On Orchestration of Segment Routing and In-band Network Telemetry

Bofan Chen, Feng Chen, Shaofei Tang, Qitao Zheng, and Zuqing Zhu, *Fellow, IEEE*

***Abstract*—With the rapid development of programmable data plane (PDP), both segment routing (SR) and in-band network telemetry (INT) have attracted intensive interests. Hence, we have previously proposed the technique of SR-INT, which explores the benefits of SR and INT simultaneously and gets rid of the hassle of the accumulated overheads of them. In this work, we further expand the advantage of SR-INT by studying how to plan the SR-INT schemes of flows at the network level to balance the tradeoff between bandwidth usage and coverage of network monitoring, namely, the problem of "SR-INT orchestration". A mixed integer linear programming model (MILP) is first formulated for the problem, and we prove its NP-hardness. Then, to reduce the time complexity of problem-solving, we propose a novel greedy algorithm based on path ranking and a column generation (CG) based approximation algorithm. Extensive simulations verify the performance of our proposed algorithms.**

***Index Terms*—Segment routing (SR), In-band network telemetry (INT), Programmable data plane (PDP), Network monitoring, Column generation, Approximation algorithm.**

I. Introduction

With the momentum gained from fast-emerging innovations, the Internet has been continuously reshaped over past decades. This makes it a better place for bandwidth-/data-intensive applications with stringent quality-of-service (QoS) demands. Recent developments on data-centers [1, 2] and 5G networks [3] have stimulated advances on network architecture [4–6], physical-layer technologies [7–10], virtualization approaches [11–13], *etc*. The flexibility brought by these advances has made the Internet more prone to faults and thus complicated network control and management (NC&M) [14]. To address the unprecedented challenges faced by today's NC&M systems, we need more powerful and adaptive network monitoring techniques that can visualize network operations in realtime and detect/locate exceptions timely and accurately.

According to [15], conventional network monitoring methods can be categorized as active measurements (*e.g.*, Ping and Traceroute [16]), passive measurements (*e.g.*, sFlow [17]), and hybrid measurements (*e.g.*, reactive measurement [18]). However, as the data collection of these methods are normally not in realtime, they have difficulty in satisfying the requirements of today's NC&M. This restriction can be overcome by leveraging the programmable data plane (PDP) [5, 6], which enables network operators to define new packet fields and customize packet processing pipelines, for facilitating novel network functions for network monitoring and troubleshooting. For instance, in-band network telemetry (INT) [19] can be realized with PDP for monitoring networks and locating exceptions in a realtime, fine-grained, and flow-oriented way.

B. Chen, F. Chen, S. Tang, Q. Zheng, and Z. Zhu are with the School of Information Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China (email: zqzhu@ieee.org).

Specifically, INT lets packets in a service flow carry telemetry instructions and data. When a packet enters the network, the ingress PDP switch inserts an INT header in it to represent the telemetry instruction for network monitoring. Then, as the packet is forwarded along its routing path, each intermediate PDP switch checks the telemetry instruction, collects the required network status of when the packet gets processed in the PDP switch, and encodes the status data as specific INT fields in the INT header of the packet. Finally, before exiting the network, the egress PDP switch extract the INT header from the packet and send it to a data analyzer. In this way, the data analyzer can analyze the telemetry data in the INT header to reply how the packet got processed in the network.

Despite its promising advantages, INT also has a few drawbacks. For instance, repeated insertion of INT fields can make a packet excessively long and bring in noticeable bandwidth overhead. Moreover, as PDP switches need to invoke "*AddField*" frequently, which is a relatively costly action for packet processing, INT can prolong packet processing latency and affect the QoS of network services [20]. These drawbacks make it difficult to use INT together with other emerging techniques that also count on manipulating packet header fields, while the benefits of INT on network monitoring and troubleshooting cannot be fully exploited by itself. Here, one example is segment routing (SR) [21], which is famous for being capable of realizing adaptive routing efficiently.

SR inserts a series of SR labels in the header of each packet at its ingress PDP switch to indicate the routing path, and thus each subsequent PDP switch can just forward the packet according to the right SR label (SRL). As both INT and SR are realized with PDP switches, they can benefit each other mutually. Specifically, INT provides rich telemetry data for the control plane to conduct traffic engineering and failure recovery with SR, while the control plane also leverages SR to adjust the data collection scheme of INT for adaptive network monitoring. Nevertheless, as INT and SR both insert a stack of header fields in each packet, they can be incompatible because of the maximum transmission unit (MTU).

Therefore, it is relevant to study how to maintain the accumulated overhead of INT and SR such that they can be used simultaneously. Note that, with SR, the routing path of a packet is encoded as a stack of SRLs, each of which denotes a path segment, and the last switch of each segment needs to update the current SRL to point to the next segment [22]. Hence, if we replace an SRL with a bundle of INT fields at the last switch of each segment, we can time-multiplex the space in each packet header for INT and SR, and maintain the length of each packet as unchanged along its routing path. This novel scheme, namely, SR-INT [23], enables operators to explore the mutual benefits of INT and SR, without worrying about the violation of MTU due to their accumulated overhead.

Compared to the conventional INT, SR-INT has the restriction that telemetry data can only be collected at the last switch of each path segment. However, we hope to point out that this restriction is actually not an issue. This is because per-switch monitoring with INT is normally not necessary in most of networks [24, 25], especially for those in which exceptions do not happen frequently and widely. Meanwhile, per-switching monitoring with INT can bring in intolerable overheads when collecting and processing telemetry data. Therefore, the impact of SR-INT's restriction can be minimized by properly balancing the tradeoff between coverage and overheads of SR-INT, *i.e.*, orchestrating the SR-INT schemes on flows to efficiently cover "critical" switches on account of resource constraints. Specifically, we only need to reasonably plan the paths of the flows for SR-INT in the network, divide the paths into suitable segments such that all or most of the "critical" switches become last switches of the segments. Meanwhile, the events in the switches that are not critical ones will be missed to reduce the overheads of SR-INT. Note that, SR-INT denotes paths with unique SRLs and instructs the switches on each path to forward packets accordingly, which is different from finding paths with the interior gateway protocol (IGP).

This work studies the aforementioned SR-INT orchestration problem, *i.e.*, how to optimize the SR-INT schemes on flows at the network level such that under resource constraints, the coverage of INT-based network monitoring can be maximized. We first formulate a mixed-integer linear programming model (MILP) to solve the problem exactly, and prove that it is $\mathcal{NP}$-hard. Next, in order to make the problem-solving much more time-efficient, we first design a novel greedy-based heuristic based on path ranking, and then propose an approximation algorithm based on column generation (CG). Finally, we conduct extensive simulations to verify the effectiveness of our proposals. The main contributions of our work are:

- To the best of our knowledge, this is the first work on orchestrating SR and INT for maximizing the coverage of INT-based monitoring under resource constraints.
- We formulate an MILP model to solve the problem exactly and prove its $\mathcal{NP}$-hardness.
- We design a CG-based approximation algorithm that can get a near-optimal solution within a reasonable number of iterations, and theoretically analyze the algorithm's convergence, relative error, and time complexity.

The rest of the paper is organized as follows. Section II surveys the related work briefly. We describe the operation principle of SR-INT and explain the network model for SR-INT orchestration in Section III. In Section IV, the MILP model and the greedy-based heuristic are designed, and we also prove the problem's $\mathcal{NP}$-hardness there. Section V explains our proposal of the CG-based approximation algorithm and show the theoretical analysis on it. We evaluate the performance of our proposals with extensive simulations in Section VI. Finally, Section VII summarizes the paper.

II. Relate Work

SR was evolved from multi-protocol label switching (MPLS) [26] and has been standardized in SRv6 [22]. Specifically, SRv6 adds a segment routing header (SRH) in the extension header of IPv6 to store a series of SRLs for specifying an explicit routing path. Since its inception, many studies have considered how to optimize the SR schemes of flows for traffic engineering (TE) [27–32]. Bhatia *et al.* [27] respectively proposed algorithms to solve the offline and online versions of SR-based TE. The studies in [28, 29] formulated integer linear programming (ILP) and MILP models, respectively, to address SR-based TE, and they also proposed time-efficient heuristics. The evaluation of SR-based TE with real-world topologies and traffic demands was conducted in [30]. The authors of [31] proposed to plan SR-based TE with a CG-based algorithm for improved time-efficiency. For the same purpose, Pereira *et al.* [32] proposed an evolutionary computation based approach.

The first specification of INT was released in [19], which suggested to collect telemetry data on a per-switch and per-packet basis. Based on this scheme, people have considered how to plan the flows with INT to effectively monitor a network, *i.e.*, the so-called INT orchestration (INTO) problem [33]. Specifically, the investigations on INTO can be categorized into two approaches: 1) planning INT-based probe flows for effective network monitoring [34–38] and 2) optimizing INT-based data collection with active service flows [33, 39].

Note that, even though using probe flows for INT-based network monitoring provides us more flexibility to solve the INTO problem, it also introduces two drawbacks. First, as probe flows do not experience exactly the same network state as service flows, the network monitoring based on them might not be accurate. Second, probe flows generate extra overheads on bandwidth usage and packet processing, and thus they might affect the performance of service flows. Therefore, we will not pursue the first approach in this work. The second approach only relies on active service flows for INT-based data collection, but it does not address the excessive overheads of per-switch and per-packet INT. Recently, based on the idea of sampling switches and packets for INT-based data collection, people have proposed a few selective INT scenarios [24, 25, 40, 41] to better balance the tradeoff between accuracy and overheads of INT-based network monitoring. However, none of these studies have addressed how to optimize the selective INT schemes on service flows at the network level.

In addition to lack of work on selective INT orchestration, existing studies did not explore the mutual benefits of INT and SR either. Although certain studies did consider to use INT together with SR (*e.g.*, the NetView in [36]), they still assumed separate packet header spaces for the stacks of SRLs and INT fields and did not address the accumulated overheads of INT and SR. To the best of our knowledge, our proposed SR-INT in [23] is the only existing scheme that can explore the mutual benefits of INT and SR and reduce their accumulated overheads simultaneously. Specifically, in [23], we designed the operation principle of SR-INT and realized a system prototype based on PDP to demonstrate its effectiveness. Nevertheless, the orchestration of SR-INT schemes on service flows at the network level has not been considered in [23].

III. SR-INT Orchestration

In this section, we first explain the operation principle of SR-INT [23], and then introduce the problem of SR-INT orchestration at the network level.

