A comprehensive survey on Segment Routing Traffic Engineering

Duo Wu, Lin Cui

PII: S2352-8648(22)00018-9

DOI: https://doi.org/10.1016/j.dcan.2022.02.006

Reference: DCAN 358

To appear in: *Digital Communications and Networks*

Received Date: 17 February 2021

Revised Date: 23 November 2021

Accepted Date: 22 February 2022

Digital Communications and Networks(DCN)

HOSTED BY ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/dcan

A Comprehensive Survey on Segment Routing Traffic Engineering

Duo Wu, Lin Cui*

Department of Computer Science, Jinan University, Guangzhou 510632, China

Abstract

Traffic Engineering (TE) enables management of traffic in a manner that optimizes utilization of network resources in an efficient and balanced manner. However, existing TE solutions face issues relating to scalability and complexity. In recent years, Segment Routing (SR) has emerged as a promising source routing paradigm. As one of the most important applications of SR, Segment Routing Traffic Engineering (SR-TE), which enables a *headend* to steer traffic along specific paths represented as ordered lists of instructions called *segment lists*, has the capability to overcome the above challenges due to its flexibility and scalability. In this paper, we conduct a comprehensive survey on SR-TE. A thorough review of SR-TE architecture is provided in the first place, reviewing the core components and implementation of SR-TE such as SR Policy, Flexible Algorithm and SR-native algorithm. Strengths of SR-TE are also discussed, as well as its major challenges. Next, we dwell on the recent SR-TE researches on routing optimization with various intents, e.g., optimization on link utilization, throughput, QoE (Quality of Experience) and energy consumption. Afterwards, node management for SR-TE are investigated, including SR node deployment and candidate node selection. Finally, we discuss the existing challenges of current research activities and propose several research directions worth of future exploration.

KEYWORDS:
Segment Routing, Traffic Engineering, SR Policy, Routing Optimization, Segment List Computation

1. Introduction

As an essential function of optimizing network services in many networks [1], Traffic Engineering (TE) [2] has attracted continuous and widespread attention from both industry and academia. TE can be regarded as a set of solutions or strategies for effective traffic routing to improve network service quality. It usually encompasses the computation of optimized paths for the given source destination pairs [3], in order to achieve the optimization of network performance, e.g., improving network throughput and resource utilization, minimizing congestion.

So far, a great number of solutions have been proposed to implement TE. For example, link weights of Interior Gateway Protocol (IGP) can be appropriately configured to distribute traffic in a balanced manner, where traditional routing protocols like Open Shortest Path First (OSPF) or Intermediate System to Intermediate System (IS-IS) can be exploited [6][7][8]. However, the optimal solution for link weight configuration has been proved to be NP-hard [6] and thus, is computationally expensive. Besides, even though a near optimal solution is possible, the configuration of link weights can be complex notably in large scale networks [9]. Multiprotocol Label Switching (MPLS) has been widely explored to implement TE, due to its explicit routing mechanism and capability to estab-

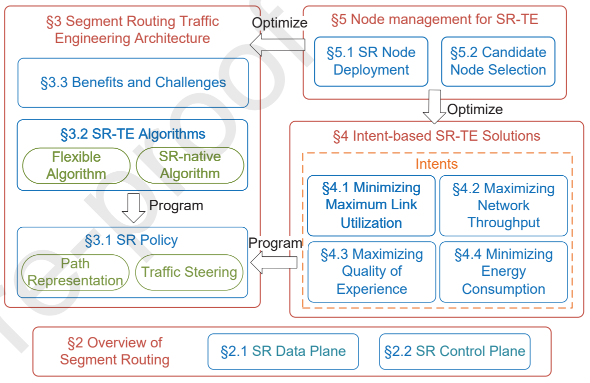
*Lin Cui (Corresponding author) (email:tcuilin@jnu.edu.cn).
[1]Duo Wu is currently an undergraduate in the Department of Computer Science at Jinan University, Guangzhou, China. His current research interests include segment routing networks and traffic engineering. (email:duowu18@outlook.com).
[2]Lin Cui is currently a professor in the Department of Computer Science at Jinan University, Guangzhou, China. He received the Ph.D. degree from City University of Hong Kong in 2013. He has broad interests in networking systems, with focuses on cloud data center networking, software defined networking (SDN), NFV, programmable networking, distributed systems and so on. (email:tcuilin@jnu.edu.cn).

