Optimal Deployment of SRv6 to Enable Network Interconnection Service

Bangbang Ren, Deke Guo*, Yali Yuan, Guoming Tang, Weijun Wang, Xiaoming Fu

***Abstract*—Many organizations nowadays have multiple sites at different geographic locations. Typically, transmitting massive data among these sites relies on the interconnection service offered by ISPs. Segment Routing over IPv6 (SRv6) is a new simple and flexible source routing solution which could be leveraged to enhance interconnection services. Compared to traditional technologies, e.g., physical leased lines and MPLS-VPN, SRv6 can easily enable quick-launched interconnection services and significantly benefit from traffic engineering with SRv6-TE. To parse the SRv6 packet headers, however, hardware support and upgrade are needed for the conventional routers of ISP. In this paper, we study the problem of SRv6 incremental deployment to provide a more balanced interconnection service from a traffic engineering view. We formally formulate the problem as an SRID problem with integer programming. After transforming the SRID problem into a graph model, we propose two greedy methods considering short-term and long-term impacts with reinforcement learning, namely GSI and GLI. The experiment results using a public dataset demonstrate that both GSI and GLI can significantly reduce the maximum link utilization, where GLI achieves a saving of 59.1% against the default method.**

I. Introduction

Nowadays, it is common that big organizations or companies have many subsidiaries, and each subsidiary has its own private local area network (LAN). To share information and communicate with each other more easily, these organizations have to connect their geo-distributed LANs together. Fig. 1 gives an example where four branch offices of an organization reside in different geo-locations. One easiest way to accomplish this goal is to connect these LANs to the public Internet and assign them public IP addresses. However, this method exposes the private network to the public. Some organizations, e.g., bank and defense departments, have special security considerations and hope to isolate their private data [1], [2]. To solve this problem, Internet Service Providers (ISPs) have provided the following alternative methods.

The first method is using physical leased lines to connect the required LANs directly. Though this method can provide the highest security level and performance, renting physical lines can be prohibitively expensive for customers [3]. The

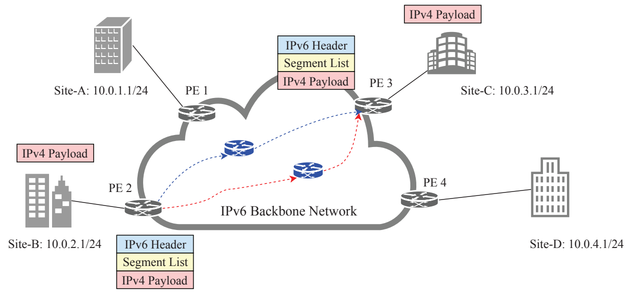


Fig. 1. An example to show the demand of interconnection service.

second method is using network technologies to build a virtual circuit in public networks. For example, ISPs usually use multi-protocol label switching (MPLS) to create a virtual private network (VPN) in public networks. Though MPLS is cheaper than renting physical lines and can also provide good performance through traffic engineering, it needs to distribute many labels to switch devices and maintain many state variables [4]. Another method is using encapsulation technologies, e.g., GRE [5] and IPSec [6], which add headers to the payloads from a customer's LAN such that they can be recognized and forwarded in public networks. Compared with MPLS, encapsulation technologies do not need to maintain states in the network and are easy to implement. However, encapsulation technologies usually rely on the best effort routing protocol, which is not friendly to traffic engineering.

Along with the worldwide deployment of IPv6, the aforementioned interconnection technologies are expected to adapt to the new data plane, which leads to the birth of IPv6-only MPLS [7] and IPv6 with IPSec [8]. On the other hand, the new data plane also calls for new interconnection technologies, which leads to the birth of Segment Routing over IPv6 (SRv6) [9]. The key idea of segment routing is to break up the routing path into multiple segments in order to enable better network utilization. The segments are represented by labels, which can be attached to packet headers. The details of SRv6 will be left in Sec. II. Here, we just show how to leverage SRv6 to connect isolated sites across an ISP network which is composed of access networks and a transport backbone network. As shown in Fig. 1, PE 1, PE 2, PE 3 and PE 4 represent the border routers of the access network. Site-B wants to send its packets to site-C crossing an IPv6 transport backbone network. With SRv6, the packets from site-B will be encapsulated with two kinds of headers in PE 2, i.e., an IPv6 header and a segment routing header which includes a segment list. Note that the encapsulated packets could be forwarded by ordinary IPv6 routers since they contain IPv6 headers. Thus, this encapsulation could

B. Ren and D. Guo are with the Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha Hunan 410073, P. R. China. Email: {renbangbang11,dekeguo}@nudt.edu.cn.

Y. Yuan, and X. Fu are with the University of Göttingen, 37077, German. Email:{yali.yuan, fu}@cs.uni-goettingen.de

G. Tang is with the Peng Cheng Laboratory, Shenzhen, Guangdong, 518055, P.R. China. Email: tanggm@pcl.ac.cn

W. Wang is with the Nanjing University, 210093, P.R. China. Email: wangalexweijun@gmail.com

Corresponding author: Deke Guo.

connect the sites successfully. Supporting traffic engineering by segment lists is the biggest benefit of SRv6, which differs from traditional encapsulation technologies. The SRv6-BE strategy which means forwarding the SRv6 packets with the best effort could accomplish the interconnection tasks [10]. Nevertheless, the network sometimes can be congested, and we need to reroute the flows to relieve the congestion. In Fig. 1, we can steer the packets along the red path or blue path with a segment list to balance the traffic.

One benefit of SRv6 is that we could easily set up a new routing path for each flow by combining several segments, which is the so-called SRv6-Traffic Engineering (SRv6-TE) [9]. Since only SRv6-enabled routers can parse segment lists, network operators need to upgrade their devices to support SRv6. However, as it happens with most novel network protocols and architectures, a "hard" transition from a pure IPv6 network to a full SRv6 network at once is nearly impossible due to the typically huge number of routers [11]. Hence, a "soft" transition, i.e., upgrading a sub-set of IPv6 routers, is what can be expected. Usually, operators upgrade their networks in traffic engineering view, which means that the network could be utilized as much as possible [11]–[13]. It remains an open problem that what the best practices are to upgrade the SRv6 network incrementally with the limit of the maximum number of upgradeable routers. In this paper, we devote to studying this "soft" transit problem, i.e., **SR**v6 **I**ncremental **D**eployment problem (SRID). In detail, given a candidate router list and the maximum number of upgradeable routers, we need to decide *which routers should be upgraded to accomplish the optimal traffic engineering goal?* After upgrading the related routers, we still need to decide *how to generate routing paths for flows to accomplish the optimal traffic engineering goal?*

In reality, the network topology and the number of flows can be much larger, which leads to a huge space of SRv6 incremental deployment solutions. This paper aims to design efficient methods to solve the SRID problem. We make the following contributions:

- We formally define the SRID problem and formulate it with integer programming. We also prove that the SRID problem is NP-hard.
- We transform the SRID problem into a graph model and then give a polynomial-time greedy algorithm named GSI, which focuses on short-term impact.
- We design a method focusing on long-term impact (GLI), which leverages reinforcement learning to solve the SRID problem from an end-to-end perspective. This framework could be trained with small-scale problem instances and then be applied to large-scale ones.
- We investigate the performance of our methods under different parameter settings. With extensive experiments, we demonstrate that both GSI and GLI methods can significantly reduce the link utilization, with the GLI method cutting down the maximum link utilization by 59.1% against the default shortest path routing method.

The rest of the paper is structured as follows. In Section II, we introduce the background of SRv6 and review the related works. We formulate the SRID problem with integer programming in Section III. We design the GSI method in Section IV and the GLI method in Section V. Section VI presents performance evaluations on the proposed methods and sensitivity analysis to the parameter settings. Section VII gives some further discussion and Section VIII concludes the paper.

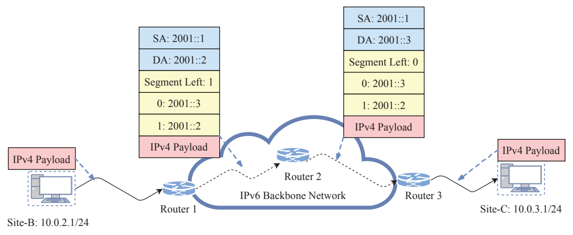


Fig. 2. An example to show interconnection service provided by SRv6. The IPv6 addresses of the three routers are 2001::1, 2001::2 and 2001::3, respectively.

II. Background and Related work

A. Segment Routing over IPv6

The key idea of segment routing is to break up the routing path into segments in order to enable better network utilization. There are two methods to implement segment routing, i.e., SR-MPLS and SRv6. Compared to SR-MPLS, SRv6 has many more special characteristics and benefits [14]. In this paper, we concern about segment routing over IPv6 data plane. Fig. 2 gives an illustrative example of SRv6. Initially, to transmit the IPv4 payload from Site-B to Site-C, Router 1 will encapsulate the packet with an IPv6 header where Router 1 and Router 3's IPv6 address will act as the source address and the destination address, respectively. Then the packet could be transmitted in the IPv6 network in a best-effort way. As a comparison, SRv6 will add an extra header between IPv6 header and IPv4 payload, i.e., segment routing header (SRH). There are two key field types in SRH: segment left denoting the current activated segment and segment lists denoting segments. With SRH, the packet could be routed along an expected path. When Router 1 receives the packet from Site-B, it will add two segments into SRH as shown in Fig. 2. Since the value of the segment left field equals 1, the segment list [1] will be activated and be copied into the destination address field, and then the packet will be forwarded to Router 2 along the best-effort path between Router 1 and Router 2. It should be emphasized that there may be multiple routers along the path between Router 1 and Router 2. If the routers do not support SRv6, they will forward the packet according to the destination address. If the routers can parse SRH, since they are not the destination of the packet, they will still forward the packet according to the destination address. When Router 2 receives the packet, it will decrease the value of the segment left field to zero and activate segment list [0] and then forward it toward 2001::3. Finally, when the packet reaches Router 3, it will be decapsulated since the value of the segment left field is zero and then forwarded to Site-C.

From the above illustration, we can know that SRv6 can bring two main functions to interconnection service. Firstly, SRv6 can build tunnels among the sites by encapsulation. Secondly, SRv6 can steer the flow along the selected paths to

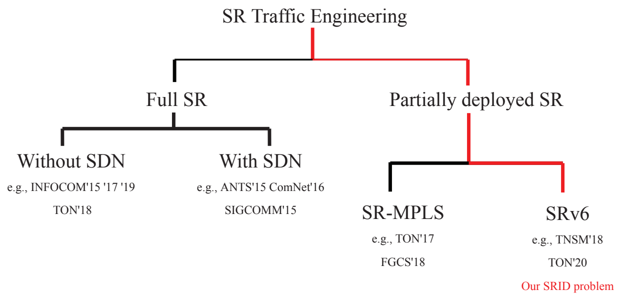


Fig. 3. Classification of works about leveraging SR in traffic engineering.

some extent, which can be leveraged to do traffic engineering. In fact, the combination of segments can represent any path. However, too many segments will incur extra header costs. Actually, some related works exposed that 2-segment routing could provide performance as nearly good as n-segments in the view of traffic engineering [15], [16]. Therefore in this paper, we primarily focus on 2-segment routing where the routing path for a flow will be composed of at most two segments[1].