A. Operation Principle

Fig. 1 illustrates the operation principle of SR-INT. We make each SRL use the same length as that of an INT field (*e.g.*, 4 bytes [19]), and the last switch of each path segment replaces the first SRL in the stack with an INT field that stores the telemetry data regarding itself[1]. Hence, the size of each SR-INT packet stays as unchanged when being forwarded along its routing path, and thus the accumulated overheads of SR and INT can be effectively reduced.

Note that, as SR-INT lets PDP switches collect and insert telemetry data into packets instantly [23], it does not sacrifice any accuracy of telemetry data collection. Our system prototype of SR-INT in [23] could collect the telemetry data about *Device ID*, *Output Port*, *Hop Latency*, and *Bandwidth*. Here, *Device ID* tells the ID of a switch, *Output Port* stores the output port that a flow used on the switch, *Hop Latency* records the processing time of a packet in the switch, and *Bandwidth* tells the bandwidth usage on the output port used by a packet. Therefore, SR-INT can be used to detect and locate network exceptions such as congestion, switch misconfiguration, *etc*.

To assign one SR-INT scheme to a service flow, the SDN controller first calculates the flow's routing path according to its source, destination and QoS requirement, then divides the routing path into several segments according to the requirements of network monitoring (*i.e.*, each "critical" switch on the path should terminate a segment such that telemetry data regarding it can be collected with SR-INT), and installs the corresponding entries of flow tables (*i.e.*, flow entries) in related switches. Hence, when a packet of the flow enters the network, the ingress switch encodes an ordered list of SRLs in it, each of which denotes a segment on its routing path. Next, the packet is forwarded along each segment according to the first SRL, and when it reaches the last switch of a segment, the first SRL is popped out, which makes the next SRL the first one, and an INT field is inserted. This procedure is repeated until the packet arrives at its egress switch.

The egress switch duplicates the packet, forwards a copy to one data analyzer (DA), which will extract, parse and index the telemetry data in INT fields, and removes the INT fields from the other copy before sending it to the destination host. Moreover, SR-INT can also make flows share SRLs [42], and thus the memory usage in switches for flow entries is saved. In this work, we assume that the PDP switches are the hardware-based ones, which can handle the operations of INT and SR at line-rates (*i.e.*, without sacrificing any packet processing throughput) [43, 44]. Then, the impact of SR-INT on the PDP switches is only the memory usage of related flow entries.

B. Problem Description

When there are multiple service flows in the network, how to plan the SR-INT schemes on them for effective network monitoring becomes a sophisticated optimization problem. Fig. 1 uses three flows to explain the problem of SR-INT orchestration. The flows are *Flow* 1: *Switch* 1$\rightarrow$*Switch* 4, *Flow* 2: *Switch* 1$\rightarrow$*Switch* 7, and *Flow* 3: *Switch* 5$\rightarrow$*Switch* 7. As shown in Fig. 1, we assume that the critical switches are *Switches* 3 and 6. Hence, the flows are routed through them for INT-based network monitoring.

As for *Flow* 1, the SDN controller divides its routing path into two segments, and thus after its ingress switch (*Switch* 1), each of its packets contains a stack of two SRLs (*i.e.*, the *SR* 1 and *SR* 3 in Fig. 1), which correspond to *Segments* 1$\rightarrow$2$\rightarrow$3 and 3$\rightarrow$4, respectively. At *Switch* 2, the packet is forwarded to the output port that goes to *Switch* 3, and as *Switch* 2 is not the last switch of a segment, the first SRL on the packet gets preserved. Next, at *Switch* 3, the SRL for *Segment* 1$\rightarrow$2$\rightarrow$3 gets replaced with an INT field that contains telemetry data about *Switch* 3. Finally, *Switch* 4 replaces the only SRL on the packet with an INT field about itself and duplicates the packet. One copy is forwarded to the DA for collecting the telemetry data on it, and another copy is sent to the destination of *Flow* 1 after *Switch* 4 removing all the INT fields.

As *Flows* 1 and 2 share *Segment* 1$\rightarrow$2$\rightarrow$3, *Switch* 1 encodes the same first SRL (*i.e.*, *SR* 1) on their packets. Hence, the two flows share one flow entry on *Switch* 2, saving some memory usage there, while their flow entries on *Switches* 1 and 3 are different. The path of *Flow* 2 consists of *Segments* 1$\rightarrow$2$\rightarrow$3, 3$\rightarrow$6 and 6$\rightarrow$7, for collecting telemetry data at *Switches* 3, 6 and 7. *Flow* 3 collects telemetry data of *Switches* 6 and 7, and *Flows* 2 and 3 share *Segment* 6$\rightarrow$7 (*i.e.*, *SR* 4).

Hence, to leverage SR-INT for efficient network monitoring, one needs to optimize the SR-INT schemes on flows at the network level. This means that under resource constraints (*i.e.*, the bandwidth capacity of links and the memory space on switches for flow entries), we need to optimize 1) the routing paths of flows[2], 2) how to partition the paths into segments, and 3) the INT-based data collection schemes, such that the coverage of INT-based network monitoring is maximized. We refer to this problem as "SR-INT orchestration". Note that, in this work, we only solve the problem of SR-INT orchestration for planning the SR-INT on flows once. However, in a dynamic network environment where the traffic condition can change, we might also need to re-plan the SR-INT schemes on flows from time to time, which will be studied in our future work.

IV. Optimization Model

In this section, we formulate the MILP for SR-INT orchestration, analyze the problem's complexity, and design a greedy-based heuristic for time-efficient problem-solving.

[1]Note that, using only one INT field to store the telemetry data regarding a switch will not restrict the effectiveness of SR-INT. This is because if the switch needs to collect multiple types of telemetry data that cannot be accommodated in an INT field, we can encode the telemetry data in multiple packets and rely on data analyzers at network edge for data aggregation [40].

[2]We assume that the network operator can plan the routing paths of all the flows involved in SR-INT orchestration freely. If a flow has restriction on path calculation, the operator will exclude it from SR-INT orchestration.

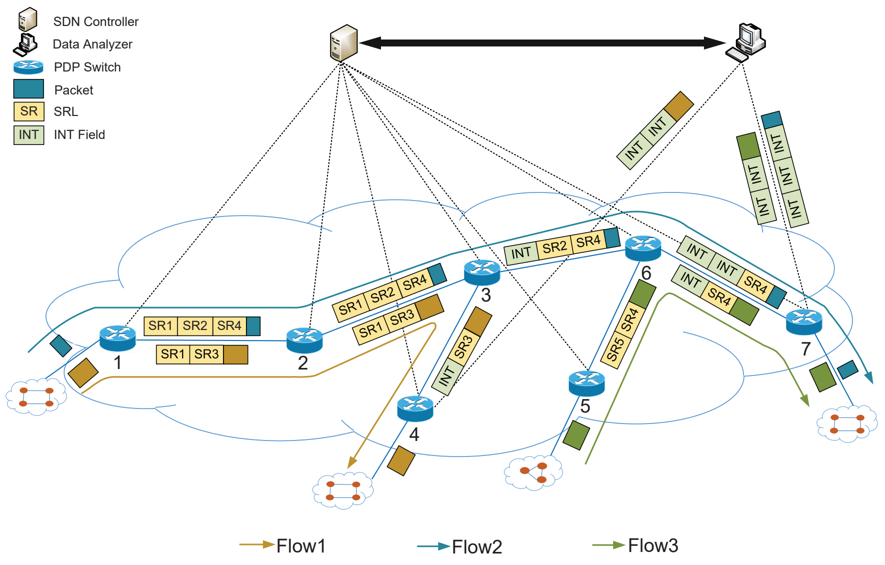


Fig. 1. Operation principle of SR-INT.

A. MILP Model

Parameters:

- $G(V, E)$: the topology of the PDP network, where V and E are the sets of switch nodes and links, respectively.
- $B_{u,v}$: the bandwidth capacity of link $(u, v) \in E$.
- F: the set of flows that are considered for SR-INT.
- s_k/d_k: the source/destination node of the k-th flow in F.
- b_k: the bandwidth demand[3] of the k-th flow in F.
- n_v^s: the number of flows that have node v as sources.
- n_v^d: the number of flows that have node v as destinations.
- γ_v: the memory capacity of the PDP switch on node v in terms of flow entries.
- W: the total bandwidth usage if all the flows in F are assumed to use their shortest paths.
- d_v: the degree of node v.
- m: the maximum degree of nodes in $G(V, E)$.
- a_v: the total bandwidth capacity of links on node v.

Variables:

- x_v: the boolean variable that equals 1 if the routing paths of multiple flows separate at node v, and 0 otherwise.
- g_v: the boolean variable that equals 1 if the routing paths of multiple flows converge at node v, and 0 otherwise.
- $f_{u,v}$: the boolean variable that equals 1 if link (u, v) carries at least one flow in F, and 0 otherwise.
- $c_{u,v}^k$: the boolean variable that equals 1 if link (u, v) carries the k-th flow in F, and 0 otherwise.
- ω_v: the real variable that denotes the importance of node v for being monitored.
- e_v: the integer variable that indicates the number of flow entries installed on the switch on node v.
- y_v: the real variable that denotes the contribution of node v in network monitoring, *i.e.*, y_v equals ω_v if node v is selected for INT-based data collection, and 0 otherwise.

Objective:

[3]The bandwidth demand here already includes the overhead of SR-INT.

The optimization objective is to minimize the total bandwidth usage of flows in F and maximize the total contribution of the nodes that are monitored with SR-INT[4]

$$\text{Minimize} \quad \frac{\alpha}{W} \cdot \sum_{k=1}^{|F|} \sum_{(u,v)\in E} c_{u,v}^k \cdot b_k - \beta \cdot \sum_{v\in V} y_v, \tag{1}$$

where α and β are the weights of the two terms in the objective. By adjusting the proportional relation between α and β, we can change their importance in the optimization. Specifically, in practice, α and β should be carefully set by the operator according to 1) what kinds of and how many of the events of interest should be detected and correctly diagnosed, and 2) how many SR-INT overheads can be tolerated. The impact of the values of α and β on the performance of SR-INT orchestration will be discussed in Section VI.