Tab. 1: Comparison of our work and other survey papers.

Works	Main Focus	SR-TE Architecture	SR-TE Researches	SR-TE Node Management
Abdullah et al. [4]	Core features of SR and its key applications in SDN	Concept of SR Policy	Concise descriptions of a few research papers	×
Ventre et al. [5]	Standardization, patents, researches and implementation of SR	Concepts of SR Policy and Flexible Algorithm	Brief summaries of many research works	×
Our work	Comprehensive survey on SR-TE architecture and related researches	SR Policy framework; Definitions, illustrations of SR-TE algorithms; Architectural analysis	Extensive summaries and analysis of existing SR-TE researches based on intents	✓

lish constrained paths [10][11][12]. However, classic MPLS TE depends on signaling protocols such as Resource Reservation Protocol - Traffic Engineering (RSVP-TE) [13] to establish end-to-end tunnels with reserved resources [12], thereby complicating its control plane. Moreover, it also suffers from the scalability issue due to the large overhead caused by maintaining and distributing per-tunnel state information across the network [14]. By decoupling the data plane and control plane, as well as centralizing network intelligence in the controller, Software Defined Networking (SDN) has opened new opportunities for the innovative design of TE strategies [15][16][17]. Successful examples include Google's B4 [18] and Microsoft's SWAN [19]. However, SDN requires to maintain per-flow state information on all switches, which consumes a great number of precious resources of flow table entries, restricting the scalability of TE in large scale networks [20].

In recent years, the source routing paradigm Segment Routing (SR) [21] has emerged as an advanced technique with the capability to overcome the challenges mentioned above. The SR architecture defines the concept of *segment* which identifies an instruction to be executed by network devices [22]. The *headend* is responsible to encapsulate an incoming packet with an ordered list of segments, i.e., *segment list*, to forward it along the specific path. Intermediate nodes only need to process the packet according to the instructions specified by segments [22]. As one of the most important applications of SR, Segment Routing Traffic Engineering (SR-TE) [23] involves the computation and implementation of segment lists according to the intents of network operators, so as to establish explicit paths in the network for traffic steering. Benefiting from the source routing mechanism, intermediate per-flow states are eliminated [24], which allows SR-TE to achieve higher scalability than SDN TE. In addition, SR-TE also simplifies the operations on intermediate nodes as most configurations of a TE solution are performed on headends. Different from MPLS TE, the implementation of SR-TE requires no development of new signaling protocols since the SR



§3 Segment Routing Traffic Engineering Architecture
§3.3 Benefits and Challenges
§3.2 SR-TE Algorithms
Flexible Algorithm
SR-native Algorithm
Program
§3.1 SR Policy
Path Representation
Traffic Steering
Optimize
§5 Node management for SR-TE
§5.1 SR Node Deployment
§5.2 Candidate Node Selection
Optimize
§4 Intent-based SR-TE Solutions
Intents
§4.1 Minimizing Maximum Link Utilization
§4.2 Maximizing Network Throughput
§4.3 Maximizing Quality of Experience
§4.4 Minimizing Energy Consumption
Program
§2 Overview of Segment Routing
§2.1 SR Data Plane
§2.2 SR Control Plane

Fig. 1: Overview of this survey on SR-TE.

architecture can reuse the existing control plane [25]. One major challenge of SR-TE is the extra overhead caused by segments in each packet, which, however, can be effectively mitigated via different measures (see Section 3).

In fact, due to its appealing features, several surveys on SR have been conducted in the past few years. Abdullah et al. [4] highlight the core features of SR and its key applications in SDN, while Ventre et al. [5] extensively review the research activities, implementation and standardization efforts of SR. Both surveys provide a general view on SR, but not much emphasis is placed on SR-TE. Hence, this paper addresses this research gap and conducts a comprehensive survey with the focus on SR-TE, ranging from its fundamental techniques, existing research works to open issues. The comparison between our survey and other works is shown in Table 1. The taxonomy of SR-TE discussed in this paper is shown in Figure 1. The main contributions of this paper are summarized as follows:

1. Thorough introduction and analysis on the SR-TE architecture, including an extensive review of core components of SR-TE, and summary of its key benefits and major challenges (Section 3).
2. Profound insights into the recent SR-TE research with the focus on routing optimization, by categorizing and comparing the optimized solutions

based on their distinct characteristics such as optimization objectives and techniques (Section 4).