B. Work Related To This Paper

In most cases, the migration to new network protocols and architectures cannot be finished at once. In [17], the authors proposed an incremental deployment solution for IPv6 over IPv4. SDN is also no exception to the concept of incremental deploying. In [11], the authors studied the SDN incremental upgrading problem that is devoted to figuring out which router should be upgraded. Their solution was made in the preference of empowering advanced traffic engineering. Our work applies this migration rule into the SRv6 network with the hope to provide better network interconnection service from a traffic engineering view.

Existing literature about leveraging segment routing in traffic engineering includes two categories: full SR domain and partially deployed SR, as shown in Fig. 3.

Traffic engineering in full SR domain: The works in this category require that all the concerned nodes support segment routing. Bhatia et al. used integer linear programming to model 2-segment routing in traffic engineering, where any logical path contains one middlepoint and thus two segments [15]. To quickly react to unexpected traffic changes and failures, Hartert et al. proposed a new approach based on local search [18]. In [19], the authors proposed an algorithm based on column generation to solve the large-scale linear problems with guaranteeing the theory gap bound. In [16], the authors evaluated segment routing mechanism to do traffic engineering in real-world topologies and traffic demands. They also pointed out that 2-SR could be near-optimal as far as minimizing link utilization and additional intermediate segments are not profitable for basic SR.

The combination of SDN and SR can improve the efficiency of traffic engineering. In [20], the authors used the SDN controller to accomplish an online energy-efficient traffic engineering method that dynamically adapts the number of powered-on links to the traffic load. Literature [21] leveraged SR and SDN to provide bandwidth-guaranteed paths as well as minimize the possibility of rejecting traffic demands. Renaud et al. built a system named DEFO which combines the declarativity of SDN and expressiveness of SR to optimize the traffic transmission in large-scale carrier network [22].

Traffic engineering in partially deployed SR domain: It is necessary to consider backward compatibility when deploying new network technologies in a production environment, e.g., SDN, IPv6 and SR. In [12], the authors first focused on the incremental deployment of an SR-MPLS network. In their work, they proposed to embed several SR domains into the network and leveraged encapsulation to guarantee the proper routing between SR domain and normal IP routers. In the SR-MPLS network, if the path between two routers covers one or more SR domain, then the packet would follow the IP rules when it traverses within the IP domain, while it is encapsulated into an SR packet every time it crosses an SR domain. However, when the SR network uses IPv6 as its data plane, additional encapsulation will never be used due to that both SRv6-enabled routers and non-SRv6 routers can forward the packet under the same packet header definition. In [23], the authors proposed an architecture for SRv6 network and designed a series of southbound APIs. In this paper, we focus on how to incrementally deploy SRv6-enabled routers into the legacy network from a traffic engineering view, i.e., minimizing the maximum link utilization under the constraint of a limited number of SRv6 routers.

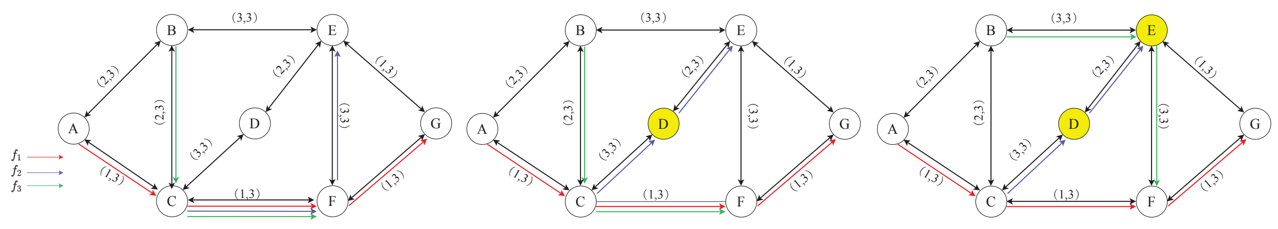
As the most relevant work to ours, Tian et al. [24] studied the problem of optimizing link weights in a partially deployed SRv6 network to minimize the most congested links. They first leveraged three different heuristic rules to enable the SRv6 nodes and then leveraged a method based on reinforcement learning to adjust the link weights and recalculate the routing paths. However, the method of adjusting OSPF link weights may influence the routing paths of background traffic, which may have different optimality criteria [25]. In this work, we leverage SRv6 to combine the shortest paths built by the default OSPF to do traffic engineering, which can be seen as an overlay routing [26]. With this routing mechanism in consideration, we study the incremental deployment problem of SRv6 for providing better network interconnection service.

III. Problem Formulation

A. Problem Illustration

Fig. 4 gives a simple example to illustrate the SRID problem. As shown in Fig. 4, there are seven nodes $\{A, B, C, D, E, F, G\}$ in the transport network and three flows $\{f_1, f_2, f_3\}$. Each flow f_i can be represented by a three-tuple (s_i, d_i, λ_i) in which s_i, d_i and λ_i denote flow source, flow destination and flow size, respectively. Without loss of generality, we assume that each flow has an unit size, then $f_1 = \{A, G, 1\}$, $f_2 = \{C, G, 1\}$ and $f_3 = \{B, F, 1\}$. Each link in the network has a two-tuple attribute (w_e, c_e) in which w_e represents the link cost and c_e represents the link capacity. Fig. 4(a) shows the case that when all flows traverse along the shortest paths, then the maximum link utilization will be $(1 + 1 + 1)/3 = 100\%$. As shown in Fig. 4(b), if the node

[1]Actually, the graph model in Sec. IV-A indicates that our model could be easily extended to the n-segments case. This part is discussed in Sec. VII



(a) Without SRv6 router as intermediate node: the bottleneck link is e_{CF} which has the 100% link utilization.

(b) With one SRv6 router as intermediate node: the bottleneck link is e_{CF} which has the 66.6% link utilization.

(c) With two SRv6 routers as intermediate nodes: the bottleneck link is e_{CF} which has the 33.3% link utilization.

Fig. 4. Three routing solutions with different maximum link utilization using different numbers of SRv6-enable routers. The two-tuple near each edge represents (cost, bandwidth).

D is upgraded to enable SRv6, then we can first steer f_2 to D then to E. In this example, the shortest path between C and D only occupies one link, i.e., e_{CD}. Also, the link e_{DE} is the shortest path connecting D and E. As a result, the maximum link utilization in Fig. 4(b) is $(1+1)/3 = 66.6\%$. Similarly, when node E is also upgraded to enable SRv6, we can steer f_3 to E before G. In this case, the maximum link utilization will be $1/3 = 33.3\%$. From the above example, we can easily find that the number of SRv6 enabled nodes and their locations have great impacts on minimizing the maximum link utilization.

From the above example, we can define our **S**Rv6-enabled **R**outer **I**ncremental **D**eployment problem (**SRID**) as follows.

Definition 1: Given a transport network $G = (V, E, W, C)$ and multiple flows $\{f_1, f_2, ..., f_n\}$. Each flow can be routed along the shortest path between its source and destination or using 2-segments routing. Assume that there is a candidate router set H and at most γ routers can be upgraded as SRv6-enabled routers, then the SRID problem is to decide which nodes should be upgraded such that the maximum link utilization of all links is minimized.

To solve the SRID problem, we must decide: *(i) How many routers should be upgraded? (ii) Where to deploy these SRv6 enabled routers? (iii) How to assign routings to all the flows such that the maximum link utilization of all links is minimized?*

B. Problem Model

To ease the presentation, we list the notation in Table II. It is noted that each segment can leverage ECMP naturally [15]. That is, when there are multiple shortest paths in one segment, the flows will be divided evenly into these paths. Let $SP(v_i, v_j)$ represent the set of shortest paths between v_i and v_j. For any edge $\forall e_{mn} \in E$, it may occur several times in $SP(v_i, v_j)$. We use $\pi_{ie_{mn}}$ to denote the amount of traffic on link e_{mn} when unit flow f_i is routed along the shortest path between its source and destination. To calculate $\pi_{ie_{mn}}$, we need to count how many times e_{mn} occur in $SP(s_i, d_i)$, which could be denoted by $|e_{mn}^{s_i d_i}|$. Then we can get

$$\pi_{ie_{mn}} = \frac{|e_{mn}^{s_i d_i}|}{|SP(s_i, d_i)|}, \forall e_{mn} \in E, \forall f_i, \tag{1}$$

where $|SP(s_i, d_i)|$ denotes the cardinality of the set $SP(s_i, d_i)$. $\pi_{ie_{mn}}$ will be fractional if ECMP is used. If the flow f_i uses 2-segment routing with an intermediate SRv6 router v_k, then we can use variable $\pi_{ie_{mn}}^{v_k}$ to denote the amount of traffic on link e_{mn}. It can be calculated as follows

$$\pi_{ie_{mn}}^{v_k} = \frac{|e_{mn}^{s_i v_k}|}{|SP(s_i, v_k)|} + \frac{|e_{mn}^{v_k d_i}|}{|SP(v_k, d_i)|}, \forall e_{mn} \in E, v_k \in H, f_i. \tag{2}$$

Specially, to unify the representation, we add a virtual SRv6 router H_0 into the network and let $H' = H \bigcup \{H_0\}$. When a flow uses H_0 as its intermediate node, it means that the flow will be routed along the default shortest path, i.e., $\pi_{ie_{mn}}^{H_0} = \pi_{ie_{mn}}$. We use a binary variable x_{ik} to denote whether f_i is routed through v_k, i.e., an intermediate SRv6 router. The variable θ represents the maximum link utilization, then we have

$$\sum_{f_i} \sum_{v_k \in H'} \pi_{ie_{mn}}^{v_k} \lambda_i x_{ik} \le \theta c_{mn}, \forall e_{mn} \in E. \tag{3}$$

To make sure that all flows are delivered to their destinations, we have

$$\sum_{v_k \in H'} x_{ik} = 1, \forall f_i. \tag{4}$$

We use a binary variable ω_k to denote whether node v_k is selected and upgraded to SRv6 router. Then we have

$$x_{ik} \le \omega_k, \forall f_i, \forall v_k \in H. \tag{5}$$

TABLE I
SUMMARY OF NOTATIONS.

