMD model is formulated to match signatures. By this means, routes are mutated by increasingly taking detours, balancing historically accumulated traffic among switches. Heuristic approaches are also developed to significantly reduce the computational complexities of signature matching, enhancing the scalability of route mutation to defend large-scale SDNs. Simulation results show that our methods can disguise strategically located, important switches and increase the difficulties for eavesdroppers to identify the switches, thereby delaying or preventing malicious attacks. Significantly reduced complexities also indicate the suitability of our algorithms to large-scale networks.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by the National Key R&D Program of China (2017YFB0802703) and the National Natural Science Foundation of China (61602052).

References

[1] T. Shu, M. Krunz, and S. Liu, "Secure data collection in wireless sensor networks using randomized dispersive routes," *IEEE Transactions on Mobile Computing*, vol. 9, no. 7, pp. 941–954, 2010.

[2] J. Jafarian, E. Al-Shaer, and Q. Duan, "Formal approach for route agility against persistent attackers," in *Computer Security—ESORICS 2013*, vol. 8134 of *Lecture Notes in Computer Science*, pp. 237–254, Springer, Berlin, Germany, 2013.

[3] S. J. Moore and M. C. Sinclair, "Design of routing tables for a survivable military communications network using genetic algorithms," in *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, pp. 1788–1795, USA, July 1999.

[4] J. Claessens, V. Dem, D. De Cock, B. Preneel, and J. Vandewalle, "On the security of todays online electronic banking systems," *Computers & Security*, vol. 21, no. 3, pp. 253–265, 2002.

[5] M. R. Garey and D. S. Johnson, *A Guide to The Theory of Np-Completeness*, WH Freemann, New York, NY, USA, 1979.

[6] Q. Duan, E. Al-Shaer, and H. Jafarian, "Efficient Random Route Mutation considering flow and network constraints," in *Proceedings of the IEEE Conference on Communications and Network Security (CNS '13)*, pp. 260–268, IEEE, National Harbor, Md, USA, October 2013.

[7] L. de Moura and N. Bjørner, "Satisfiability modulo theories: an appetizer," in *Formal Methods: Foundations and Applications*, vol. 5902 of *Lecture Notes in Computer Science*, pp. 23–36, Springer, Berlin, Germany, 2009.

[8] J. M. Jaffe, "Algorithms for finding paths with multiple constraints," *Networks*, vol. 14, no. 1, pp. 95–116, 1984.

[9] R. Hassin, "Approximation schemes for the restricted shortest path problem," *Mathematics of Operations Research*, vol. 17, no. 1, pp. 36–42, 1992.

[10] T. Korkmaz and M. Krunz, "Multi-constrained optimal path selection," in *Proceedings of the IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society*, pp. 834–843, Anchorage, Alaska, USA.

[11] R. G. Garroppo, S. Giordano, and L. Tavanti, "A survey on multi-constrained optimal path computation: Exact and approximate algorithms," *Computer Networks*, vol. 54, no. 17, pp. 3081–3107, 2010.

[12] A. Paulos, P. Pal, R. Schantz, and B. Benyo, "Moving target defense (MTD) in an adaptive execution environment," in *Proceedings of the the Eighth Annual Cyber Security and Information Intelligence Research Workshop*, Oak Ridge, Tenn, USA, January 2013.

[13] E. Al-Shaer, Q. Duan, and J. H. Jafarian, "Random host mutation for moving target defense," in *Security and Privacy in Communication Networks*, A. D. Keromytis and R. Di Pietro, Eds., vol. 106 of *Lecture Notes of the Institute for Computer Sciences*, pp. 310–327, Springer, Berlin, Germany, 2013.

[14] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: transparent moving target defense using software defined networking," in *Proceedings of the 1st Workshop on Hot Topics in Software Defined Networks (HotSDN '12)*, pp. 127–132, ACM, Helsinki, Finland, August 2012.

[15] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "An effective address mutation approach for disrupting reconnaissance attacks," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2562–2577, 2015.

[16] S. C. Brailsford, C. N. Potts, and B. M. Smith, "Constraint satisfaction problems: algorithms and applications," *European Journal of Operational Research*, vol. 119, no. 3, pp. 557–581, 1999.

[17] C. Frei, B. Fallings, and M. Hamdi, “Resource allocation in communication networks using abstraction and constraint satisfaction,” *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 2, pp. 304–320, 2005.