3. Detailed exploration of arrangement and methodologies of node management for SR-TE (Section 5), as well as identification of future research directions to facilitate the development of SR-TE (Section 6).

The remainder of this paper is organized as follows. Section 2 provides an overview of SR architecture. Next, Section 3 highlights the core components of SR-TE, demonstrates its implementation and outlines its key benefits and major challenges. Section 4 then provides a comprehensive analysis of the research activities on SR-TE from different perspectives. Section 5 explores the schemes and methodologies of node management for SR-TE, and Section 6 discusses several issues for future research directions of SR-TE. Finally, Section 7 gives a conclusion to this paper.

2. Overview of Segment Routing

This section provides an overview of Segment Routing (SR) architecture, which SR-TE is built up on. Before introducing the main components of SR, some architectural concepts are described as follows:

- *SR domain:* The set of nodes participating in the SR source-routing model defines an SR domain [26]. These nodes can be either physically connected (e.g., a Serivce Provider's network) or remotely connected to each other (e.g., an enterprise Virtual Private Network) [22].

- *Segment:* SR defines a segment as an instruction encoded in the packet header for SR capable nodes to execute [22]. Instructions of segments can be topology or service related. For example, a segment can represent an instruction to forward a packet along the shortest path to a specific destination or deliver it to a given application/service instance. Each segment is associated with an identifier named Segment Identifier (SID), whose form depends on the instantiation of SR data plane. According to its effective scope, a segment can be classified as global segment or local segment [22]. Instruction of a global segment can be recognized and executed by all SR nodes in the SR domain, while instruction of a local segment can be executed only by its originating node. Specially, the SID of global segment, i.e., global SID, must be unique across the SR domain.

- *Segment list:* A segment list refers to an ordered list of segments, which specifies a path to transmit a packet and is used for traffic steering [4]. The first segment in the segment list is called active segment, which specifies the current instruction for SR nodes to execute. In addition to executing instructions of segments, SR nodes can also maintain the segment list via three operations [26]:

  1. *PUSH:* Insert one or more segments in front of the segment list. The first segment in the list will become the new active segment.
  2. *CONTINUE:* Keep the active segment unchanged when its instruction is not completed yet (e.g., a packet has not reached the advertising node of the active segment).
  3. *NEXT:* Activate the next segment in the segment list (if any) as new active segment when the instruction of current active segment is completed (e.g., a packet has reached the advertising node of the active segment).

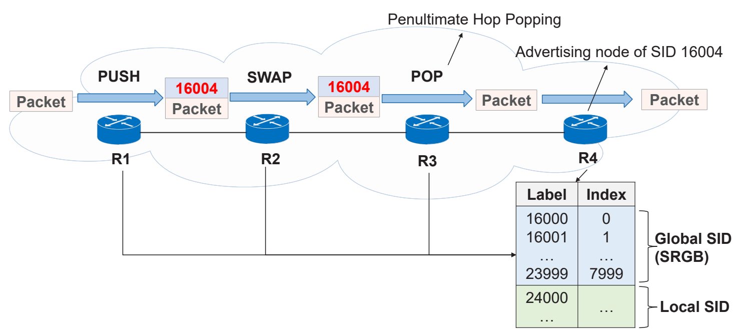
The following of this section will describe two main components of SR architecture: SR data plane (Section 2.1) and SR control plane (Section 2.2), with the focus on MPLS data plane and IGP control plane, respectively.

2.1. SR Data plane

SR can be deployed in the existing MPLS data plane (SR-MPLS) [27], where SID is encoded as MPLS label. The segment list corresponds to the MPLS label stack and the active segment corresponds to the top-most label in the stack. The operations on the segment list can be directly mapped to those on the label stack in MPLS. The PUSH operation is mapped to PUSH, which pushes a number of SIDs on top of the label stack. The CONTINUE operation is mapped to SWAP, which swaps the top-most SID with the same one. The NEXT operation is mapped to POP, which pops the top-most SID out of the stack. Figure 2 provides an illustration of the segment list operations for SR-MPLS instantiation.