Notation	Description
G	network
V	node set of G
E	link set of G
H	candidate router set
w_{ij}	the cost of link e_{ij}
c_{ij}	the capacity of link e_{ij}
f_i	the i^{th} flow request
γ	the number of routers that can be upgraded
h_k	whether node v_k has already been upgraded
s_i	the source of f_i
d_i	the destination of f_i
λ_i	the size of f_i
$SP(v_i, v_j)$	the set of shortest paths between v_i and v_j
$\lvert e_{mn}^{v_i v_j}\rvert$	denote the frequency of e_{mn} in $SP(v_i, v_j)$
$\pi_{ie_{mn}}$	the fractional traffic on e_{mn} if f_i is routed along the shortest path between s_i and d_i
$\pi_{ie_{mn}}^{v_k}$	the fractional traffic on e_{mn} if f_i selects v_k as intermediate node
x_{ik}	denote if f_i routes through v_k
ω_k	denote if node v_k is upgraded as SRv6 router
θ	the maximum link utilization

Definition 1 imposes that there are at most γ nodes that can be upgraded as SRv6 routers. To make our model more general which can be applied in the different stages of incremental deployment, we assign a binary variable h_k to denote whether node v_k has already been upgraded in previous stages. Thus, we have

$$\sum_{v_k \in H} \omega_k \times (1 - h_k) \leq \gamma. \tag{6}$$

With the aforementioned constraints in mind, we can model the SRID problem with Integer Linear Programming (ILP) as follows,

$$(ILP) \qquad \min_{x_{ik}, \omega_k} \quad \theta \tag{7a}$$

$$\text{s.t.} \quad (1) - (6). \tag{7b}$$

C. Complexity Analysis

To prove that our SRID problem is NP-hard, we first give the definition of a classic NPC problem, i.e., the subset sum problem [27]:

Definition 2: Given a set of integers $W = \{w_1, w_2, ..., w_m\}$, the *subset sum problem* is to decide whether there exists a subset $A \subsetneq W$ such that $\sum A = \frac{\sum W}{2}$.

Theorem 1: The SRID problem formulated in Model (7) is NP-hard.

Proof: Assume that there is a subset sum instance, e.g., $W = \{w_1, w_2, ..., w_m\}$, we could construct an instance of the SRID problem from this subset sum problem instance. As shown in Fig. 5, there are nine nodes. The links related to node M are bidirectional while others are unidirectional. The two-tuple attached to each link represents its weight and capacity in which $M \gg \sum W$. There are k flows, i.e., (A, G, w_1), (A, G, w_2), ..., (A, G, w_k), needed to be transmitted from A to G. Also, there are another $n - k$ flows needed to be transmitted from B to H, i.e., (B, H, w_{k+1}), (B, H, w_{k+2}), ..., (B, H, w_m). Also, G and H can be selected as SRv6-enabled routers. This means that the flows that hope to visit G can be first steered to H and vice versa.

At beginning, the flows would be routed along the shortest path. Thus, all the flows from A will route along $A \rightarrow C \rightarrow E \rightarrow G$ and all the flows from B will route along $B \rightarrow D \rightarrow F \rightarrow H$. Then the maximum link utilization will be $max(\frac{\sum_{i=1}^{k} w_i}{\sum W}, \frac{\sum_{i=k+1}^{m} w_i}{\sum W})$. If we could make G and H support SRv6, then we could steer some flows between A and G through H using 2-segments routing, i.e., $A \xrightarrow{D,F,H,M} G$. Similarly, flows between B and H could route through G with 2-segments routing. Finally, if we could solve the SRID problem in the constructed example optimally, then we could solve the subset sum problem. If the link utilization of e_{CE} and e_{DF} are 50%, then the answer to the subset problem is yes, otherwise not. However, the subset problem is NPC, which shows that the SRID problem is also NP-hard. Thus, Theorem 1 is proved. ■

IV. Solution I: Greedy with Short-term Impact

The example in Fig. 4 shows that the number of candidate SRv6 nodes and their locations play key roles in our SRID

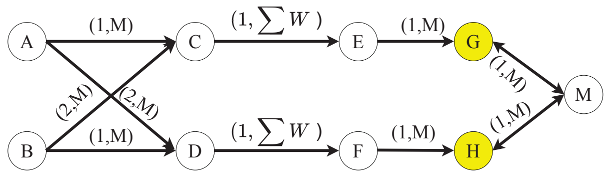


Fig. 5. An illustrative SRID example originated from subset sum problem.

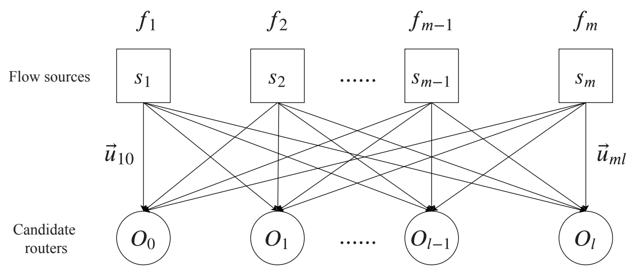


Fig. 6. Rethinking the SRID problem from the graph model view.

problem. Theorem 1 indicates that we cannot find the optimal solution in polynomial time. Thus, we propose to find the solution in a sequential way that decides the SRv6 nodes one by one without violating the constraints. In this section, we first transform the SRID problem from the graph theory view and then give a greedy algorithm.

A. Model Transformation

Dividing both sides of Equation (3) with c_{mn}, then we will get

$$\sum_{i} \sum_{v_k \in H'} \frac{\pi_{ie_{mn}}^{v_k} \lambda_i x_{ik}}{c_{mn}} \leq \theta, \forall e_{mn} \in E. \tag{8}$$

Let vector variable $\vec{u}_{ik} = \{\frac{\pi_{ie_{mn}}^{v_k} \lambda_i}{c_{mn}} | \forall e_{mn} \in E\}$, then Equation (8) could be rewritten as

$$||\sum_{i} \sum_{v_k \in H'} \vec{u}_{ik} x_{ik}||_\infty \leq \theta. \tag{9}$$

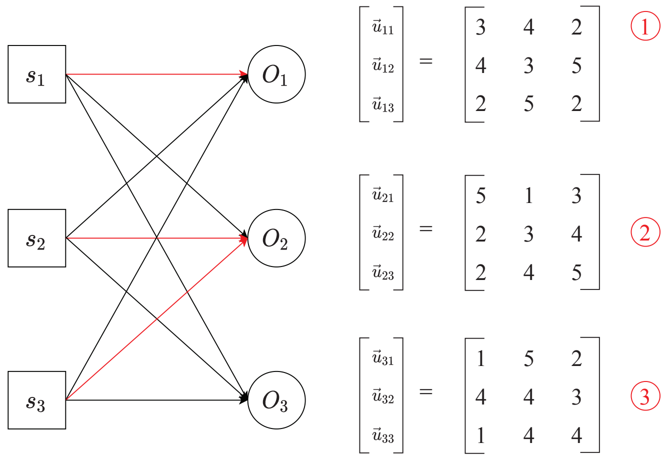
Then the model (7) is equal to

$$\min_{x_{ik}, \omega_k} \quad ||\sum_{i} \sum_{v_k \in H'} \vec{u}_{ik} x_{ik}||_\infty \tag{10a}$$

$$\text{s.t.} \quad (1), (2), (4), (5), (6). \tag{10b}$$

The above model can be reviewed from a graph model view. As shown in Fig. 6, there are m nodes on top representing the source nodes of flows and $|H| = l$ nodes on the bottom representing the candidate routers. O_l represents the virtual router. Each link has a vector weight $\vec{u}_{ik}$ representing the utilization of all links when f_i is forwarded by O_k. The vector weights attaching to virtual router O_l represent the utilization of all links when the corresponding flows are delivered along the shortest paths. Model (10) indicates that we need to find a subgraph that satisfies the following three constraints,

- The degree of each source node on the top is one.
- There are at most γ candidate routers on the bottom are covered in the subgraph.
- The sum vector of all the weights in the subgraph should be minimized in the term of its infinite norm.



s_1
s_2
s_3
O_1
O_2
O_3
$\begin{bmatrix}\vec{u}_{11}\\ \vec{u}_{12}\\ \vec{u}_{13}\end{bmatrix} = \begin{bmatrix}3&4&2\\4&3&5\\2&5&2\end{bmatrix}$ ①
$\begin{bmatrix}\vec{u}_{21}\\ \vec{u}_{22}\\ \vec{u}_{23}\end{bmatrix} = \begin{bmatrix}5&1&3\\2&3&4\\2&4&5\end{bmatrix}$ ②
$\begin{bmatrix}\vec{u}_{31}\\ \vec{u}_{32}\\ \vec{u}_{33}\end{bmatrix} = \begin{bmatrix}1&5&2\\4&4&3\\1&4&4\end{bmatrix}$ ③

Fig. 7. The GSI method greedily selects edges with the order shown on the right side. The final solution is represented by the red subgraph.

For an instance of SRID problem, we could transform it into an instance of the graph model in Fig. 6[2]. Thus, in the later section, we use them interchangeably and use $\{G' = (V', E', W'), \gamma\}$ to denote the graph model instance.

B. Greedy with Short-term Impact

For the graph model, we can easily find a greedy heuristic algorithm. Initially, we set the solution $M = \emptyset$ in the graph model, i.e., $\{x_{ik} = 0|\forall i, k\}$ in the corresponding SRID problem instance. Since our goal is to minimize the infinite norm, we could add one edge that increases the infinite norm most lightly. This means that we will set $x_{ik} = 1$ where $(i, k) = \arg\min_{pq} ||\sum_i \sum_k x_{ik}\vec{u}_{ik} + \vec{u}_{pq}||_\infty$, i.e., adding e_{ik} to M in the graph model. Once we set $x_{ik} = 1$, we delete all edges connecting to node s_i in G' in order to ensure that the degree of the node is one. We could repeat the above steps until all nodes on top have been covered by the subgraph M. Meanwhile, we need to ensure that the number of covered candidate routers cannot be larger than γ. Once the number of covered candidate routers reaches γ, all the edges attaching uncovered candidate routers should be deleted. As we can see, the above algorithm selects the edge which increases θ lightest in the current step. Thus, we call the above method as **G**reedy algorithm with **S**hort-term **I**mpacts (GSI). The details of GSI are shown in Algorithm 1.

Fig. 7 gives a simple example to illustrate GSI. Assume that there are three flows originating from $\{s_1, s_2, s_3\}$ and two candidate SRv6 routers $\{O_1, O_2\}$. O_3 represents that the flows are routed along the shortest paths and $\gamma = 2$. The weights of all edges are set as the right part. At first, $\{\vec{u}_{11}, \vec{u}_{22}, \vec{u}_{32}, \vec{u}_{33}\}$ have the smallest infinite norm. Without loss of generality, we add edge e_{11} into M. Then, we add edge e_{22} into M since $\vec{u}_{11} + \vec{u}_{22}$ has the smallest infinite norm. Finally, we add edge e_{32} into M. As we can see, the GSI method will select the edges $e_{s_1O_1}$, $e_{s_2O_2}$ and $e_{s_3O_2}$ sequentially, then the maximum link utilization will be $||\vec{u}_{11} + \vec{u}_{22} + \vec{u}_{32}||_\infty = \max\{9, 11, 9\} = 11$. However, we can easily find that the optimal solution should cover the edges

[2]For n-segments case, we could extend this graph model to contain n layers where every layer contains the candidate SRv6 routers list. We leave this discussion in Sec. VII.