Constraints:

$$\sum_{v:(u,v)\in E} c_{u,v}^k - \sum_{v:(v,u)\in E} c_{v,u}^k = \begin{cases} 1, & u = s_k, \\ -1, & u = d_k, \\ 0, & \text{otherwise}, \end{cases} \quad \forall k \in [1, |F|]. \tag{2}$$

Eq. (2) ensures that the routing path of each flow satisfies the flow conservation condition.

$$\sum_{k=1}^{|F|} c_{u,v}^k \cdot b_k \le B_{u,v}, \quad \forall (u, v) \in E. \tag{3}$$

Eq. (3) ensures that the bandwidth consumed by the flows on each link does not exceed the link's capacity.

$$\frac{1}{|F|} \cdot \sum_{k=1}^{|F|} c_{u,v}^k \le f_{u,v} \le \sum_{k=1}^{|F|} c_{u,v}^k, \quad \forall (u, v) \in E. \tag{4}$$

Eq. (4) ensures that the value of each variable $f_{u,v}$ is set correctly, that is, when there are flow(s) passing through link (u, v), $f_{u,v}$ equals 1, and 0 otherwise.

[4]Note that, by not invoking INT data collection on all the nodes in a network but focusing on those whose importance is relatively high, our MILP model actually implicitly reduces the overhead/load of INT data processing on DAs.

$$\begin{cases} x_v \leq \dfrac{|F|-1+n_v^d-n_v^s+\sum\limits_{u:(v,u)\in E} f_{v,u}-\sum\limits_{u:(u,v)\in E} f_{u,v}}{|F|}, \\ x_v \geq \dfrac{n_v^d-n_v^s+\sum\limits_{u:(v,u)\in E} f_{v,u}-\sum\limits_{u:(u,v)\in E} f_{u,v}}{|F|}, \end{cases} \quad \forall v \in V, \tag{5}$$

$$\begin{cases} g_v \leq \dfrac{|F|-1+n_v^s-n_v^d+\sum\limits_{u:(u,v)\in E} f_{u,v}-\sum\limits_{u:(v,u)\in E} f_{v,u}}{|F|}, \\ g_v \geq \dfrac{n_v^s-n_v^d+\sum\limits_{u:(u,v)\in E} f_{u,v}-\sum\limits_{u:(v,u)\in E} f_{v,u}}{|F|}, \end{cases} \quad \forall v \in V. \tag{6}$$

Eqs. (5) and (6) ensure that the values of x_v and g_v are set correctly, to make sure that the segments on the routing path of each flow are defined correctly and the last switch on each segment is selected by SR-INT for telemetry data collection.

$$\begin{cases} e_v \leq \gamma_v, \\ e_v \leq n_v^d + (1-g_v-x_v)\cdot|F| + \sum\limits_{u:(v,u)\in E}\sum\limits_{k=1}^{|F|} c_{v,u}^k, \\ e_v \geq n_v^d - (1-g_v-x_v)\cdot|F| + \sum\limits_{u:(v,u)\in E}\sum\limits_{k=1}^{|F|} c_{v,u}^k, \quad \forall v \in V. \\ e_v \leq n_v^d + (g_v+x_v)\cdot|F| + \sum\limits_{u:(v,u)\in E} f_{v,u}, \\ e_v \geq n_v^d - (g_v+x_v)\cdot|F| + \sum\limits_{u:(v,u)\in E} f_{v,u}, \end{cases} \tag{7}$$

Eq. (7) ensures that the number of flow entries installed on each switch does not exceed its memory capacity. According to the design in [23], the number of flow entries consumed by SR-INT can be estimated as follows. First of all, if a flow uses a node as its source/destination in the PDP network, one flow entry is installed there for it. Then, if a node is the first/last one on a path segment, each in/out flow that uses the segment consumes a flow entry there, respectively. Finally, if a node is an intermediate one on a segment, all the flows that uses the segment share a flow entry there.

$$\omega_v = \left(\frac{d_v + n_v^d}{m}\right) + \left\lceil \frac{\sum\limits_{v:(v,u)\in E}\sum\limits_{k=1}^{|F|} c_{v,u}^k\cdot(a_v + m\cdot b_k)}{m\cdot a_v} \right\rceil, \ \forall v \in V. \tag{8}$$

Eq. (8) ensures that the importance of each node for being monitored by SR-INT is set correctly. In this work, we assume that the importance of a node depends on the total bandwidth usage of and the number of the flows passing through it.

$$\begin{cases} y_v \geq 0, \\ y_v \leq 2 + \dfrac{|F|}{m}\cdot(x_v+g_v), \\ y_v \leq \omega_v, \\ y_v \geq \omega_v - \left[2 + \dfrac{|F|}{m}\cdot(1-x_v-g_v)\right], \end{cases} \quad \forall v \in V. \tag{9}$$

Eq. (9) ensures that the contribution of each node to the network monitoring with SR-INT is determined correctly.

By solving the MILP, we can get the optimal solution of the SR-INT orchestration problem. Specifically, the values of variables $\{c_{u,v}^k\}$ determines the routing paths of flows, the values of variables $\{x_v\}$, and $\{g_v\}$ tell us how to partition the paths into segments, and the values of variables $\{y_v\}$ define the INT-based data collection schemes. For example, if we assume that the path of a flow is 1→2→3→4→5 and the MILP obtains $x_1 = x_2 = 1$ for *Nodes* 1 and 2 and $x_3 = x_4 = x_5 = 0$ for *Nodes* 3, 4 and 5, respectively, and $g_4 = 1$ and $g_1 = g_2 = g_3 = g_5 = 0$. Then, the MILP's solution partitions the flow's path into three segments as 1→2, 2→3→4 and 4→5.

B. Complexity Analysis

Theorem 1: SR-INT orchestration is an $\mathcal{NP}$-hard problem.

Proof: We prove the $\mathcal{NP}$-hardness of SR-INT orchestration by restricting it to the generation case of a well-known $\mathcal{NP}$-hard problem [45]. We apply the restriction of $\beta = 0$, which means that the optimization objective of the SR-INT orchestration becomes to minimize the total bandwidth usage of flows in F only. This makes the optimization be equivalent to planning the flows's paths in a resource-constrained network such that the total bandwidth usage is minimized, which is just the general case of the multi-commodity flow problem (MCF) [46]. As MCF is known to be $\mathcal{NP}$-hard [46], we can prove the $\mathcal{NP}$-hardness of SR-INT orchestration. ■

C. Greedy-based Heuristic with Path Ranking (G-PR)

Since SR-INT orchestration is an $\mathcal{NP}$-hard problem, we first resort to designing a polynomial-time heuristic to solve it quickly. *Algorithm* 1 shows the detailed procedure, which is a greedy-based heuristic with path ranking (G-PR). *Lines* 1-5 are for the initialization. Here, for each flow in F, we calculate K shortest paths for it in $G(V, E)$ and will determine the SR-INT scheme of the flow based on these paths (*Lines* 2-5). Moreover, in *Line* 4, we assign a weight to each flow as

$$\eta_k = \frac{1}{K}\cdot\sum_{p\in P_k}\left[b_k\cdot hop(p)\right], \tag{10}$$

where $hop(\cdot)$ returns the hop-count of a path. The rationale behind Eq. (10) is to assign a larger weight to a flow that has a larger bandwidth demand and a longer average path length. Then, *Line* 6 sorts the flows in F in descending order of their weights to ensure that the flow with a larger weight will be handled earlier for saving bandwidth resources.

Next, the for-loop of *Lines* 7-19 determines the SR-INT scheme of each flow greedily. *Lines* 8-9 initialize the variables for each iteration. In *Line* 10, we check all the pre-calculated paths for flow f_k to select those that can satisfy the bandwidth constraints in Eq. (3) and the switch memory constraints in Eq. (7) and put the paths in set P_k'. The for-loop covering *Lines* 11-16 finds the path for flow f_k, which leads to the smallest objective, and store it in p. *Lines* 17-18 route flow f_k over p, and update network status and optimization objective ψ. The time complexity of *Algorithm* 1 is $O(K\cdot|F|\cdot(|V|^3+|F|))$. Specifically, we first use the Yen's algorithm to calculate K shortest paths for each flow in F, which has the complexity of $O(K\cdot|F|\cdot|V|^3)$, then the complexity of sorting flows is

$O(K \cdot |F|^2)$, and finally, the complexity of finding all the feasible paths for a flow is $O(K \cdot |F| \cdot |V|)$.

Algorithm 1: Greedy-based Heuristic (G-PR)

```
Input  : $G(V,E)$, $\{B_{u,v}\}$, $\{\gamma_v\}$, and $F$.
Output: objective $\psi$, and path set of flows $P$.
1  $\psi = 0$, $P = \emptyset$;
2  for $k \in [1, |F|]$ do
3      calculate $K$ shortest paths for the $k$-th flow in $F$;
4      store the paths in set $P_k$ and get weight $\eta_k$ of the
       flow with Eq. (10);
5  end
6  sort flows in $F$ in descending order of their weights;
7  for $k \in [1, |F|]$ do
8      $P'_k = \emptyset$, denote the $k$-th flow as $f_k$;
9      $\psi_k = +\infty$, $p = \emptyset$;
10     find all the paths in $P_k$ that satisfy resource
       constraints for carrying $f_k$ and store them in $P'_k$;
11     for each path $p' \in P'_k$ do
12         route $f_k$ over path $p'$ hypothetically to obtain
           the new objective $\psi'_k$ with Eq. (1);
13         if $\psi'_k < \psi_k$ then
14             $\psi_k = \psi'_k$, $p = p'$;
15         end
16     end
17     $\psi = \psi_k$;
18     insert $p$ in $P$ for $f_k$ and update network status;
19 end
20 return($\psi$, $P$);
```

V. CG-based Approximation Algorithm

Both the MILP and G-PR designed in the previous section have drawbacks, because the MILP will become intractable for large-scale problems while G-PR cannot guarantee any performance gap to the optimal solution. Therefore, in this section, we leverage CG [47] to develop a time-efficient approximation algorithm based on the MILP in Section IV-A.

A. Overall Procedure of CG-based Algorithm

In SR-INT orchestration, the essential part is to plan the routing path of each flow. Hence, if we denote the a feasible path of a flow as one column c and get the column set C_k for each flow $f_k \in F$, we can leverage CG to optimize the path selection with columns in iterations to obtain a near-optimal solution for SR-INT orchestration. We first decompose the MILP in Section IV-A into a master problem and a pricing problem. Then, we relax the integer variables in the master problem to real ones and obtain a restricted master problem (RMP). Since the optimal solution of RMP might not be that of the original MILP, we use the pricing problem to determine whether the objective of RMP can be reduced by selecting columns from the master problem to add into RMP. If yes, we add the selected columns in RMP and update the pricing problem accordingly. These steps are repeated until we cannot further reduce the objective of RMP. Then, the optimal solution of RMP becomes the near-optimal solution of the original MILP, which can ensure a bounded approximation ratio [47].