[18] M. Dunlop, S. Groat, W. Urbanski, R. Marchany, and J. Tront, “MT6D: a moving target IPv6 defense,” in *Proceedings of the Military Communications Conference (MILCOM ’11)*, pp. 1321–1326, IEEE, Baltimore, Md, USA, November 2011.

[19] R. Moore, S. Groat, R. Marchany, and J. Tront, “Using transport layer multihoming to enhance network layer moving target defenses,” in *Proceedings of the 8th Annual Cyber Security and Information Intelligence Research Workshop: Federal Cyber Security R and D Program Thrusts, CSIIRW 2013*, USA, January 2013.

[20] M. Albanese, A. De Benedictis, S. Jajodia, and K. Sun, “A moving target defense mechanism for MANETs based on identity virtualization,” in *Proceedings of the 1st IEEE International Conference on Communications and Network Security, CNS 2013*, pp. 278–286, USA, October 2013.

[21] P. Kampanakis, H. Perros, and T. Beyene, “SDN-based solutions for Moving Target Defense network protection,” in *Proceedings of the 15th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM ’14)*, pp. 1–6, Sydney, Australia, June 2014.

[22] S. Achleitner, T. La Porta, P. McDaniel, S. Sugrim, S. V. Krishnamurthy, and R. Chadha, “Cyber deception: Virtual networks to defend insider reconnaissance,” in *Proceedings of the 8th ACM CCS International Workshop on Managing Insider Security Threats, MIST 2016*, pp. 57–68, Austria.

[23] Z. Zhao, D. Gong, B. Lu, F. Liu, and C. Zhang, “SDN-based double hopping communication against sniffer attack,” *Mathematical Problems in Engineering*, vol. 2016, Article ID 8927169, 13 pages, 2016.

[24] S. Bohawek, J. P. Hespanha, J. Lee, C. Lim, and K. Obraczka, “Game theoretic stochastic routing for fault tolerance and security in computer networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 9, pp. 1227–1240, 2007.

[25] Z. Wang, J. Wu, G. Cheng, and Y. Jiang, “Mutine: a mutable virtual network embedding with game-theoretic stochastic routing,” in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, San Diego, Calif, USA, December 2015.

[26] M. Zalewski, *Silence on the wire: a field guide to passive reconnaissance and indirect attacks*, Starch Press, 2005.

[27] T. Urban, *Cacti 0.8 beginner’s guide*, Packt Publishing Ltd, 2011.

[28] J. Moy, “OSPF Version 2,” 1997.

[29] R. E. Tarjan, “Depth-first search and linear graph algorithms,” *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146–160, 1972.

[30] D. Zhang and G. Lu, “Evaluation of similarity measurement for image retrieval,” in *Proceedings of the 2003 International Conference on Neural Networks and Signal Processing, ICNNSP’03*, pp. 928–931, China, December 2003.

[31] Y. Rubner, C. Tomasi, and L. J. Guibas, “Earth mover’s distance as a metric for image retrieval,” *International Journal of Computer Vision*, vol. 40, no. 2, pp. 99–121, 2000.

[32] J. E. Mitchell, “Branch-and-cut algorithms for combinatorial optimization problems,” in *Handbook of Applied Optimization*, pp. 65–77, 2002.

[33] I. Maros, *Computational techniques of the simplex method*, vol. 61, Springer Science & Business Media, 2012.

[34] N. Megiddo, *On the Complexity of Linear Programming*, IBM Thomas J. Watson Research Division, 1986.

[35] I. Adler and N. Megiddo, “A simplex algorithm whose average number of steps is bounded between two quadratic functions of the smaller dimension,” *Journal of the ACM*, vol. 32, no. 4, pp. 871–895, 1985.

[36] W. L. Winston, M. Venkataramanan, and J. B. Goldberg, *Introduction to Mathematical Programming*, vol. 1, Thomson/Brooks/Cole Duxbury, Pacific Grove, Calif, USA, 2003.

[37] A. A. Hagberg, D. A. Schult, and P. J. Swart, “Exploring network structure, dynamics, and function using NetworkX,” in *Proceedings of the 7th Python in Science Conference (SciPy 2008)*, pp. 11–15, Pasadena, Calif, USA, 2008.

[38] S. Skiena, “Dijkstra’s algorithm,” in *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*, Addison-Wesley, Reading, Mass, USA, 1990.