Algorithm 1 Greedy with Short-term Impact (GSI)

```
Input : $G = (V, E, W, C)$, candidate router set $H$, flows
    $\{f1, f_2, ..., f_m\}$, $\gamma$.
Output : $x_{ik}$, $\omega_i$, $\theta$.
 1: Calculate $\{\vec{u}_{ik}|\forall i, \forall v_k \in H\}$ and construct the SRID graph
    model instance $G'$
 2: $\{x_{ik} = 0|\forall i, k\}$, $\{\omega_k = 0|\forall k\}$.
 3: while True do
 4:     $(i, k) = \arg\min_{p,q} ||(\sum_i \sum_k x_{ik}\vec{u}_{ik} + \vec{u}_{pq})||_\infty$
 5:     $\omega_k = 1$, $x_{ik} = 1$, delete all edges connecting to $s_i$.
 6:     if $\sum_{k=1}^{|H|} \omega_k = \gamma$ then
 7:         for $k = 1$ to $|H|$ do
 8:             Delete all edges connecting to $O_k$ in $G'$ where $\omega_k = 0$
 9:     else if $\sum_i \sum_k x_{ik} = m$ then
10:         break
11: $\theta = ||\sum_i \sum_k x_{ik}\vec{u}_{ik}||_\infty$
```

$\{e_{s_1O_1}, e_{s_2O_1}, e_{s_3O_3}\}$ with the maximum link utilization being $||\vec{u}_{11} + \vec{u}_{21} + \vec{u}_{33}||_\infty = \max\{9, 9, 9\} = 9$. Above example indicates that GSI method sometimes may lose the opportunity to find the global optimal solution. To solve this problem, we redesign the **G**reedy algorithm with considering **L**ong-term **I**mpacts (GLI).

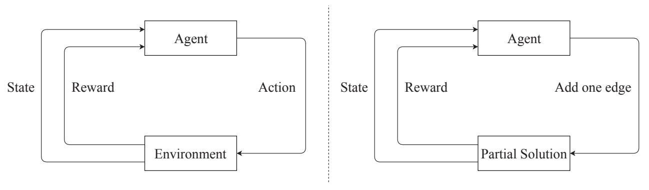


Fig. 8. The framework of reinforcement learning and its application in SRID.

V. Solution II: Greedy with Long-term Impact

Reinforcement learning is a theory that trains agents to find the best solution with considering long-term reward in each decision step [28]. As shown in Fig. 8, the environment is the surroundings of the agent with which the agent can interact through observations, actions, and rewards on actions. Specifically, in each step t, the agent observes state s_t and chooses action a_t, which has the maximum long-term reward. The long-term reward could be acquired by learning. For a given SRID problem instance[3], we can easily transform it into a graph optimization problem and train an agent to solve it sequentially. The right subfigure of Fig. 8 depicts the case when the reinforcement learning framework is applied to the SRID problem. However, such an agent trained for one SRID problem instance cannot be used to solve other SRID problem instances. As a comparison, the GSI method in Section IV can solve any SRID problem instances because GSI has no specific relation with the problem instances.

We hope to train an agent that can be used to solve different SRID problem instances. The agent with good generalization ability will have many benefits since there may be different problem instances in different ISP networks. This expectation equals to the following problem:

[3]A given SRID problem instance means that the network $G = (V, E, W, C)$, the candidate router set H, the value of γ and the flows are given.

Algorithm 2 State Representation with Network Embedding based on Mean-field Inference

```
Input : $G' = (V', E', W')$, hyper parameter $d$, the number of iterations $T$, $X = \{x_{ik}|\forall i, k\}$.
Output : $\{\widetilde{\mu}_{ik}|\forall e_{ik} \in E'\}$
1: Initialize $\{\widetilde{\mu}_{ik} = 0|\forall e_{ik} \in E'\}$, $W_1 \in R^d$, $W_2 \in R^{d\times d}$, $W_3 \in R^d$
2: for $i = 1; i <= T; i++$ do
3:     for $\forall e_{ik} \in E'$ do
4:         $\widetilde{\mu}_{ik} = \sigma(W_1 x_{ik} + W_2 \sum_{e_{uv}\in N(e_{ik})} \widetilde{\mu}_{uv} + W_3 || \sum_{e_{uv}\in N(e_{ik})} \vec{u}_{uv} ||_\infty)$
5: Return $\phi(S)$ using Equation (12)
```

Given the SRID problem P and a series of problem instances $\{P_1, P_2, ..., P_n\}$, can we learn an agent that could solve any unseen problem instance of P?

Fortunately, the results in literature [29] indicated that a general agent could be trained for the minimum vertex cover problem and the maximum cut problem. In this paper, we will leverage the theory in [29] to propose our GLI method. The GLI method includes state representation, action and reward design, state-action value function approximation and Q-learning.

A. Action and State Representation

In the greedy algorithm, we add one edge into the partial solution M greedily based on the graph itself and the current partial solution. Thus, we use the combination of graph and current solution to represent the state S. The action is represented by adding one edge into the partial solution. However, there are $m\times(l+1)$ edges in the graph model; thus, the number of the states could reach $2^{m\times(l+1)}$, which is a huge state space. Besides, the general agent requires that there is a general state representation to cover different problem instances. The above direct representation cannot satisfy the requirement, and it cannot find the potential structure of the graph model. Literature [30] proposed an embedding framework named *structure2vec*. The authors extracted features by performing a sequence of function mappings in a way similar to graphical model inference procedures. In detail, they embedded latent variable models into feature spaces and learned such features spaces using discriminative information. In this paper, we will leverage this theory and design the embedding framework for our SRID graph model. The detailed background theories, e.g., Hilbert space embedding of distributions and mean-field inference, could be found in [30].

The basic idea of embedding our SRID graph model is to add a latent variable $\widetilde{\mu}_{ik} \in R^d$ to each edge e_{ik} in which d is a hyperparameter chosen using cross-validation. Besides, some edges in the SRID graph model are neighbors, which indicates that they share a common node. To model these neighbor relations into the embedding space, we connect the corresponding latent variables together. Fig. 9 shows the graph embedding model of the example in Fig. 7, these latent variables are connected if the corresponding edges are neighbor edges in the original SRID graph model. Since our solution is decided by which edges are selected, thus, we add a 0-1 variable x_{ik} to denote whether the edge e_{ik} is selected.

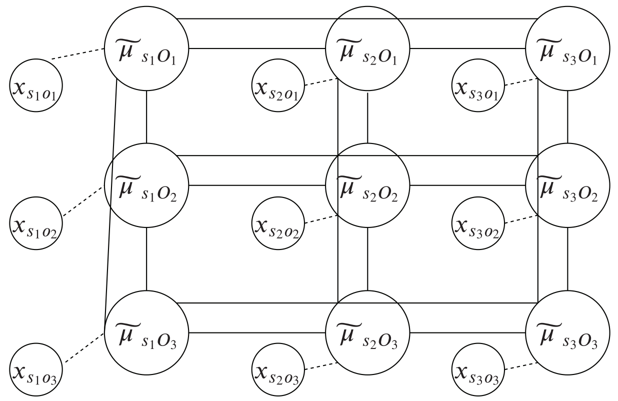

$\widetilde{\mu}_{s_1O_1}$
$\widetilde{\mu}_{s_2O_1}$
$\widetilde{\mu}_{s_3O_1}$
$\widetilde{\mu}_{s_1O_2}$
$\widetilde{\mu}_{s_2O_2}$
$\widetilde{\mu}_{s_3O_2}$
$\widetilde{\mu}_{s_1O_3}$
$\widetilde{\mu}_{s_2O_3}$
$\widetilde{\mu}_{s_3O_3}$
$x_{s_1o_1}$
$x_{s_2o_1}$
$x_{s_3o_1}$
$x_{s_1o_2}$
$x_{s_2o_2}$
$x_{s_3o_2}$
$x_{s_1o_3}$
$x_{s_2o_3}$
$x_{s_3o_3}$

Fig. 9. The graph embedding of the example in Fig. 7. In this figure, each edge is built if the two end nodes represent a pair of neighbor edges in the original SRID graph model.

Obviously, the value of $\widetilde{\mu}_{ik}$ is influenced by the following three parts.

- x_{ik}, which represents whether e_{ik} is included in the solution.
- Other latent variables of its neighbor edges $N(e_{ik})$, which represent the combinatorial structure of the problem.
- $|| \sum_{e_{uv}\in N(e_{ik})} \vec{u}_{uv} ||_\infty$, which are used to distinguish different states further.

Similar to [30], we use a neural network to build this relation:

$$\widetilde{\mu}_{ik} = \sigma(W_1 x_{ik} + W_2 \sum_{e_{uv}\in N(e_{ik})} \widetilde{\mu}_{uv} + W_3 || \sum_{e_{uv}\in N(e_{ik})} \vec{u}_{uv} ||_\infty), \tag{11}$$

where $W_1 \in R^d$, $W_2 \in R^{d\times d}$, $W_3 \in R^d$ and σ represents a ReLU activation function, i.e., $\sigma(x) = max(0, x)$. Note that there is a recursion in Equation (11). The more update iterations we carry out, the farther away the node features will propagate and get aggregated nonlinearly at distant nodes. If we terminate after T iterations, each edge embedding vector $\widetilde{\mu}_{ik}$ will contain information about its $T-$hop neighborhoods.

Finally, we will get a d-dimension embedding vector for each edge $\{\widetilde{\mu}_{ik}|\forall e_{ik} \in E'\}$ in G'. Then the state could be denoted by

$$\phi(S) = \sum_{e_{ik}\in E'} \widetilde{\mu}_{ik} + W_4 \frac{\sum_{e_{ik}\in E'} x_{ik}}{\gamma}, \tag{12}$$

where $W_4 \in R^d$. The above method could embed a SRID graph model instance with a d-dimension vector no matter what size of the original network is. As indicated in literature [29], this general d-dimension vector could represent the state of G' with the partial solution in a unified way for different SRID problem instances. Combining the above network embedding and mean-field inference, we will find the final network embedding as shown in Algorithm 2.

B. Reward

The action a that selects edge e_{ik} could be represented by $x_{ik} = 1$. Given a state S and an action a, we could easily get $(\phi(S_t), a) \rightarrow \phi(S_{t+1})$ using Algorithm 2.