Algorithm 2 shows the overall procedure of the CG-based approximation algorithm. *Line* 1 defines all the parameters for denoting a column c, which represents a feasible routing path of one flow. We then decompose the original MILP of SR-INT orchestration into a master problem and a pricing problem and formulate an MILP (MILP-MP) and an ILP (ILP-PP) to represent them, respectively (*Lines* 2-3). *Line* 4 leverages *Algorithm* 1 to solve the SR-INT orchestration for an initial solution (*i.e.*, a feasible routing path for each flow in F). Then, we initialize the column set C_k for each flow f_k with *Lines* 5-9. Specifically, for f_k, we generate a column c based on its routing path in the initial solution P_0 (*Line* 7), and insert c into its column set C_k (*Line* 8). Next, *Line* 10 constructs RMP, which is the linear programming (LP) relaxation of MILP-MP, with the column sets of all the flows in F.

After constructing RMP, we solve the problem of SR-INT orchestration with the while-loop that covers *Lines* 11-23. *Line* 12 solves RMP to obtain the values of primal and dual variables, which can be done in polynomial-time [48]. Then, the for-loop of *Lines* 13-18 generates a new column for each flow in F. Specifically, *Line* 14 updates the ILP-PP of flow f_k based on the solution of RMP, *Line* 15 solves the ILP-PP to get its objective ϕ_k, *Line* 16 generates a new column c based the ILP-PP's solution, and *Line* 17 inserts c into the column set of f_k (*i.e.*, C_k). *Line* 19 updates RMP with the newly-generated columns. Next, if the minimum objective of all the ILP-PPs is non-negative, the CG cannot get a better solution with more iterations (*Lines* 20-22). Finally, the near-optimal solution of the original MILP is obtained by building an MILP-MP with the most updated column set and solve it (*Lines* 23-24).

B. CG Model

Our CG model uses the following parameters to denote a column c, which represents a feasible routing path of one flow.

- $\rho^{u,v}_{k,c}$: the boolean that equals 1, if the path in column c suggests that flow f_k uses link (u, v), and 0 otherwise.

1) Master Problem: In the following, we formulate the MILP model of the master problem (MILP-MP) based on the solution space defined by the existing column sets ($\{C_k, \ \forall k \in [1, |F|]\}$), to optimize the routing paths of all the flows in F.

Variables:

- $\lambda_{k,c}$: the boolean variable that equals 1 if f_k uses the routing path represented by $c \in C_k$, and 0 otherwise.
- $\tilde{y_v}$: the non-negative real variable that denotes the contribution of node v in network monitoring.
- $x_v, g_v, f_{u,v}, e_v$: the variables whose definitions are the same as those in Section IV.

Objective:

The objective is similar as that of the original MILP, but we only consider the existing columns ($\{C_k, \ \forall k \in [1, |F|]\}$).

$$\text{Minimize } \frac{\alpha}{W} \cdot \sum_{k=1}^{|F|} \sum_{c \in C_k} \lambda_{k,c} \cdot \left(\sum_{(u,v) \in E} \rho^{u,v}_{k,c} \cdot b_k \right) - \beta \cdot \sum_{v \in V} \tilde{y_v}. \quad (11)$$

Constraints:

Algorithm 2: CG-based Approximation Algorithm

```
1  define the parameters to denote a column $c$;
2  formulate MILP-MP for the master problem;
3  formulate ILP-PP for the pricing problem;
4  get an initial solution $P_0$ with *Algorithm* 1;
5  for $k \in [1, |F|]$ do
6      $C_k = \emptyset$;
7      generate a column $c$ for flow $f_k$ based on $P_0$;
8      insert $c$ into $C_k$;
9  end
10 build the LP relaxation of MILP-MP with
   $\{C_k,\ \forall k \in [1, |F|]\}$ to obtain RMP;
11 while TRUE do
12     solve RMP to get values of primal/dual variables;
13     for $k \in [1, |F|]$ do
14         update the ILP-PP of flow $f_k$ according to the
           solution of RMP;
15         solve the ILP-PP to get its objective $\phi_k$;
16         generate a new column $c$ based on the
           solution of the ILP-PP;
17         insert $c$ into $C_k$;
18     end
19     update RMP with $\{C_k,\ \forall k \in [1, |F|]\}$;
20     if $\min_k(\phi_k) \geq 0$ then
21         break;
22     end
23 end
24 use $\{C_k,\ \forall k \in [1, |F|]\}$ to build an MILP-MP;
25 solve the MILP-MP to obtain a near-optimal solution
   to the original MILP;
```

For the constraints are related to variables $\{\lambda_{k,c}\}$ and $\{\tilde{y_v}\}$, we define their dual variables in "()". These dual variables provide the reduction on the objective in Eq. (11). For those constraints where the relation of "$\leq$" exists, we add a minus sign before each of their dual variables. Hence, none of the dual variables will be negative.

$$\sum_{c\in C_k} \lambda_{k,c} = 1, \quad \forall k \in [1, |F|], \quad (\epsilon_k). \tag{12}$$

Eq. (12) ensures that each flow only uses the routing path defined in one column.

$$\sum_{k=1}^{|F|}\sum_{c\in C_k} \lambda_{k,c}\cdot\rho_{k,c}^{u,v}\cdot b_k \leq B_{u,v}, \quad \forall (u,v)\in E, \quad (-\xi_{u,v}). \tag{13}$$

Eq. (13) ensures that the bandwidth used by the flows on a link does not exceed the link's bandwidth capacity.

$$\begin{cases} \displaystyle\sum_{k=1}^{|F|}\sum_{c\in C_k} \lambda_{k,c}\cdot\rho_{k,c}^{u,v} \geq f_{u,v}, \quad (\mu_{u,v}), \\ \displaystyle\frac{1}{|F|}\cdot\sum_{k=1}^{|F|}\sum_{c\in C_k} \lambda_{k,c}\cdot\rho_{k,c}^{u,v} \leq f_{u,v}, \quad (-\nu_{u,v}), \end{cases} \quad \forall (u,v)\in E. \tag{14}$$

Eq. (14) ensures that each variable $f_{u,v}$ has a correct value.

$$\begin{cases} x_v \leq \dfrac{|F| - 1 + n_v^d - n_v^s + \sum\limits_{u:(v,u)\in E} f_{v,u} - \sum\limits_{u:(u,v)\in E} f_{u,v}}{|F|}, \\ x_v \geq \dfrac{n_v^d - n_v^s + \sum\limits_{u:(v,u)\in E} f_{v,u} - \sum\limits_{u:(u,v)\in E} f_{u,v}}{|F|}, \end{cases} \quad \forall v \in V. \tag{15}$$

$$\begin{cases} g_v \leq \dfrac{|F| - 1 + n_v^s - n_v^d + \sum\limits_{u:(u,v)\in E} f_{u,v} - \sum\limits_{u:(v,u)\in E} f_{v,u}}{|F|}, \\ g_v \geq \dfrac{n_v^s - n_v^d + \sum\limits_{u:(u,v)\in E} f_{u,v} - \sum\limits_{u:(v,u)\in E} f_{v,u}}{|F|}, \end{cases} \quad \forall v \in V. \tag{16}$$

Similar as Eqs. (5) and (6), Eqs. (15) and (16) ensure that the values of x_v and g_v are set correctly.

$$\begin{cases} e_v \leq \gamma_v, \\ e_v \leq \displaystyle\sum_{k=1}^{|F|}\sum_{c\in C_k}\sum_{u:(v,u)\in E} \lambda_{k,c}\cdot\rho_{k,c}^{v,u} + n_v^d + (1 - g_v - x_v)\cdot|F|, \ (\xi_v), \\ e_v \geq \displaystyle\sum_{k=1}^{|F|}\sum_{c\in C_k}\sum_{u:(v,u)\in E} \lambda_{k,c}\cdot\rho_{k,c}^{v,u} + n_v^d - (1 - g_v - x_v)\cdot|F|, \ (-\varphi_v), \\ e_v \leq \displaystyle\sum_{u:(v,u)\in E} f_{v,u} + n_v^d + (g_v + x_v)\cdot|F|, \\ e_v \geq \displaystyle\sum_{u:(v,u)\in E} f_{v,u} + n_v^d - (g_v + x_v)\cdot|F|, \end{cases} \quad \forall v \in V. \tag{17}$$

Eq. (17) ensures that the number of flow entries installed on each switch does not exceed its memory capacity.

$$\begin{cases} \tilde{y_v} \geq 0, \\ \tilde{y_v} \leq 2 + \dfrac{|F|}{m}\cdot(x_v + g_v), \\ \tilde{y_v} \leq \dfrac{\sum\limits_{k=1}^{|F|}\sum\limits_{c\in C_k}\sum\limits_{v:(v,u)\in E} [(m\cdot b_k + a_v)\cdot\rho_{k,c}^{v,u}]\cdot\lambda_{k,c}}{m\cdot a_v} + \dfrac{d_v + n_v^d}{m}, \quad (\iota_v), \\ \tilde{y_v} \geq \dfrac{\sum\limits_{k=1}^{|F|}\sum\limits_{c\in C_k}\sum\limits_{v:(v,u)\in E} [(m\cdot b_k + a_v)\cdot\rho_{k,c}^{v,u}]\cdot\lambda_{k,c}}{m\cdot a_v} + \dfrac{d_v + n_v^d}{m} - (2 + \dfrac{|F|}{m})\cdot(1 - x_v - g_v), \quad (-\kappa_v), \end{cases} \quad \forall v \in V. \tag{18}$$

Eq. (18) ensures that the contribution of each node to the network monitoring with SR-INT is determined correctly. As when building the CG model, we need to calculate the dual variable of each constraint on variable $\lambda_{k,c}$ in the master problem, we replace the variable ω_v in the original MILP with its expression based on other variables in Eq. (18).

2) Pricing Problem: In *Algorithm* 2, by solving the new ILP-PPs built in each iteration, we check whether the value of the objective in Eq. (11) can be further reduced. If yes, there is at least one ILP-PP whose objective value is negative (*i.e.*, $\min_k(\phi_k) < 0$). Then, we can generate $|F|$ new columns based on the solutions of the ILP-PPs and include them in $\{C_k, \forall k \in [1, |F|]\}$. Otherwise, if the solutions of the ILP-PPs suggest that the value of the objective in Eq. (11) cannot be reduced any more, the iterations in the CG should be terminated. Based on the aforementioned considerations, we formulate the ILP-PP of each flow in F as follows.

Variables:

- $c^k_{u,v}$: the boolean variable whose definition is the same as that in Section IV.