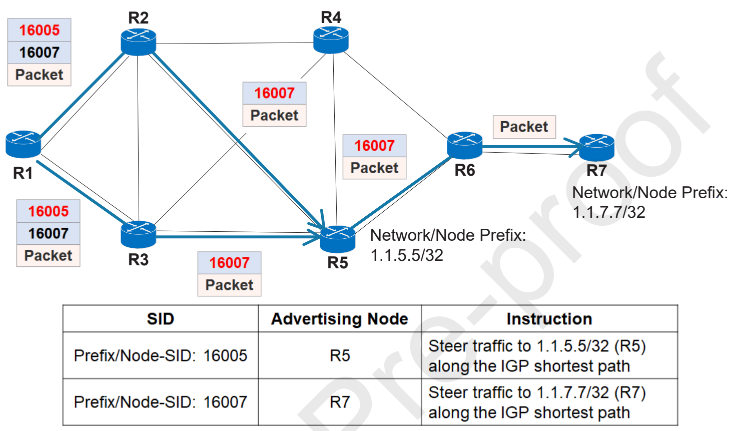
In SR-MPLS, each node in the SR domain is required to preserve a local label range for global SID assignment, which is known as Segment Routing Global Block (SRGB). Each node must advertise their SRGBs along with their own indexes of global SIDs across the domain. For instance, assume that an SR node preserves an SRGB [1000, 2000] and assigns label 1003 to a global SID. Then, the SRGB and index 3 are advertised in the network. Upon receiving the advertisement, other SR nodes can learn the global SID of the sender through simple calculation: $1000 + 3 = 1003$. Note that although SR nodes are allowed to preserve different SRGBs, it is strongly recommended that all SR nodes share the same one for simplicity [26]. For ease of explanation, in this paper, we assume that the SRGB of each SR node is [16000, 23999]. Local SIDs are allocated from the label range outside the SRGB (Figure 2).

In addition to SR-MPLS, SR can also be instantiated in IPv6 data plane (SRv6) [28]. In this case, SID



Label	Index
16000	0
16001	1
…	…
23999	7999
24000 …	…

Fig. 2: Illustrations of segment list operations for SR-MPLS instantiation. The SID colored in red represents the active SID. The segment of SID 16004 represents an instruction to forward traffic to the advertising node R4.



SID	Advertising Node	Instruction
Prefix/Node-SID: 16005	R5	Steer traffic to 1.1.5.5/32 (R5) along the IGP shortest path
Prefix/Node-SID: 16007	R7	Steer traffic to 1.1.7.7/32 (R7) along the IGP shortest path

Fig. 3: Illustrations of Prefix/Node Segment. SIDs colored in red represent active SIDs. R1 pushes <16005, 16007> in the packets and forwards them along ECMP-aware (Equal Cost Multiple Path) paths. R2 and R3 pop SID 16005 out due to PHP (Penultimate Hop Popping) mechanism. Finally, R6 pops out the last SID and delivers the packets to R7.

is encoded as IPv6 address. The segment list corresponds to the list of IPv6 addresses contained in a new type of IPv6 extension header, i.e., Segment Routing Header (SRH) [28]. The active segment is identified by a pointer in SRH.

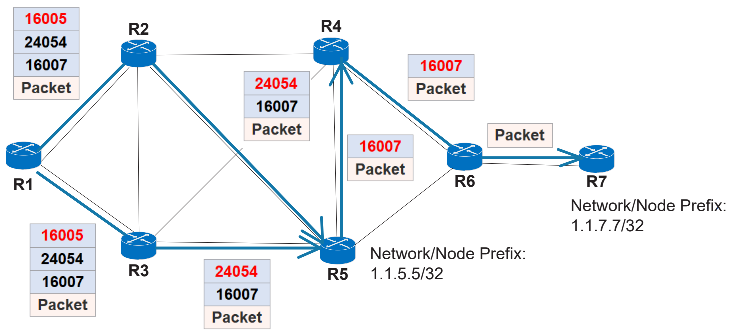
For simplicity, all illustrations of SR in the following of this paper are provided using SR-MPLS as example. However, all mechanisms and ideas can be applied to SRv6 easily.