For any partial solution S_t, we could define a function $c(\phi(S_t), G)$ to evaluate its quality. In our SRID graph model,

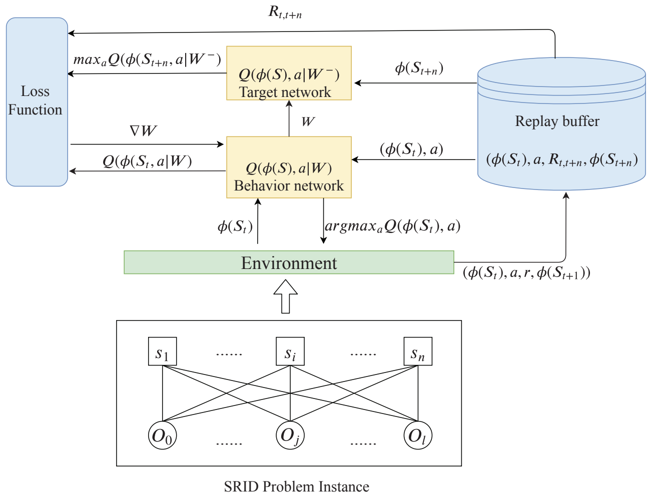

$R_{t,t+n}$
Loss Function
$max_a Q(\phi(S_{t+n}), a|W^-)$
$Q(\phi(S), a|W^-)$ Target network
$\phi(S_{t+n})$
Replay buffer
$(\phi(S_t), a, R_{t,t+n}, \phi(S_{t+n}))$
W
∇W
$Q(\phi(S_t, a|W)$
$Q(\phi(S), a|W)$ Behavior network
$(\phi(S_t), a)$
$\phi(S_t)$
$argmax_a Q(\phi(S_t), a)$
Environment
$(\phi(S_t), a, r, \phi(S_{t+1}))$
s_1 s_i s_n
O_0 O_j O_l
SRID Problem Instance

Fig. 10. The framework of Q-learning.

the goal is to minimize $||\sum_i \sum_{v_k \in H'} \vec{u}_{ik} x_{ik}||_\infty$, it equals to maximize $-||\sum_i \sum_{v_k \in H'} \vec{u}_{ik} x_{ik}||_\infty$. Besides, we should avoid selecting the edges that make the solution infeasible. Thus, we could set $c(\phi(S_t), G)$ as

$$c(\phi(S_t), G) = -||\sum_{e_{ik} \in E' := S} \vec{u}_{ik}||_\infty + h(S_t), \tag{13}$$

where

$$h(S_t) = \begin{cases} 0, & \text{if } X := S_t \text{ not violates model (10)} \\ -\infty, & \text{Otherwise.} \end{cases} \tag{14}$$

Then the reward could be represented by

$$r(\phi(S_t), a) = c(\phi(S_{t+1}), G) - c(\phi(S_t), G). \tag{15}$$

It is easily to prove that the cumulative reward equals to the goal function value of model (10). Let us review the example in Fig. 7 to check the reward design. In the first action, i.e., adding e_{11} into $\emptyset$, the reward is $r_1 = -||\vec{u}_{11}||_\infty - 0 = -||(3,4,2)||_\infty = -4$. In the second action, we add e_{22} into $\{e_{11}\}$ and get a reward $r_2 = -||(3,4,2) + (2,3,4)||_\infty - (-||(3,4,2)||_\infty) = -3$. In the third action, we add e_{32} into $\{e_{11}, e_{22}, e_{32}\}$ and get a reward $r_2 = -||(3,4,2) + (2,3,4) + (4,4,3)||_\infty - (-||(3,4,2) + (2,3,4)||_\infty) = -2$. Finally, we get the cumulative reward $r_1 + r_2 + r_3 = -9$, which equals to the goal function value of the solution in the view of maximization.

C. Q-Learning

In reinforcement learning framework, the state-action function, i.e., $Q(\phi(S), a)$ is used to evaluate the long-term reward of the action. Every time when the agent is in $\phi(S)$, it will take the action $a^* = \max_a Q(\phi(S), a)$. Similar to [29], we use a neural network to approximate the Q function as follows,

$$Q(\phi(S), a_{x_{ik}=1}) = W_5 \sigma([W_6 \phi(S), W_7 \widetilde{u}_{ik}]). \tag{16}$$

In Equation (16), $[\cdot,\cdot]$ is the concatenation operator and $W_5 \in R^{2d}$, $W_6, W_7 \in R^{d \times d}$. Since $\phi(S)$ is decided by $\{W_i\}_{i=1}^3$, $Q(\phi(S), a)$ will be decided by $\{W_i\}_{i=1}^7$. In Fig. 10, we give the framework of learning these parameters.

Algorithm 3 Q-learning for the GLI method

```
Input : A series of the SRID graph model instances $P = \{P_1, P_2, P_3, ...\}$, the experience replay buffer size $|E|$, the number of episodes $L$
Output : $\{W_i\}_{i=1}^5$
 1: Initialize $\{W_i = 0\}_{i=1}^5$ and $\{W_i^- = 0\}_{i=1}^5$
 2: for $i = 1$; $i <= L$; $i++$ do
 3:     Select a problem instance $P_i$ and initialize $\{x_{ik} = 0|\forall i, k\}$
 4:     while Not Terminated do
 5:         Select an edge randomly with probability $\epsilon$, otherwise take action $a_t = argmax_a Q(\phi(S_t), a|W)$
 6:         if $t \geq n$ then
 7:             Add tuple $(\phi(S_{t-n}), a_{t-n}, R_{t-n,t}, \phi(S_t))$ to $E$.
 8:             Sample random batch $B$ from $E$
 9:             Update $\{W_i\}_{i=1}^5$ using stochastic gradient descent over $\sum(y - Q(\phi(S), a)|W)^2$ for $B$.
10:         let $W^- = W$ peridically
11: Return $\{W_i\}_{i=1}^5$.
```

We use an *episode* to represent a complete sequence of edge additions starting from an empty solution in SRID problem and *step* to represent a single action, i.e., adding an edge, in an episode. There are two networks in Q-learning, behavior network and target network, which both of them represent the approximation of Q function but with different parameter values (W and W^-, respectively). The behavior network is responsible for helping the agent to take the next action and generate the episode, while the target network is responsible for providing the forecast value of the Q function. In fact, the setting of these two networks is to increase the stability of the network model [31]. These two networks are same at first, and then the behavior network will update the parameters in each step of the agent and send these updates to the target network periodically.

The standard deep Q-learning algorithm updates the approximation function's parameters at each step of an episode by performing a gradient step to minimize the squared loss

$$(y_t - Q(\phi(S_t), a_t|W))^2 \tag{17}$$

where

$$y_t = \begin{cases} r_{t+1}, & \text{if } S_{t+1} \text{ is terminal state} \\ r(\phi(S_t), a_t) + \gamma \max_a Q(\phi(S_{t+1}), a|W^-), & \text{Otherwise.} \end{cases} \tag{18}$$

As introduced in [29], the final objective value of a solution for a combinatorial optimization problem is only revealed after many edge additions, thus, we use n-step Q-learning [32] to train the parameters. In detail, we compute the forecast value of Q function for non-terminal state with the following equation,

$$y_t = \sum_{i=0}^{n-1} r(\phi(S_{t+i}), a_{t+i}) + \gamma \max_a Q(\phi(S_{t+n}), a). \tag{19}$$

To further improve the convergence speed, one method called fitted Q-iteration has been proposed [29], [33]. In fitted Q-iteration, the parameters of Q-function are updated with a batch of samples instead of sample-by-sample. The batch of samples is randomly selected from a dataset E which is called relay buffer in Fig. 10. In the step $t+n$ of each episode,

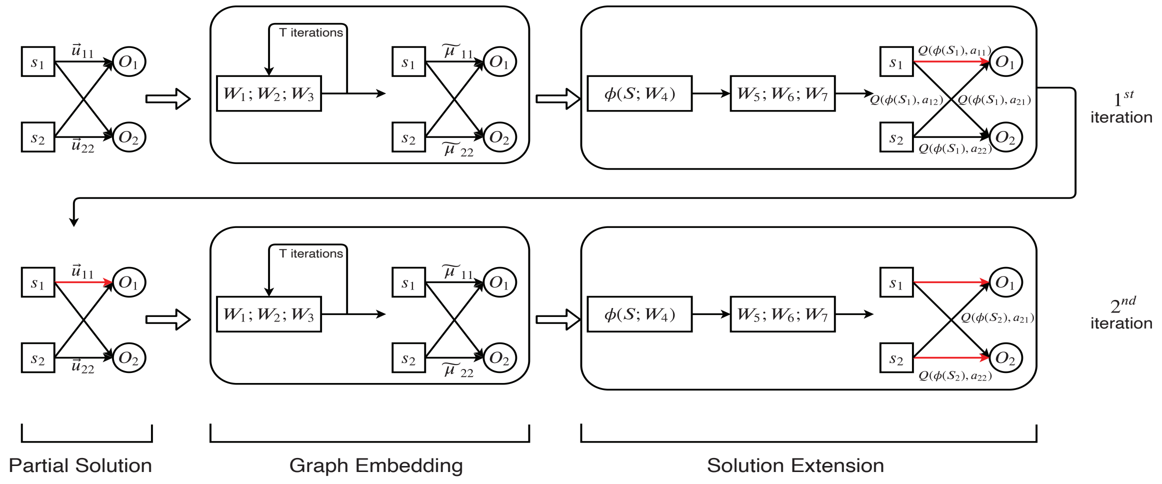


Fig. 11. The overview of **G**reedy method on **L**ong-term **I**mpact (GLI) .

TABLE II
PARAMETER SETTING.

Notation	Description
Network Topology ID	{synth50, synth100, synth200, rf1755, rf3257, rf3967, rf6461}
# flows	$\{10, 20, 30, ..., 200\}$
# candidate routers	$\{10, 15, 20, ..., 40\}$
γ	$\{1, 2, ..., 10\}$
the dimension of latent variables	$d = 64$
the batch size	$\lvert B\rvert = 64$
the propagation iterations	$\lvert T\rvert = 5$
the delay steps in n-step learning	$n = 5$

the tuple $(\phi(S_t), a_t, R_{t,t+n}, \phi(S_{t+n}))$ is added to E, with $R_{t,t+n} = \sum_{i=0}^{n-1} r(S_{t+i}, a_{t+i})$. With these samples and the loss function, the stochastic gradient descent algorithm updates the parameters in the behavior network. The behavior network will synchronize itself to the target network periodically. To describe the parameters training more clearly, we summarize the details in Algorithm 3.

Once we get the value of $\{W_i\}_{i=1}^5$, we could realize the greedy algorithm with considering the long-term impact. Fig. 11 gives an example of GLI. After finishing the training, we will get values of all parameters. Initially, we use Algorithm 2 to embed the graph, then use parameters $\{W_i\}_{i=4}^7$ to get the Q values of all available actions. Next, we select the action with the maximum Q value and extend the corresponding partial solution. In the example, $e_{s_1O_1}$ is added into the partial solution. Repeat the above steps until the terminate condition is triggered, and then we will get the final solution. It is emphasized that the action space is influenced by the state. In the second iteration of the example, the available actions are adding $e_{s_2O_1}$ or $e_{s_2O_2}$ since there is no need to considering edges connecting s_1. The biggest difference between GLI and GSI is that GLI greedily selects the edge based on Q value, while GSI is based on estimating the degree of increase in the goal function.

VI. EVALUATION

A. Experiment Setting

1) Dataset and parameter setting: We use the network topologies and flow demands provided by DEFO [22], [34].