The relation between $c^k_{u,v}$ and $\rho^{u,v}_{k,c}$ (*i.e.*, the parameter for denoting a column c) is

$$c^k_{u,v} = \rho^{u,v}_{k,c}, \quad \forall k \in [1,|F|],\ (u,v) \in E. \tag{19}$$

Eq. (19) ensure that in each iteration of the CG, the ILP-PP of a flow only considers one routing path for it.

Objective:

According to the relation between the CG's primal and dual problems, the reduction on the objective due to $\lambda_{k,c}$ is

$$\begin{aligned}&\frac{\alpha}{W}\cdot\left(\sum_{(u,v)\in E} b_k\cdot\rho^{u,v}_{k,c}\right)-\epsilon_k+\sum_{v\in V}\sum_{(u,v)\in E}(\varphi_v-\xi_v)\cdot\rho^{u,v}_{k,c}\\&+\sum_{(u,v)\in E}\left(\frac{\nu_{u,v}}{|F|}-\mu_{u,v}\right)\cdot\rho^{u,v}_{k,c}+\sum_{(u,v)\in E}\xi_{u,v}\cdot b_k\cdot\rho^{u,v}_{k,c}\\&+\frac{\beta\cdot\rho^{u,v}_{k,c}}{m\cdot a_v}\cdot\sum_{v\in V}\sum_{v:(v,u)\in E}(m\cdot b_k+a_v)\cdot(\kappa_v-\iota_v).\end{aligned} \tag{20}$$

We then substitute Eq. (19) into Eq. (20) and get the objective of the pricing problem (ILP-PP) as

$$\text{Minimize } \phi_k = \sum_{(u,v)\in E}\left(\frac{\alpha\cdot b_k}{W}+\zeta_{u,v}+\pi_v+\upsilon_v+\chi_{u,v}\right)c^k_{u,v}-\epsilon_k, \tag{21}$$

where we introduce the following notations to simplify it,

$$\begin{cases}\zeta_{u,v}=\dfrac{\nu_{u,v}}{|F|}-\mu_{u,v},\\ \pi_v=\varphi_v-\xi_v,\\ \upsilon_v=\dfrac{\beta\cdot(m\cdot b_k+a_v)\cdot(\kappa_v-\iota_v)}{m\cdot a_v}.\\ \chi_{u,v}=\xi_{u,v}\cdot b_k.\end{cases} \tag{22}$$

Constraints:

The ILP-PP reuses the constraints defined in Eqs. (2), (4) and (7) in Section IV.

Finally, by observing the formulation of the ILP-PP, we can see that it is equivalent to finding the least-weighted path for one flow in a network whose links have preset positive weights. This problem can solved exactly with the well-known Dijkstra algorithm (*i.e.*, a linear time algorithm). Therefore, in *Line* 15 of *Algorithm* 2, we can use the Dijkstra algorithm to solve the optimization described by ILP-PP time-efficiently.

C. Theoretical Analysis of CG-based Algorithm

For the master problem (MP), we generate a restricted master problem (RMP) based on it and evaluate the reduced costs only by implicit enumeration. Hence, as long as the set $\{C_k, \forall k \in [1,|F|]\}$ is finite, our CG-based algorithm is accurate [47]. Since for each flow $f_k \in F$, the number of its feasible paths in the PDP network $G(V,E)$ is finite, we can prove that the set $\{C_k, \forall k \in [1,|F|]\}$ is finite too. To this end, our CG-based algorithm will use a finite number of iterations to solve the RMP, which verifies its convergence.

In our CG-based algorithm (*Algorithm* 2), *Lines* 4-9 are for obtaining an initial feasible solution, which can be completed in polynomial time. *Line* 10 builds the LP relaxation of MILP-MP, and it can also be accomplished in polynomial time [48]. As for *Lines* 11-23, the major contributor to the time complexity is solving the ILP-PP, but as we use the Dijkstra algorithm for it, the ILP-PP can be solved with a complexity of $O(V^2)$. Hence, *Lines* 11-23 run in polynomial time too. *Lines* 24-25 need to solve the MILP-MP, which is the only part in *Algorithm* 2 that might not be completed in polynomial time. But compared to the original MILP in Section IV-A, the MILP-MP has fewer variables and constraints, and thus it takes less time to solve. Specifically, the MILP-MP reduces the numbers of variables and constraints by $(|V|+|F|\cdot(|V|^2-|C_k|))$ and $((|V|-1)\cdot|F|+2\cdot|V|^2+|V|)$, respectively. On the other hand, the optimization in the MILP-MP can also be solved by the G-PR algorithm, *i.e.*, inputting the column set C_k of each flow f_k in the MILP-MP in *Algorithm* 1. This can make *Algorithm* 2 more time-efficient but also degrade its performance.

We define the solution of the LP relaxation in *Algorithm* 2 and the final integer solution as z_{LP} and z, respectively.

Theorem 2: Algorithm 2 is an approximation algorithm for the SR-INT orchestration problem defined in Section IV-A, and the upper bound on its relative error is $\frac{z-z_{\text{LP}}}{z_{\text{LP}}}$.

Proof: We assume that the exact solution of the MILP in Section IV-A is z^*. Since the original problem is for minimization, the solution obtained by the MILP-MP after LP relaxation (*i.e.*, z_{LP}) is the lower bound of the original problem's solution (*i.e.*, z^*). Meanwhile, as *Algorithm* 2 obtains a feasible solution to the original problem, z provides an upper-bound on z^*. Therefore, the upper bound on the relative error of *Algorithm* 2 can be computed as

$$\eta = \frac{z-z^*}{z^*} \le \frac{z-z_{\text{LP}}}{z_{\text{LP}}}, \tag{23}$$

which proves the approximation of *Algorithm* 2. ■

VI. Performance Evaluations

In this section, we discuss the numerical simulations for evaluating the performance of our proposals.

A. Simulation Setup

We use two topologies for the simulations, which are the 14-node NSFNET topology [49, 50] and a random topology (RT-50) that is generated with the GT-ITM tool [51, 52] and consists of 50 nodes and 100 links. In each topology, we assume that the bandwidth capacity on each link (u,v) is evenly distributed as $B_{u,v} \in [2,3]$ Gbps and the switch on each node v can allocate $\gamma_v \in [10,15]$ flow entries to SR-INT. Note that, the resource capacities mentioned above are just for the service flows on which will be implemented SR-INT, while the PDP network can have much more resources for other network services. Also, we assign a relatively small value to the memory capacity of each switch (γ_v) for highlighting the benefit brought by the flow converging in SR-INT.

The source and destination of each flow $f_k \in F$ are randomly selected from V, while its bandwidth demand follows the distribution of practical cases [53]. Specifically, we select the bandwidth demands from three ranges: $[0,10]$, $[10,100]$, and

$[100, 1000]$ Mbps, and make sure that the ratios of the flows in F using the ranges are 50%, 30%, and 20%, respectively. As for the coefficients α and β in the optimization objective in Eq. (1), we assume that they satisfy $\alpha+\beta=1$, and will run simulations with different settings of α and β to study their effects on the performance of SR-INT orchestration.

The simulations consider the MILP formulated in Section IV-A, G-PR, and our CG-based approximation algorithm (CG). In addition to the algorithms that are designed in this work, we also adopt two benchmarks from the literature: the K-MILP in [29] and INTO-CH in [33]. In the simulations, we solve the MILPs and the CG with Gurobi toolbox [54], and implement the other algorithms with C++. The simulation environment is a computer with 2.10 GHz Intel Xeon Silver 4110 CPU and 64 GB memory. To ensure enough statistical accuracy, we make the simulations average the results from 5 and 10 independent runs [55, 56] to get each data point for the scenarios with NSFNET and RT-50 topologies, respectively.

B. Simulations with NSFNET

We first consider the NSFNET topology and have $\alpha \gg \beta$ to make minimizing the total bandwidth usage of flows in F as the primary objective. Fig. 2 shows the performance comparisons of the algorithms, where CG-LP stands for the result obtained by solving the RMP (*i.e.*, the lower-bound of the exact solution). In Fig. 2(a), we can see that when the number of flows is 4, MILP, CG and G-PR have similar performance. But when the number of flows increases, CG can still approximate the optimal result provided by the MILP, while G-PR cannot follow the trend of MILP as CG does. Meanwhile, we can also see that the gap between CG and CG-LP is always larger than that between CG and MILP, which verifies CG's relative error derived in Eq. (23). Therefore, for the large-scale SR-INT orchestration problems that MILP has become intractable, we can use the result from CG-LP as a reasonably good baseline to approximate the exact solution.

When comparing the results of MILP with those of CG and G-PR in Figs. 2(b), 2(d) and 2(e), we find that MILP consumes less bandwidth and similar flow entries but uses a shorter average path length. Meanwhile, it can be seen that compared with K-MILP, the three algorithms from this work (*i.e.*, MILP, CG and G-PR) use less network resources. Specifically, related to K-MILP, our algorithms reduce the bandwidth usage, the number of flow entries, and the average path length by 16%, 14%, and 18% on average, respectively. In Fig. 2(c), we can see that as the number of flows increases, the monitoring coverage of SR-INT also increases, and MILP can monitor more nodes than CG and G-PR.

Fig. 2(f) compares the average number of monitored links from G-PR and INTO-CH. The reason why we choose these two algorithms is because INTO-CH needs to have the routing paths of flows predetermined and we let INTO-CH use the routing paths obtained by G-PR. Since we have $\alpha \gg \beta$ (*i.e.*, the primary objective is to minimize the total bandwidth usage of flows in F), G-PR performs slightly worse than INTO-CH in Fig. 2(f). But as the number of flows increases, the performance gap between G-PR and INTO-CH decreases.

Next, we have $\beta \gg \alpha$ to change the primary objective as maximizing the total contribution of nodes being monitored. Fig. 3 shows the simulation results, which in general illustrate similar trends as those in Fig. 2, *i.e.*, MILP performs the best and CG can approximate the results from MILP well. When comparing the results in Figs. 2(c) and 3(c), we can see that when the value of β is relatively large, our three algorithms tend to monitor more nodes and can cover almost all the nodes with only a small number of flows in F. However, as the primary objective is to maximize the total contribution of nodes being monitored, our algorithms use more network resources than K-MILP in Figs. 3(b), 3(d) and 3(e). In Fig. 3(f), G-PR still monitors less links than INTO-CH, but the gap between them is much smaller than that in Fig. 2(f).