2.2. *SR Control Plane*

The SR control plane can be implemented with IGP routing protocols (e.g., OSPF and IS-IS). Any path in an IGP domain can be expressed as a segment list. Two important segments, namely IGP-Prefix Segment and IGP-Adjacency Segment, are defined as follows:

- *IGP-Prefix Segment:* The IGP-Prefix Segment (or Prefix Segment) is the global segment attached to a network prefix, which instructs an SR node to forward a packet to the associated prefix along the path calculated with a specific algorithm. The SID of Prefix Segment is also called Prefix-SID. Each Prefix Segment is associated with an algorithm identified by a numeric value within [0, 255] for path calculation. By default, a Prefix Segment is associated with algorithm 0, i.e., the IGP Shortest Path First (SPF) algorithm [22]. In this case, an SR node will forward a packet along the IGP shortest path upon receiving the Prefix-SID. Two subtypes of Prefix Segment are defined: IGP-Node Segment (or Node Segment) and IGP-Anycast Segment (or Anycast Segment). Node Segment is the Prefix Segment attached to a node prefix (e.g., the loopback address of a node), which shares the same semantics as Prefix Segment. Anycast Segment refers to the Prefix Segment attachted to an anycast prefix, which instructs a node to forward traffic to the nearest member(s) of the anycast group. Figure 3 shows an example of Prefix/Node Segment.

- *IGP-Adjacency Segment:* The IGP-Adjacency Segment (or Adjacency Segment) is the segment attached to a unidirectional adjacency (set), which instructs an SR node to steer traffic through the corresponding adjacency (set) regardless of the shortest path. The SID of Adjacency Segment is referred to as Adj-SID. In general, an Adjacency Segment is configured as local segment and thereby only the originating node can execute it. In this case, a global segment (e.g., Prefix/Node Segment) should be placed be-



SID	Advertising Node	Instruction
Prefix/Node-SID: 16005	R5	Steer traffic to 1.1.5.5/32 (R5) along the IGP shortest path
Adj-SID: 24054	R5	Steer traffic towards the interface R5→R4
Prefix/Node-SID: 16007	R7	Steer traffic to 1.1.7.7/32 (R7) along the IGP shortest path

Fig. 4: Illustrations of Adjacency Segment. SIDs colored in red represent active SIDs. SID 24054 denotes the Adj-SID of R5, attached to the adjacency from R5 to R4. Upon receiving the active SID 24054, R5 forwards the packets towards the interface R5→R4 regardless of the shortest path.

fore the Adjacency Segment in the segment list to forward a packet to the originating node [26]. Figure 4 provides an illustration of Adjacency Segment.

The IGP link-state protocols IS-IS [29][30] and OSPF [31][32][33] have been extended to advertise the SR information of a node, such as the support of SR, local and global SIDs, supported algorithms and so forth. In addition to IGP routing protocols, the Border Gateway Protocol (BGP) is also extended to support SR, in which two important segments BGP-Prefix Segment [34] and BGP-Peer Segment [35] are introduced.

3. Segment Routing Traffic Engineering Architecture

The Segment Routing Traffic Engineering (SR-TE) architecture expresses the intent of network operator through an ordered list of segments, and programs the corresponding segment list in the network to steer traffic along the SR path [23]. Segment lists play an essential role in SR-TE. Hence, this section provides a thorough analysis of SR-TE with emphasis on two significant aspects: *segment list instantiation* and *segment list computation*. We first present an introduction of a critical framework SR Policy for segment list instantiation, then highlight several important SR-TE algorithms for segment list computation. Benefits and challenges are also summarized along with effective measures to resolve the challenges.

3.1. SR Policy

SR Policy refers to the framework that enables traffic steering on a node by instantiating the segment lists on it to implement a source routing policy associated with a specific intent [36]. SR Policy is responsible to use segment lists to represent specific path(s) in the network (i.e., *path representation*) and steer traffic to that path (i.e., *traffic steering*) [23].

3.1.1. Identification of SR Policy

An SR Policy can be uniquely identified by a tuple $< headend, color, endpoint >$ [36]:

- *Headend* indicates the node that instantiates/implements the SR Policy. Its entity depends on the scenario of SR deployment [5]. For example, it can be the source of a packet (i.e., host) or an edge router of an SR domain.
- *Color* is used to associate the policy with an intent (e.g., low delay).
- *Endpoint* represents the destination of an SR Policy.