- Training dataset: We generate SRID problem instances from DEFO dataset. There are 7 network topologies in DEFO, for each network topology, we randomly select $\{10, 20, 30, ..., 200\}$ flow demands, $\{10, 15, 20, ..., 40\}$ candidate routers and set γ from $\{1, 2, ..., 10\}$. For each parameter combination, we construct 2 problem instances, thus, we have total $7 \times 20 \times 7 \times 10 \times 2 = 19600$ problem instances. Like [29], we set the dimension of latent variables $d = 64$, the batch size $|B| = 64$, the propagation iterations in state representation $|T| = 5$ and the delay steps in Q-learning $n = 5$. All the parameters in the training phase are summarized in Table II.
- Validation dataset: We select China Education and Research Network (CERNET), rf1221 and rf1239 as the network topology in validation experiments. There are 37, 151 and 972 edges in CRENT, rf1221 and rf1239, respectively. Thus, the weights in the three corresponding graph models will be 1×37, 1×151 and 1×972 vectors, respectively.

2) Compared Methods: Since the method in literature [12] focuses on SR-MPLS, which has a different data plane mechanism from our SRID, we did not select it as our benchmark. We use Gurobi, SP, DEG and BTW as the benchmarks. The Gurobi method uses an academic free programming solver, i.e., gurobi [35], to find the optimal solution of the model (10) directly. We restrict the solver time of Gurobi within one hour to ensure that the problem is solved efficiently. The SP method steers all flows along the shortest paths, which are the default paths in OSPF. DEG and BEW are proposed in [24]. In DEG (degree), the nodes are selected in the descending order of node degree. In BTW (betweenness centrality), the nodes are selected in the descending order of betweenness centrality, which means the number of shortest paths passing through the node. It is needed to emphasize that the DEG and BTW methods are only capable of deciding which nodes could be upgraded but not finding the routing. Thus, after DEG and BTW deciding the upgraded nodes, we use our GSI method to assign the routings.

3) Implementation: The GLI algorithm is implemented using Pytorch based on the codes in graph_com_bopt [36]. And we trained GLI on Amazon EC2 p2.xlarge instance, which is CUDA K80-enabled. Then we evaluate all the algorithms on MacBook Pro with Intel Core i5.

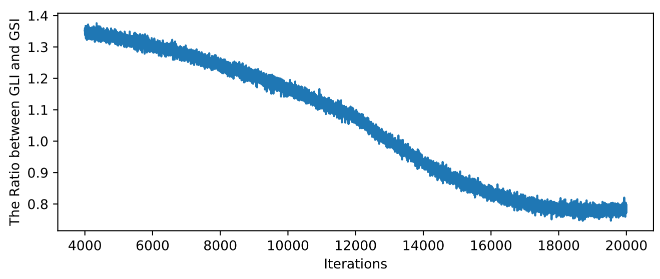


Fig. 12. Convergence of GLI training process which is measured by the ratio between the solutions of GLI and GSI.

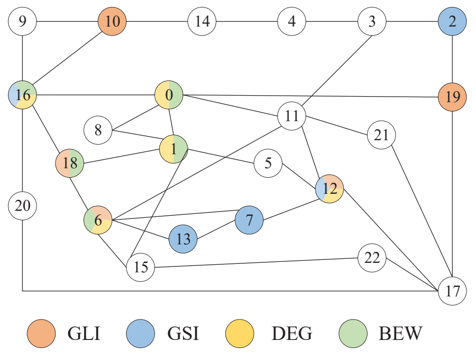
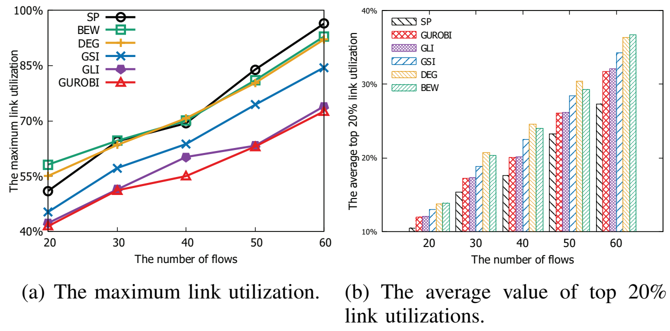


Fig. 13. The different SRv6 deployment solutions in CERENT with the setting that $|H| = 15$, $\gamma = 5$.

4) Convergence of GLI: We use GSI to get the benchmark solutions to the different SRID problem instances in the training phase. In Fig. 12, we plot GLI's convergence which is measured by the ratios between the solutions of GLI and GSI. As we can see, GLI performs better than GSI after around 12500 iterations and finally converges to approximately 0.8.

B. Visualize the different solutions in CERENT

We first evaluate the different methods with CERENT topology since we can visualize the different solutions in this small-scale case easily. We generate a traffic matrix with 400 flows, set the candidate router set with $|H| = 15$ and $\gamma = 5$. It is noted that we set the above parameters without special requirements. In fact, we can set other values for these parameters without influencing the conclusions. As shown in Fig. 13, the compared four methods have different deployment solutions. The DEG method deploys the SRv6 routers in the nodes which have the highest degrees, and the BEW method deploys them in the nodes that have the highest between centrality values. However, we can see that the solution of GLI method presents different characteristics that both node 10 and node 19 do not have high degrees or high between centrality values. Instead, the GLI method deploys the SRv6 routers more evenly. We think that this balanced deployment can avoid the bottleneck links around the nodes with high between centrality values. In this experiment, we find that the solution of the GLI method is the same as the optimal solution resulted from Gurobi. The maximal link utilizations of GSI, DEG and BEW are higher than the optimal value by 47.2%, 65.4% and 65.6%, respectively.



(a) The maximum link utilization. (b) The average value of top 20% link utilizations.

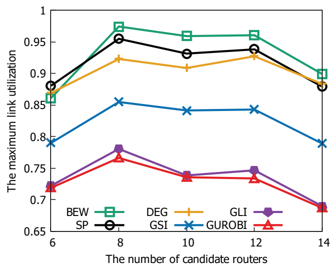
Fig. 14. Small-scale: The impacts of the number of flows on two metrics.

C. Experiment results in small-scale case

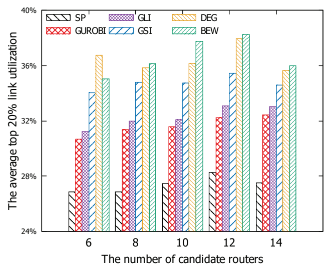
As shown in model (10), its complexity is decided by the number of flows, the number of candidate routers and the number of network edges. To evaluate the impacts of the above parameters with the optimal solution, we restrict the values of the above variables. In this experiment group, we select rf1221 as our network topology. To better display the experiment results, we scale the capacities of the network so that the maximum link utilization in the most congested case is nearly 100%.

1) The impacts of the number of flows: In this experiment, we set $\gamma = 5$ and select $|H| = 20$ candidate routers randomly. Fig. 14(a) indicates that the value of the maximum link utilization grows up as the number of flows increases. This phenomenon is easy to understand since more flows will consume more bandwidth. The optimal solution can reduce the maximum link utilization by 21.93% on averagely comparing to default shortest paths. The average result of GLI method is only 2.83% higher than the optimal solution, while the average result of GSI method is 14.27% higher than the optimal solution. The above results indicate that GLI method certainly outperforms GSI method. Besides, we can see that the curves of DEG and BEW are almost the same. This phenomenon may be caused by two factors. The first factor is that the nodes with higher degrees usually have high betweenness centrality values. The example in Fig. 13 shows that the DEG method and the BEW method select as high as 80% of common nodes. The second factor is that the number of flows in this experiment is low such that the selected nodes may not need to handle the flows. Fig. 14(b) shows that SP method can produce the lightest value in terms of the top 20% highest congested links. The reason behind this tendency is that the SRv6 strategy will steer more flows to traverse the links connecting to SRv6 routers. This result indicates that we should also enlarge the link bandwidth near the SRv6 routers.

2) The impacts of the candidate routers: In this experiment, the number of flows is 60 and $\gamma = 5$. Since the SP method will neglect all candidate routers, its curve should be flat. However, as shown in Fig. 15(a), the curve representing SP method has a variation. We think that this variation is produced by selecting flow demands randomly. The other curves show a similar variation, which indicates that the number of candidate routers has no impact on reducing the maximum link utilization as long as it is larger than the value of γ. In Fig. 15(b), the variations of the methods could reach 5.74% at most, which proves the above conclusion further. One strange point of

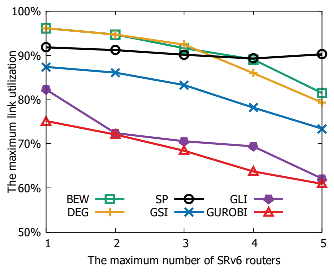


(a) The maximum link utilization.

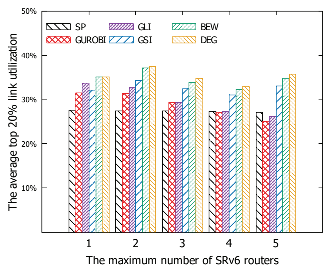


(b) The average value of top 20% link utilizations.

Fig. 15. Small-scale: The impacts of the number of candidate routers on two metrics.

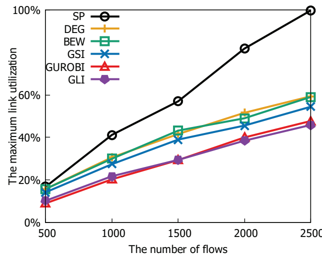


(a) The maximum link utilization.

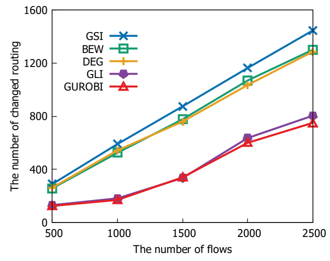


(b) The average value of top 20% link utilizationss.

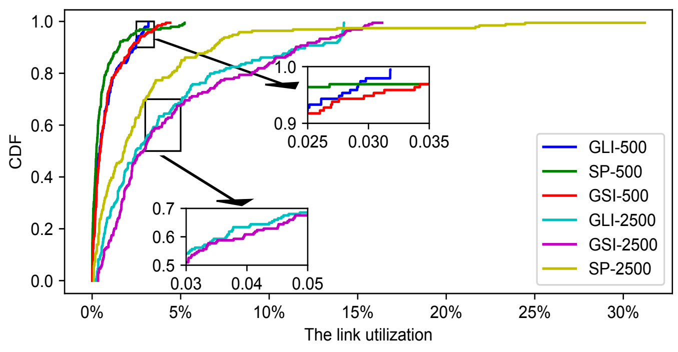
Fig. 16. Small-scale: The impacts of γ on two metrics.



(a) The maximum link utilization.



(b) The number of flows that have to change their default routings.



(c) The cumulative distribution function of top 20% link utilizations.

Fig. 17. Large-scale: The impacts of the number of flows on three metrics.

Fig. 15(a) is that the curve of the BEW method lies above the curve of SP. The reason behind this phenomenon is that the BEW method only selects the nodes with the highest betweenness centrality values while the detail routings are greedily found by GSI.