The results in Figs. 2 and 3 indicate that the settings of α and β can affect the performance gaps between our proposals and the benchmarks (*i.e.*, K-MILP and INTO-CH). When α dominates, our proposals treat minimizing the total bandwidth usage of flows in F as the primary objective. Hence, they can provide shorter average path lengths per flow and consume smaller numbers of flow entries per switch than K-MILP in Figs. 2(d) and 2(e), respectively. However, the shorter average path lengths per flow achieved by our proposals also led to smaller number of monitored links than INTO-CH in Fig. 2(f). On the other hand, when we use a large β to make maximizing the total contribution of nodes being monitored as the primary objective, our proposals try to cover more "critical" nodes with path planning. Therefore, they provide longer average path lengths per flow and consume larger numbers of flow entries per switch than K-MILP in Figs. 3(d) and 3(e), respectively. Meanwhile, the gaps on number of monitored links between G-PR and INTO-CH in Fig. 3(f) also becomes smaller.

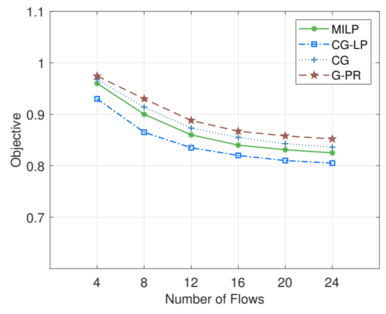
To investigate more on the effect of the ratio between α and β on the performance of SR-INT orchestration, we fix the number of flows as $|F| = 8$ and select $\frac{\alpha}{\beta}$ from $\{\frac{1}{125}, \frac{1}{25}, \frac{1}{5}, 1, 5, 25, 125\}$. Fig. 4 shows the simulation results. It can be seen that the objective values from our three algorithms increase with $\frac{\alpha}{\beta}$ and the gaps between them decrease with $\frac{\alpha}{\beta}$. In Fig. 4(b), all the algorithms use less bandwidth when $\frac{\alpha}{\beta}$ is larger and there is a rapid decrease when the ratio increases from 1 to 5. Also, it is interesting to observe that when α is relatively small, MILP uses the most bandwidth resources, but when α increases, MILP eventually uses the least bandwidth resources. This clearly shows the effect of α and β on balancing the two objectives in Eq. (1). The similar effect can also be seen in Figs. 4(c) and 4(d).

The running time of the algorithms is listed in Table I. MILP and K-MILP take relatively long time to run, followed by CG, the running time of G-PR and INTO-CH is the shortest.

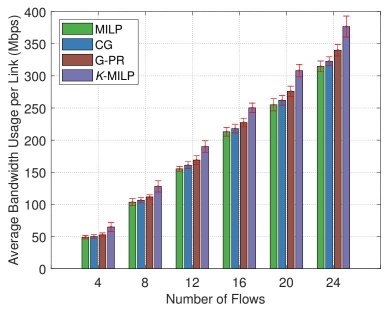
C. Simulations with RT-50

For the simulations with the RT-50 topology, we do not consider MILP and K-MILP for their high time complexity. Meanwhile, to adapt the larger number of flows in RT-50, we increase the number of flow entries per switch as $\gamma_v \in [15, 20]$.

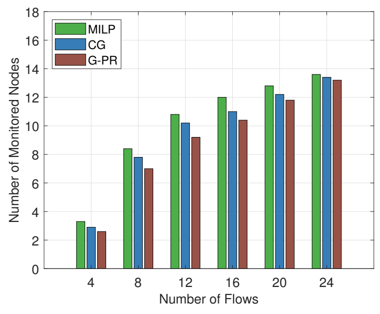
We still consider the cases of $\alpha \gg \beta$ first, and show the simulation results in Fig. 5. In Fig. 5(a), CG always



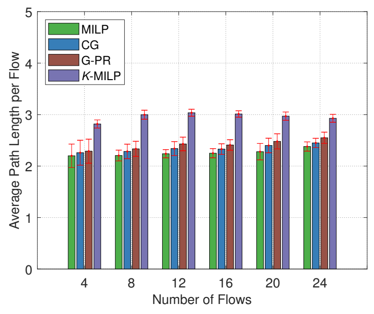
(a) Optimization objective



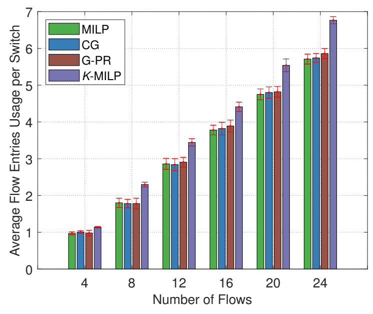
(b) Bandwidth usage



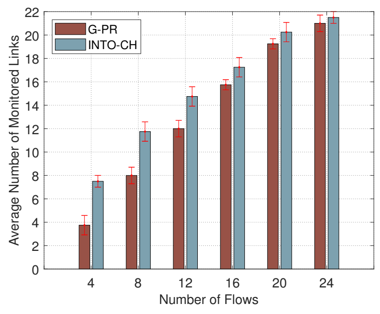
(c) INT monitoring coverage



(d) Flow path length

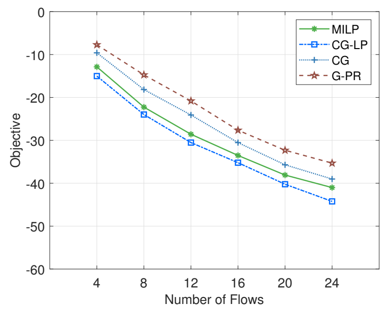


(e) Switch memory usage

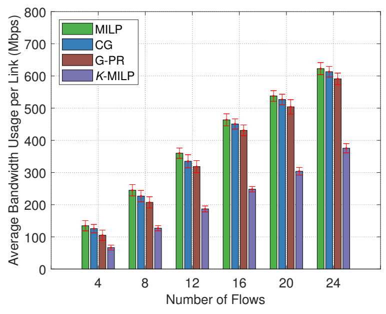


(f) Monitored links

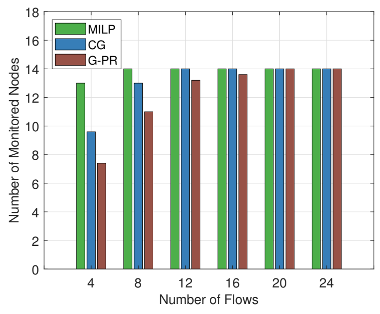
Fig. 2. Results of simulations with NSFNET topology ($\alpha \gg \beta$).



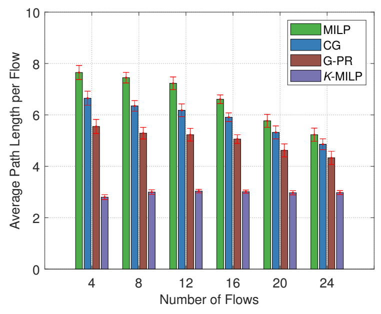
(a) Optimization objective



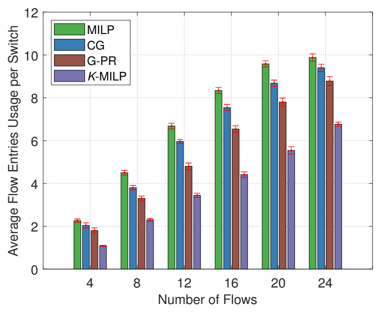
(b) Bandwidth usage



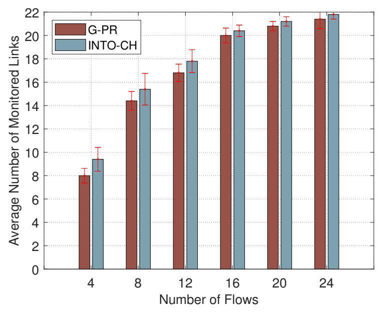
(c) INT monitoring coverage



(d) Flow path length

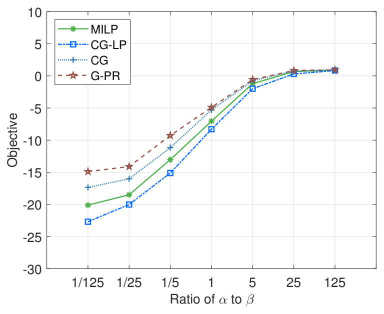


(e) Switch memory usage

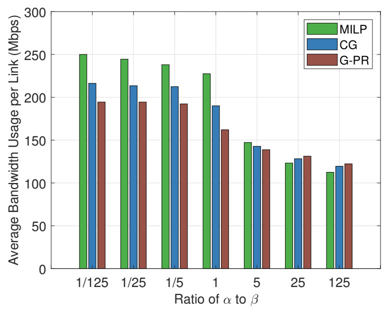


(f) Monitored links

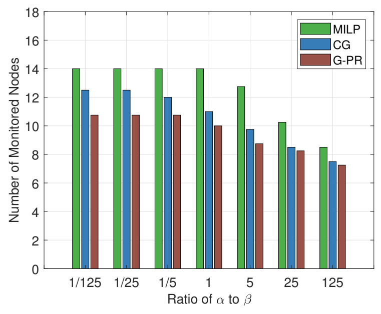
Fig. 3. Results of simulations with NSFNET topology ($\alpha \ll \beta$).



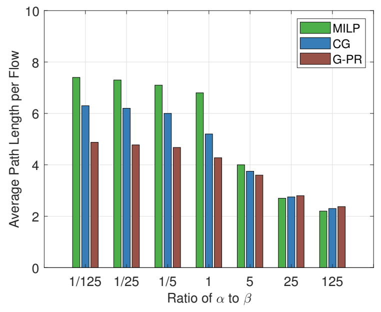
(a) Optimization objective



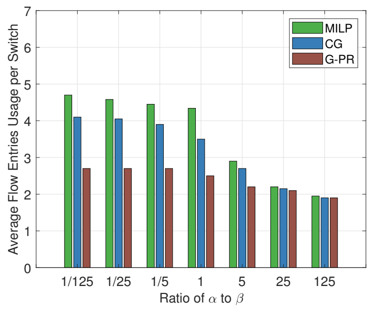
(b) Bandwidth usage



(c) INT monitoring coverage

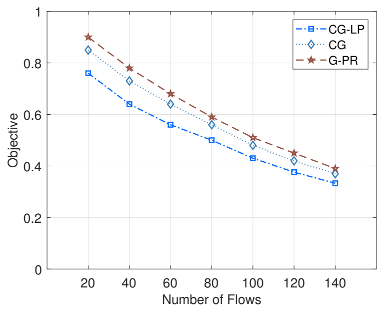


(d) Flow path length

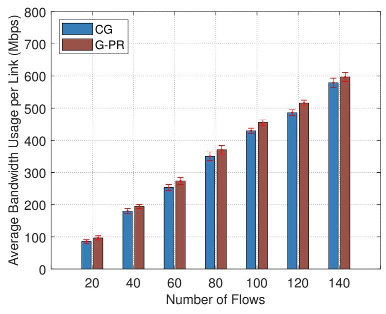


(e) Switch memory usage

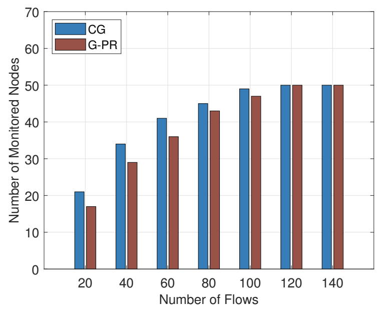
Fig. 4. Results of simulations with NSFNET topology (various ratios between α and β).