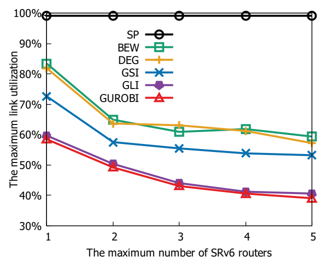
3) The impacts of γ: In this experiment, the number of flows is 60 and the number of candidate routers is 10. As shown in Fig. 16(a), the curve representing SP method is nearly flat, while the other five curves decline as the number of candidate routers increases. This is because that SP method neglects the candidate routers while the other five methods will adjust their routing solutions with different γ. Particularly, the DEG method and the BEW method can relieve the congestion of SP as γ increases. However, DEG and BEW still perform worse than GSI due to the inflexibility in selecting the SRv6 routers. Compared to SP, the maximum link utilizations of GUROBI, GLI and GSI could be reduced by 24.84%, 21.2% and 11.26%, respectively. Fig. 16(b) shows the same tendency, i.e., the average top 20% link utilization of GUROBI and GLI decreases as γ increases. Besides, the bars of GSI method do not decline as the γ increases. This may be because that GSI overvalues short-term impact and is trapped in the local optimal solution zone.

D. Experiment results in large-scale case

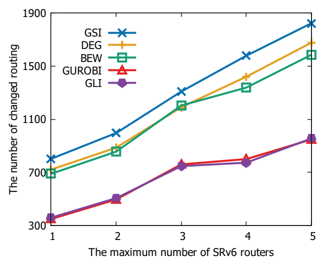
In this group of experiments, we use rf1239 as our network topology and evaluate the impacts of the number of flows and γ since the results in small-scale showed that the number of candidate routers almost have no impacts. Instead of presenting the average value of the top 20% link utilizations in the small-scale experiments, we present the cumulative distribution function of the top 20% link utilizations. Besides, we compare the number of flows that have to change their routings with two different greedy methods. Like the experiments in Sec. VI-C. We also scale the capacities of the network to make the maximum link utilization in the most congested case be nearly 100%.

1) The impacts of the number of flows: In this experiment, we set the number of candidate routers as 25 and $\gamma = 5$. Like Fig. 14(a), Fig.17(a) also shows that the maximum link utilization grows up as the number of flows increases. The GLI method can reduce the maximum link utilization by 48.3% and 20.8% averagely against SP and GSI, respectively. Fig. 17(b) shows that the number of flows that have to change their default routing increases as the number of flows grows up. Considering the results in Fig. 14(a) and Fig.17(a) together, we can see that GLI method changes 55.1% fewer routings averagely than GSI method but can reduce the maximum link utilization by 20.8% on average. This result indicates that the greedy strategy which considers long-term impact can reduce the maximum link utilization with less overhead. As a comparison, DEG and BEW change fewer routing paths and produce higher congestion. Fig. 17(c) presents the cumulative distribution function of the top 20% link utilizations. To make the figure readable, we only show the case with 500 flows and 2500 flows. It is easy to find that the curves with 2500 flows lie on the right of curves with 500 flows. Meanwhile, the curves of GLI-500 and GLI-500 lie on the left of GSI-500's and GSI-500's, respectively. This phenomenon indicates that GLI method could reduce both the average link utilization and the maximum link utilization.

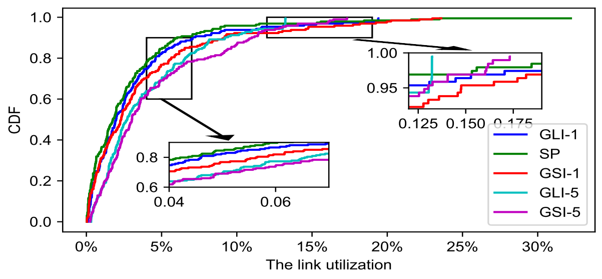
2) The impacts of γ: In this experiment, we set the number of candidate routers as 25 and the number of flows as 2500. Fig. 18(a) shows that GLI method can reduce the maximum link utilization by 59.1% and 23.7% at most against SP and GSI, respectively. Also, we can see that the maximum link utilization of the two greedy methods reduces as the γ



(a) The maximum link utilization.



(b) The number of flows that have to change their default routings.



(c) The cumulative distribution function of top 20% link utilizations .

Fig. 18. Large-scale: The impacts of the γ on three metrics.

increases, which results from that larger γ gives much more space to change the default routing paths. This assumption can be proved by Fig. 18(b) where the number of flows that change default routings increases as γ grows up. Similar to Fig. 17(b), GLI method is more efficient than GSI method since the number of changed routings of GLI is average 49.2% and is less than that of GSI method. In Fig. 18(c), we plot the cumulative distribution function curves of the top 20% link utilizations. To make it clear, we only present the curves with $\gamma = 1$ and $\gamma = 5$. Under the same method, the curve with larger γ lies on the right of the curve with lower γ(e.g., see the curves of GLI-1 and GLI-5). The reason behind this phenomenon is that larger γ will change more flow routings (as shown in Fig 18(b)) and lead to the utilizations of the links near to the SRv6 routers increase. Under the same γ, the curve of GLI method lies on the left of the curve of GSI method (e.g., see the curves of GLI-5 and GSI-5). This further indicates that GLI method outperforms the GSI method since it could minimize both the maximum link utilization and the average link utilization.

VII. Discussion

A. Model Generality

In the previous section, we have pointed out that our method could be extended to the general case where n-segments routing is permitted. In 2-segments case, there are $m\times|E|\times(l+1)$ different cases for constraint (2) where m represents the number of flows, $|E|$ represents the number of edges in G and l represents the number of candidate routers. While in n-segments case, there will be $m \times |E| \times \sum_{k=0}^{n} A_l^k$ different cases since any permutation of the candidate routers from H could be chained together. Fortunately, the graph model of SRID problem could convey these massive constraints in a simple way. For n-segments routing case, we could construct a

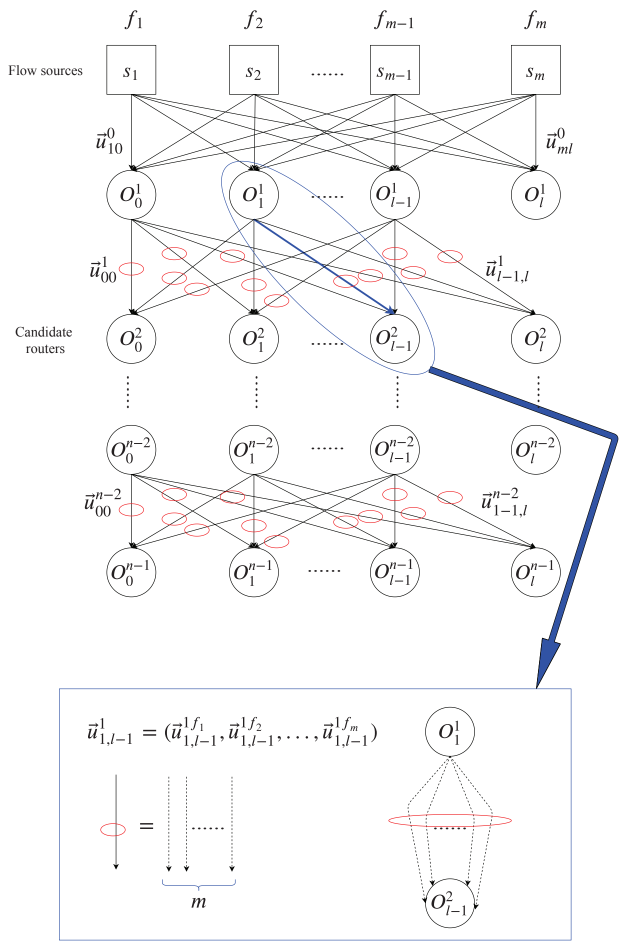


Fig. 19. The multi-level multigraph model for SRID problem.

multi-level multigraph for the SRID problem as Fig. 19 shown. The 0^{th} level represents the source nodes of m flows. There are $l+1$ nodes in the following each level where the left l nodes represent the candidate routers in H ($|H| = l$) and the rightest node represents the virtual router. From the 1^{st} level, each arrow towards the next level contains m parallel edges. Each arrow has a composite weight which is composed of m vectors. For the example in the bottom of Fig. 19, we can find $\vec{u}_{1,l-1}^1 = (\vec{u}_{1,l-1}^{1f_1}, \vec{u}_{1,l-1}^{1f_2}, ..., \vec{u}_{1,l-1}^{1f_m})$ and $\vec{u}_{1,l-1}^{1f_1}$ represents the link utilization of all edges in the original graph G when f_1 traverses the shortest path between O_1 and O_{l-1} in G. It is noted that for the arrows connecting nodes in the last row and the last column, the vectors are calculated differently. For example, in the last row, $\vec{u}_{j,k}^{n-2}(j < l, k < l)$, $\vec{u}_{j,k}^{n-2f_i}$ represents the link utilization of all edges when f_i is routed along the shortest path between O_j and O_k and then the shortest path between O_k and the flow destination d_i. In the last column, $\vec{u}_{j,l}^{h}(h \leq n-2, j < l)$, $\vec{u}_{j,l}^{hf_i}(h \leq n-2, j < l)$ represents the link utilization of all edges when f_i is routed along the shortest path between O_j and the flow destination d_i since O_l represents the virtual router.

To find the routing for a flow f_i, we need to find a path in this multi-level multigraph that satisfies the following conditions:

- Condition 1: The path starts from s_i and ends in any node of the last level or the rightest column, i.e., $\{O_j^{n-1}|j =$

$0, 1, 2, ..., l\} \cup \{O_l^j | j = 1, 2, ..., n-1\}$.

- Condition 2: All the edges apart from the first one must have the weights about f_i, i.e., annotated by $\vec{u}_{j,k}^{hf_i}$.

With this multi-level multigraph model, we could solve the SRID problem by finding a subgraph satisfying the following constraints:

- The degree of each source node on top is one.
- There exists a path from the source to the node in the last level or the node in the rightest column.
- Without considering the source nodes, there are at most γ columns are covered in the subgraph.
- For each source node, there exists a path satisfying condition 1 and condition 2.
- The sum vector of all the weights in the subgraph should be minimized in the term of its infinite norm.

With this multi-level multigraph model, we could extend the GSI method and the GLI method to solve the SRID problem. In the GSI method, we could first select the edge between the 0^{th} level and 1^{st} level greedily to ensure which flows should be first routed. Then we extend the path for the selected flows with the same greedy rule, i.e., increase the objection value lightest. In the GLI method, we need an agent to evaluate the long-term impact of adding one edge into the subgraph. In the extension procedures of GSI and GLI, we just compare the edges related to the current flow in each level. Such a design could ensure that the produced result is a feasible solution to the SRID problem. To reduce the impact of the maximal number of available segments, we should train an agent for each possible value of n. In this way, we could promise that each agent is trained without embedding the parameter of n.

B. Multi-stage Deployment

One important assumption of the SRID problem is that the network infrastructure cannot be fully upgraded at once and usually are upgraded with several stages. Our SRID model can be used to solve the incremental deployment problem even though it includes several stages. In fact, we have modeled this multi-stage problem into the SRID model with Equation (6). In this equation, we assign a binary variable h_k to denote whether node v_k has already been upgraded in previous stages. Then we could treat a multi-stage deployment problem as multiple independent single-stage deployment problems, which can be solved by the GSI method and GLI method. The only cost of this transformation is that we need to recalculate the weights of edges that are related to the upgraded SRv6 nodes.

VIII. Conclusion

The network infrastructure should be upgraded incrementally; thus, we studied the problem of SRv6 incremental deployment (SRID) for ISPs in this paper. Firstly, we formally defined and formulated the SRID problem with integer programming. Then we transform this problem into a graph model. To solve the problem efficiently, we designed two methods, i.e., GSI and GLI, by greedily extending the partial solution. In detail, GSI focused on enabling those SRv6 routers that increase the goal function lightest in the current step. In comparison, GLI devoted to enabling SRv6 routers with considering the long-term reward. To realize GLI, we designed an end-to-end reinforcement learning framework including network embedding, reward design and Q-learning. The experimental results over a public dataset showed that both GSI and GLI could significantly reduce the maximum link utilization, with GLI cutting down the maximum link utilization by 59.1% at most against the default shortest path routing method.

Acknowledgment

This work is partially supported by the National Key Research and Development Program of China (No. 2018YFE0207600 and No. 2020YFE0200500), the National Natural Science Foundation of China (No. U19B2024 and No. 61802421), European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie Grant (No. 824019), and Tianjin Science and Technology Foundation (No.18ZXJMTG00290).

References

[1] M. Ikram, N. Vallina-Rodriguez, S. Seneviratne, M. A. Kâafar, and V. Paxson, "An analysis of the privacy and security risks of android VPN permission-enabled apps," in *Proc. of ACM IMC*, 2016.
[2] "Three Different Approaches to Managing Your Bank's WAN," Available:https://www.safesystems.com/blog/2016/02/three-different-approaches-to-managing-your-banks-wan/.
[3] "Leased Line Costs vs Down Time Costs," Available:https://leased-line-comparison.co.uk/leased-line-costs-vs-time-costs/.
[4] K. Muthukrishnan and A. G. Malis, "A core MPLS IP VPN architecture," *RFC*, vol. 2917, pp. 1–16, 2000.
[5] Y. Rekhter, R. P. Bonica, and E. C. Rosen, "Use of provider edge to provider edge (PE-PE) generic routing encapsulation (GRE) or IP in BGP/MPLS IP virtual private networks," *RFC*, vol. 4797, pp. 1–10, 2007.
[6] V. Bollapragada, M. Khalid, and S. Wainner, *IPSec VPN Design*. Cisco Press, 2005.
[7] W. George and C. Pignataro, "Gap analysis for operating ipv6-only MPLS networks," *RFC*, vol. 7439, pp. 1–28, 2015.
[8] W. T. Penzhorn and J. Amsenga, "Ipv6, ipsec, and vpns," in *The Industrial Information Technology Handbook*, R. Zurawski, Ed. CRC Press, 2005, pp. 1–18.
[9] Z. N. Abdullah, I. Ahmad, and I. Hussain, "Segment routing in software defined networks: A survey," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 1, pp. 464–486, 2019.
[10] "SRv6-BE," Available:https://support.huawei.com/enterprise/en/doc/EDOC1100087998/bf5b67bc/srv6-be.
[11] K. Poularakis, G. Iosifidis, G. Smaragdakis, and L. Tassiulas, "One step at a time: Optimizing SDN upgrades in ISP networks," in *In Proc. of INFOCOM*, 2017.
[12] A. Cianfrani, M. Listanti, and M. Polverini, "Incremental deployment of segment routing into an ISP network: a traffic engineering perspective," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 3146–3160, 2017.
[13] Y. Tian, Z. Wang, X. Yin, X. Shi, Y. Guo, H. Geng, and J. Yang, "Traffic engineering in partially deployed segment routing over ipv6 network with deep reinforcement learning," *IEEE/ACM Trans. Netw.*, vol. 28, no. 4, pp. 1573–1586, 2020.
[14] Srv6 network programming. [Online]. Available: https://tools.ietf.org/html/draft-filsfils-spring-srv6-network-programming-07
[15] R. Bhatia, F. Hao, M. S. Kodialam, and T. V. Lakshman, "Optimized network traffic engineering using segment routing," in *In Proc. of IEEE INFOCOM*, 2015, pp. 657–665.
[16] T. Schuller, N. Aschenbruck, M. Chimani, M. Horneffer, and S. Schnitter, "Traffic engineering using segment routing and considering requirements of a carrier IP network," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1851–1864, 2018.
[17] M. K. Mukerjee, D. Han, S. Seshan, and P. Steenkiste, "Understanding tradeoffs in incremental deployment of new network architectures," in *In Proc. of ACM CoNEXT*, 2013.

[18] S. Gay, R. Hartert, and S. Vissicchio, “Expect the unexpected: Sub-second optimization for segment routing,” in *In Proc. of IEEE INFOCOM*, 2017, pp. 1–9.

[19] M. Jadin, F. Aubry, P. Schaus, and O. Bonaventure, “CG4SR: near optimal traffic engineering for segment routing with column generation,” in *In Proc. of IEEE INFOCOM*, 2019, pp. 1333–1341.

[20] R. Carpa, O. Glück, and L. Lefèvre, “Segment routing based traffic engineering for energy efficient backbone networks,” in *In Proc. of IEEE ANTS*, 2014, pp. 1–6.

[21] M. Lee and J. Sheu, “An efficient routing algorithm based on segment routing in software-defined networking,” *Computer Networks*, vol. 103, pp. 44–55, 2016.

[22] R. Hartert, S. Vissicchio, P. Schaus, O. Bonaventure, C. Filsfils, T. Telkamp, and P. François, “A declarative and expressive approach to control forwarding paths in carrier-grade networks,” in *In Proc. of ACM SIGCOMM*, 2015, pp. 15–28.

[23] P. L. Ventre, M. M. Tajiki, S. Salsano, and C. Filsfils, “SDN architecture and southbound apis for ipv6 segment routing enabled wide area networks,” *IEEE Trans. Network and Service Management*, vol. 15, no. 4, pp. 1378–1392, 2018.

[24] Y. Tian, Z. Wang, X. Yin, X. Shi, Y. Guo, H. Geng, and J. Yang, “Traffic engineering in partially deployed segment routing over ipv6 network with deep reinforcement learning,” *IEEE/ACM Trans. Netw.*, vol. 28, no. 4, pp. 1573–1586, 2020.

[25] J. L. Sobrinho and M. A. Ferreira, “Routing on multiple optimality criteria,” in *In Proc. of ACM SIGCOMM*, 2020.

[26] P. L. Ventre, S. Salsano, M. Polverini, A. Cianfrani, A. Abdelsalam, C. Filsfils, P. Camarillo, and F. Clad, “Segment routing: a comprehensive survey of research activities, standardization efforts and implementation results,” *CoRR*, 2019.

[27] “Subset Sum is NP-complete,” Available:http://www.cs.cornell.edu/courses/cs4820/2018fa/lectures/subset_sum.pdf.

[28] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[29] E. B. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, “Learning combinatorial optimization algorithms over graphs,” in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, 2017, pp. 6348–6358.

[30] H. Dai, B. Dai, and L. Song, “Discriminative embeddings of latent variable models for structured data,” in *In Proc. of ICML*, M. Balcan and K. Q. Weinberger, Eds., 2016.

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nat.*, vol. 518, no. 7540, pp. 529–533, 2015.

[32] R. S. Sutton and A. G. Barto, “Reinforcement learning: An introduction,” 2011.

[33] M. A. Riedmiller, “Neural fitted Q iteration - first experiences with a data efficient neural reinforcement learning method,” in *In Proc. of ECML*, 2005.

[34] “Declarative and Expressive Forwarding Optimizer,” Available:https://sites.uclouvain.be/defo/#code.

[35] “GUROBI,” Available:https://www.gurobi.com/.

[36] “Graphcombopt,” Available:https://github.com/Hanjun-Dai/graph_comb_opt.

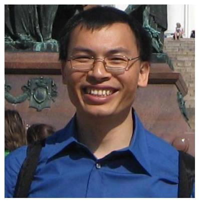
Bangbang Ren received the B.S.degree and M.S. degree in management science and engineering from National University of Defense Technology in 2015 and 2017, respectively. He is currently working toward Ph.D in the National University of Defense Technology. His research interests include software-defined network, network function virtualization, approximation algorithm.

Deke Guo received the B.S. degree in industry engineering from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2001, and the Ph.D. degree in management science and engineering from the National University of Defense Technology, Changsha, China, in 2008. He is currently a Professor with the College of Information System and Management, National University of Defense Technology, and a Professor with the School of Computer Science and Technology, Tianjin University. His research interests include distributed systems, software-defined networking, data center networking, wireless and mobile systems, and interconnection networks. He is a senior member of the IEEE and a member of the ACM.

Yali Yuan received the M.Sc. degree from University of Lanzhou, Lanzhou, China, in 2015 and the Ph.D. degree in University of Göttingen, Göttingen, Germany in 2018 where she is currently working as a Postdoctoral Fellow. Her research interests include intelligent internet of things, network intrusion and attack detection, privacy protection, and network localization and security.

Guoming Tang (S’12-M’17) is currently a research fellow at the Peng Cheng Laboratory, Shenzhen, Guangdong, China. He received his Ph.D. degree in Computer Science from the University of Victoria, Canada, in 2017, and the Bachelor’s and Master’s degrees from the National University of Defense Technology, China, in 2010 and 2012, respectively. He was also a visiting research scholar of the University of Waterloo, Canada, in 2016. His research mainly focuses on computational sustainability, edge/cloud computing and intelligent transportation systems.

Weijun Wang received the B.S. degree in the Department of Computer and Software from Nanjing University of Post and Telecommunication, Nanjing, China, in 2014 and M.E. degree in the Department of Communication Engineering from PLA University of Science and Technology, Nanjing, China, in 2017. He is studying towards the Ph.D degree in the Department of Computer Science and Technology in Nanjing University. His research interests focus on UAV monitoring problem, and video analytics system. He is a student member of IEEE.



Xiaoming Fu (M’02-SM’09) received the Ph.D. degree in computer science from Tsinghua University, China, in 2000. He was then a research staff with the Technical University Berlin until joining the University of Gottingen, Germany, in 2002, where he has been a professor in computer science and heading the Computer Networks Group since 2007. He has also held visiting positions at ETSI, University of Cambridge, Columbia University, Tsinghua University, and UCLA. He is a Distinguished Lecturer of IEEE, a member of ACM, a fellow of IET, and a member of Academia Europaea. His research interests include network architectures, protocols, and applications.