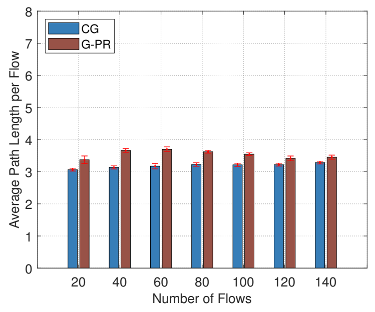
(a) Optimization objective



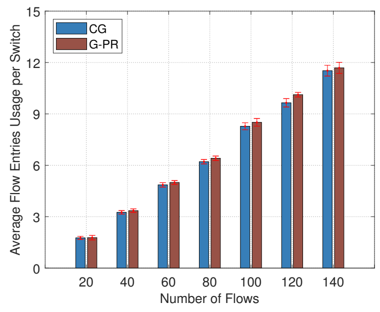
(b) Bandwidth usage



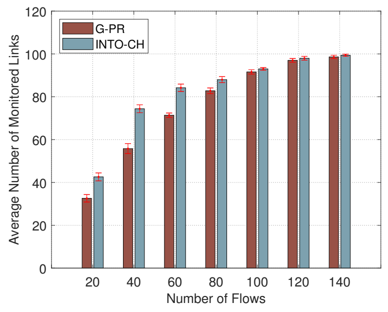
(c) INT monitoring coverage



(d) Flow path length

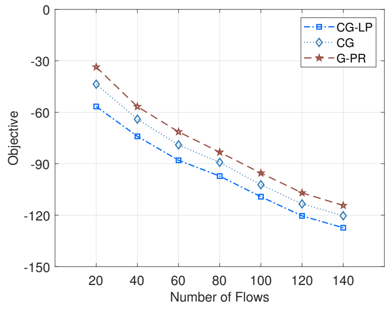


(e) Switch memory usage

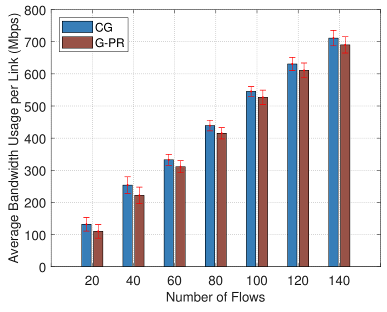


(f) Monitored links

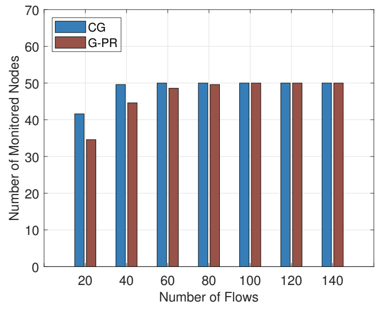
Fig. 5. Results of simulations with RT-50 topology ($\alpha \gg \beta$).



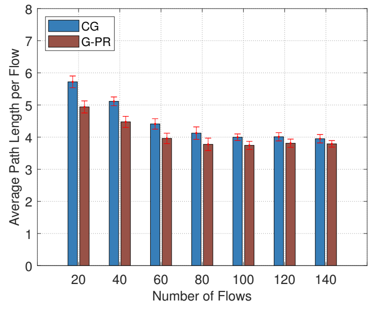
(a) Optimization objective



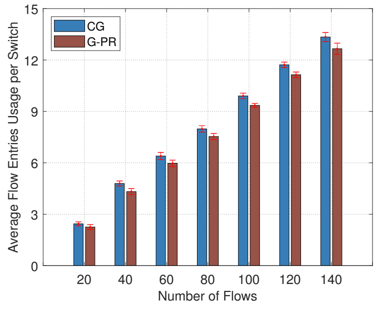
(b) Bandwidth usage



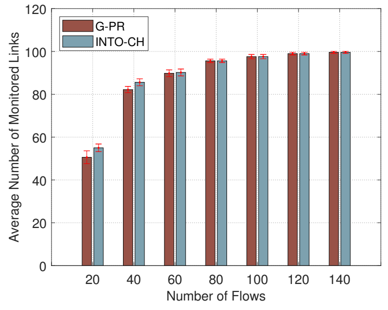
(c) INT monitoring coverage



(d) Flow path length

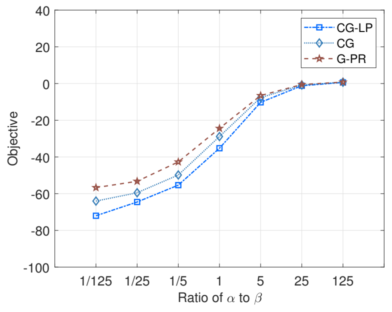


(e) Switch memory usage

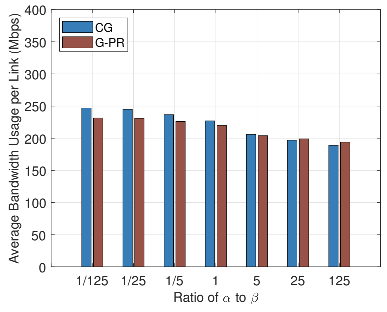


(f) Monitored links

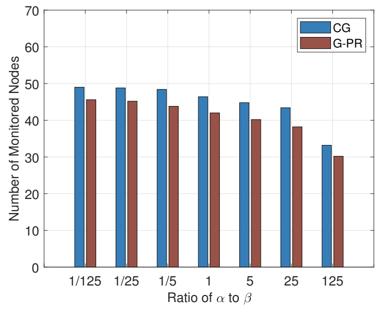
Fig. 6. Results of simulations with RT-50 topology ($\alpha \ll \beta$).



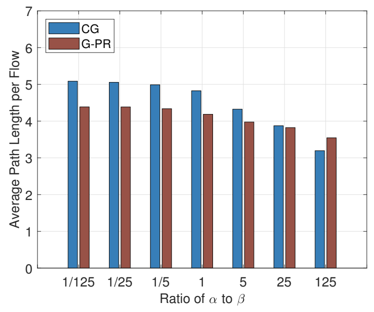
(a) Optimization objective



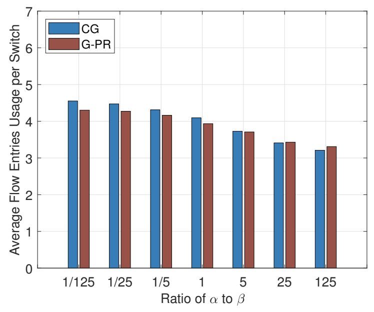
(b) Bandwidth usage



(c) INT monitoring coverage



(d) Flow path length



(e) Switch memory usage

Fig. 7. Results of simulations with RT-50 topology (various ratios between α and β).

TABLE I
AVERAGE RUNNING TIME WITH NSFNET TOPOLOGY (SECONDS)

$\|F\|$	4	8	12	16	20	24
MILP	86.24	192.82	450.45	992.25	2213.67	4535.29
G-PR	0.06	0.08	0.09	0.10	0.11	0.12
CG	0.82	1.14	1.91	2.82	3.55	4.32
K-**MILP**	12.42	20.23	32.34	46.72	76.43	124.86
INTO-CH	0.02	0.03	0.03	0.04	0.05	0.06

outperforms G-PR but takes longer running time as indicated in Table II. CG consumes less bandwidth in Fig. 5(b), covers more nodes in Fig. 5(c), use a shorter average path length in Fig. 5(d), and occupies less flow entries in Fig. 5(e). Meanwhile, it is interesting to notice that in Fig. 5(d), the average path length of G-PR reaches the longest when we have $|F| = 60$, but as the number of flows continues to grow, the average path length begins to decrease. This is because when the number of flows is relatively small, G-PR makes flows with smaller bandwidth go through longer paths to monitor more nodes without significantly increasing the overall bandwidth usage. However, when there are many flows in F, G-PR does not need to make more flows go through long paths, because most of the nodes in the PDP network have already been monitored. This analysis can be verified by checking Fig. 5(c), where both CG and G-PR can monitor all the nodes when we have $|F| = 120$. In Fig. 5(f), the results of G-PR and INTO-CH show similar trends as those in Fig. 2(f).

Next, we have $\beta \gg \alpha$ and the performance of CG, G-PR and INTO-CH is shown in Fig. 6. By comparing Figs. 5(c) and 6(c), we can see that the numbers of nodes monitored by both algorithms increase when the value of β increases. When the number of flows reaches 60, CG can monitor all the nodes, but G-PR cannot achieve this before there are 100 flows. In Fig. 6(d), both CG and G-PR use longer average paths than they do in Fig. 5(d), respectively. Fig. 6(f) shows that G-PR provides almost the same network monitoring capability as INTO-CH, and when $|F|$ increases, the number of links monitored by G-PR and INTO-CH increases simultaneously. Fig. 7 show the cases in which we fix $|F|$ as 40 and change $\frac{\alpha}{\beta}$. The results show similar trends as those in Fig. 4.

TABLE II
AVERAGE RUNNING TIME WITH RT-50 TOPOLOGY (SECONDS)

$\|F\|$	20	40	60	80	100	120	140
G-PR	0.21	0.28	0.38	0.50	0.61	0.76	0.93
CG	7.23	15.91	25.77	39.14	52.75	71.14	86.26
INTO-CH	0.13	0.14	0.16	0.17	0.19	0.20	0.21

VII. CONCLUSION

In this work, we studied the problem of SR-INT orchestration, which tries to balance the tradeoff between bandwidth usage and coverage of network monitoring by planning the routing paths of flows and partition the paths for SR. We first formulated an MILP to model the problem and solve it exactly. Then, in order to reduce the time complexity of problem solving, we designed both a greedy algorithm based on path ranking and a CG-based approximation algorithm. Simulation results showed that our algorithms outperformed existing SR-based algorithm when the primary objective is to minimize bandwidth usage, and provided similar network monitoring coverage as the existing algorithm for INT orchestration.

ACKNOWLEDGMENTS

This work was supported in part by the National Key R&D Program of China (2020YFB1806400), NSFC project 61871357, SPR Program of CAS (XDC02070300), and Fundamental Funds for Central Universities (WK3500000006).

REFERENCES

[1] P. Lu *et al.*, "Highly-efficient data migration and backup for Big Data applications in elastic optical inter-datacenter networks," *IEEE Netw.*, vol. 29, pp. 36–42, Sept./Oct. 2015.

[2] W. Lu *et al.*, "AI-assisted knowledge-defined network orchestration for energy-efficient data center networks," *IEEE Commun. Mag.*, vol. 58, pp. 86–92, Jan. 2020.

[3] P. Marsch *et al.*, "5G radio access network architecture: Design guidelines and key considerations," *IEEE Commun. Mag.*, vol. 54, pp. 24–32, Nov. 2016.

[4] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: an intellectual history of programmable networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 87–98, Apr. 2014.

[5] P. Bosshart *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 87–95, Jul. 2014.

[6] S. Li *et al.*, "Protocol oblivious forwarding (POF): Software-defined networking with enhanced programmability," *IEEE Netw.*, vol. 31, pp. 12–20, Mar./Apr. 2017.

[7] Z. Zhu, W. Lu, L. Zhang, and N. Ansari, "Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing," *J. Lightw. Technol.*, vol. 31, pp. 15–22, Jan. 2013.

[8] L. Gong *et al.*, "Efficient resource allocation for all-optical multicasting over spectrum-sliced elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. 836–847, Aug. 2013.

[9] Y. Yin *et al.*, "Spectral and spatial 2D fragmentation-aware routing and spectrum assignment algorithms in elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. A100–A106, 2013.

[10] X. Chen *et al.*, "Deep-RMSA: A deep-reinforcement-learning routing, modulation and spectrum assignment agent for elastic optical networks," in *Proc. of OFC 2018*, pp. 1–3, Mar. 2018.

[11] L. Gong and Z. Zhu, "Virtual optical network embedding (VONE) over elastic optical networks," *J. Lightw. Technol.*, vol. 32, pp. 450–460, Feb. 2014.

[12] J. Liu *et al.*, "On dynamic service function chain deployment and readjustment," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, pp. 543–553, Sept. 2017.

[13] L. Gong, H. Jiang, Y. Wang, and Z. Zhu, "Novel location-constrained virtual network embedding (LC-VNE) algorithms towards integrated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 24, pp. 3648–3661, Dec. 2016.

[14] R. Proietti *et al.*, "Experimental demonstration of machine-learning-aided QoT estimation in multi-domain elastic optical networks with alien wavelengths," *J. Opt. Commun. Netw.*, vol. 11, pp. A1–A10, Jan. 2019.

[15] A. Morton, "Active and passive metrics and methods (with hybrid types in-between)," *RFC 7799*, May 2016. [Online]. Available: https://tools.ietf.org/html/rfc7799.

[16] J. Quittek and K. White, "Definitions of managed objects for remote Ping, Traceroute, and lookup operations," *RFC 4560*, Jun. 2006. [Online]. Available: https://tools.ietf.org/html/rfc4560.

[17] P. Phaal, S. Panchen, and N. McKee, "InMon corporation's sFlow: A method for monitoring traffic in switched and routed networks," *RFC 3176*, Sept. 2001. [Online]. Available: https://tools.ietf.org/html/rfc3176.

[18] M. Allman and V. Paxson, "A reactive measurement framework," in *Proc. of PAM 2008*, pp. 1–10, Apr. 2008.

[19] C. Kim *et al.*, "In-band network telemetry (INT)," *Tech. Spec.*, Jun. 2016. [Online]. Available: https://p4.org/assets/INT-current-spec.pdf.

[20] N. Van Tu, J. Hyun, and J. Hong, "Towards ONOS-based SDN monitoring using in-band network telemetry," in *Proc. of APNOMS 2017*, pp. 76–81, Sept. 2017.

[21] Segment Routing. [Online]. Available: https://www.segment-routing.net/.

[22] SRv6. [Online]. Available: https://www.segment-routing.net/tutorials/2017-12-05-srv6-introduction/.

[23] Q. Zheng, S. Tang, B. Chen, and Z. Zhu, "Highly-efficient and adaptive network monitoring: When INT meets segment routing," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, pp. 2587–2597, Sept. 2021.

[24] Y. Kim, D. Suh, and S. Pack, "Selective in-band network telemetry for overhead reduction," in *Proc. of CloudNet 2018*, pp. 1–3, Oct. 2018.

[25] S. Tang *et al.*, "Sel-INT: A runtime-programmable selective in-band network telemetry system," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, pp. 708–721, Jun. 2020.

[26] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," *RFC 3031*, Jan. 2001. [Online]. Available: https://tools.ietf.org/html/rfc3031.

[27] R. Bhatia, F. Hao, M. Kodialam, and T. Lakshman, "Optimized network traffic engineering using segment routing," in *Proc. of INFOCOM 2015*, pp. 657–665, Apr. 2015.

[28] E. Moreno, A. Beghelli, and F. Cugini, "Traffic engineering in segment routing networks," *Comput. Netw.*, vol. 114, pp. 23–31, Jan. 2017.

[29] X. Li and K. Yeung, "Traffic engineering in segment routing using MILP," in *Proc. of ICC 2019*, pp. 1–6, Jun. 2019.

[30] T. Schuller *et al.*, "Traffic engineering using segment routing and considering requirements of a carrier IP network," *IEEE/ACM Trans. Netw.*, vol. 26, pp. 1851–1864, Jul. 2018.

[31] M. Jadin, F. Aubry, P. Schaus, and O. Bonaventure, "CG4SR: Near optimal traffic engineering for segment routing with column generation," in *Proc. of INFOCOM 2019*, pp. 1333–1341, Apr. 2019.

[32] V. Pereira, M. Rocha, and P. Sousa, "Traffic engineering with three-segments routing," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, pp. 1896–1909, May 2020.

[33] J. Marques, M. Luizelli, R. Filho, and L. Gaspary, "An optimization-based approach for efficient network monitoring using in-band network telemetry," *J. Internet Serv. Appl.*, vol. 10, pp. 1–20, Jun. 2019.

[34] Z. Liu *et al.*, "NetVision: Towards network telemetry as a service," in *Proc. of ICNP 2018*, pp. 247–248, Sept. 2018.

[35] T. Pan *et al.*, "INT-path: Towards optimal path planning for in-band network-wide telemetry," in *Proc. of INFOCOM 2019*, pp. 487–495, May 2019.

[36] Y. Lin *et al.*, "NetView: Towards on-demand network-wide telemetry in the data center," in *Proc. of ICC 2020*, pp. 1–6, Jun. 2020.

[37] G. Simsek, D. Ergenc, and E. Onur, "Efficient network monitoring via in-band telemetry," in *Proc. of DRCN 2021*, pp. 1–6, Apr. 2021.

[38] A. Castro *et al.*, "Near-optimal probing planning for in-band network telemetry," *IEEE Commun. Lett.*, vol. 25, pp. 1630–1634, Jan. 2021.

[39] Z. Zhang, W. Su, and L. Tan, "In-band network telemetry task orchestration based on multi-objective optimization," in *Proc. of APNOMS 2021*, pp. 354–357, Sept. 2021.

[40] B. Niu *et al.*, "Visualize your IP-over-optical network in realtime: A P4-based flexible multilayer in-band network telemetry (ML-INT) system," *IEEE Access*, vol. 7, pp. 82 413–82 423, 2019.

[41] R. Basat *et al.*, "PINT: Probabilistic in-band network telemetry," in *Proc. of ACM SIGCOMM 2020*, pp. 662–680, Jul. 2020.

[42] D. Hu *et al.*, "Flexible flow converging: A systematic case study on forwarding plane programmability of protocol-oblivious forwarding (POF)," *IEEE Access*, vol. 4, pp. 4707–4719, 2016.

[43] 100G in-band network telemetry with Netcope P4. [Online]. Available: https://www.netcope.com/Netcope/media/content/100G-In-band-Network-Telemetry-With-Netcope-P4.pdf.

[44] Building a PoC of segment routing at 100G using FPGA Smart NIC and P4 language. [Online]. Available: https://www.netcope.com/Netcope/media/content/100G-In-band-Network-Telemetry-With-Netcope-P4.pdf.

[45] M. Garey and D. Johnson, *Computers and Intractability: a Guide to the Theory of NP-Completeness*. W. H. Freeman & Co. New York, 1979.

[46] R. Karp, "On the computational complexity of combinatorial problems," *Netw.*, vol. 5, pp. 45–68, Jan. 1975.

[47] G. Desaulniers, J. Desrosiers, and M. Solomon, *Column Generation*. Springer, 2005.

[48] D. Goldfarb and M. Todd, "Modifications and implementation of the ellipsoid algorithm for linear programming," *Math. Program.*, vol. 23, pp. 1–19, 1982.

[49] W. Fang *et al.*, "Joint spectrum and IT resource allocation for efficient vNF service chaining in inter-datacenter elastic optical networks," *IEEE Commun. Lett.*, vol. 20, pp. 1539–1542, Aug. 2016.

[50] Q. Lv, F. Zhou, and Z. Zhu, "On the bilevel optimization to design control plane for SDONs in consideration of planned physical-layer attacks," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, pp. 3221–3230, Sept. 2021.

[51] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. of INFOCOM 1996*, pp. 594–602, Mar. 1996.

[52] W. Shi, Z. Zhu, M. Zhang, and N. Ansari, "On the effect of bandwidth fragmentation on blocking probability in elastic optical networks," *IEEE Trans. Commun.*, vol. 61, pp. 2970–2978, Jul. 2013.

[53] T. Benson, A. Akella, and D. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. of IMC 2010*, pp. 267–280, Nov. 2010.

[54] Gurobi. [Online]. Available: http://www.gurobi.com.

[55] M. Zhang *et al.*, "Bandwidth defragmentation in dynamic elastic optical networks with minimum traffic disruptions," in *Proc. of ICC 2013*, pp. 3894–3898, Jun. 2013.

[56] Q. Sun, P. Lu, W. Lu, and Z. Zhu, "Forecast-assisted NFV service chain deployment based on affiliation-aware vNF placement," in *Proc. of GLOBECOM 2016*, pp. 1–6, Dec. 2016.