Both *headend* and *endpoint* are specified by an IPv4/IPv6 address, whereas *color* is a 32-bit integer. For instance, operator can specify the *headend* and *endpoint* as 1.1.1.1/32 and 1.1.4.4/32, respectively, and associate the intent of low delay with *color* = 10, resulting in an SR Policy < 1.1.1.1, 10, 1.1.4.4 >. In the context of a specific headend, an SR Policy can be simply identified by $< color, endpoint >$ [36].

3.1.2. Path Representation

An SR Policy consists of one or more candidate paths. Each candidate path can be associated with one or more segment lists. The segment lists of a candidate path are allowed to consist of arbitrary types of segments. Each segment list is assigned with a weight to implement weighted load balancing. For instance, assume that a candidate path is associated with two segment lists SL_1 and SL_2 with the weights of w_1 and

w_2, respectively. When the candidate path is selected to implement the corresponding SR Policy (i.e., for traffic steering), traffic will be load balanced between SL_1 and SL_2 with the fraction of $w_1/(w_1 + w_2)$ and $w_2/(w_1 + w_2)$, respectively.

According to how segment lists are provisioned, a candidate path can be classified as explicit candidate path or dynamic candidate path. Segment lists of an explicit candidate path are provisioned and maintained by network operators or a controller. On the other hand, segment lists of a dynamic path are calculated dynamically by the headend according to the network states through a specific algorithm, which is implemented on the headend to solve an optimization problem under a set of constraints [36]. In this case, an SR-TE database is maintained at the headend to provide information (e.g., IGP link state information, TE attributes of links) for dynamic path calculation. In particular, the headend is allowed to commit the authority to compute and maintain the dynamic path to Path Computation Element (PCE).

Another type of candidate path is the composite candidate path, which acts as a container for grouping SR Policies [36]. A composite candidate path enables the combination of SR Policies, each potentially with different intents, to achieve load-balanced traffic steering over those policies.

In order to select out the best path for an SR Policy, each candidate path is assigned with a preference. The candidate path with the highest preference will be selected as active path and its segment lists will be used to implement the SR Policy. In the context of an SR Policy, a candidate path is uniquely identified by a tuple <*protocol-origin, originator, discriminator*>, which will be used for active path selection in the case of multiple candidate paths of the same preference:

- *Protocol-origin* is an 8-bit integer to indicate the means by which a headend learns the candidate path. Its value is recommended to be 10 or 20 if the candidate path is signaled via Path Computation Element Communication Protocol (PCEP) [37][38] or BGP [39], while it is recommended to be 30 if the path is originated via configuration [36], e.g., command line interface (CLI) or Network Configuration Protocol (NETCONF) [23].

- *Originator* is a 160-bit number that identifies the node providing the candidate path. It uses 32 bits to indicate the AS number of the node and 128 bits to indicate the IPv4 (using the lowest 32 bits) or IPv6 address. When the candidate path is provisioned via configuration, the AS number and IP address can be set to those of the headend or the controller/node provisioning the path [36].

- *Discriminator* is a 32-bit value used to distinguish a candidate path from multiple paths of the same protocol-origin and originator.

If there are multiple candidate paths with the same preference, the active path is selected through the following ordered rules [36]:

1. Prefer higher protocol-origin.
2. If specified by configuration, prefer the existing installed path.
3. Prefer lower originator.
4. Prefer higher discriminator.

Note that only valid paths can participate in the selection of active path. The headend must validate all candidate paths with its SR-TE database. One common criterion for path validation is the reachability of SIDs in the segment lists [36].

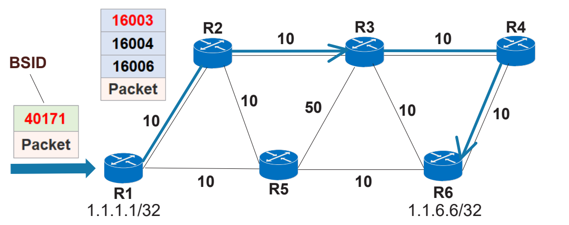
3.1.3. Traffic Steering

To implement traffic steering, segment lists of the active path for an SR Policy are installed in the Forwarding Information Base (FIB), where a Binding SID (BSID) is used as a key entry. Segment lists of the policy will be encapsulated in packets that match this entry. BSID refers to a special type of SID which binds to an SR Policy [36]. A BSID can be either local or global. If it is a local SID, a global SID (e.g., Node-SID) should be placed before it in the segment list to forward traffic to the corresponding headend. Unless otherwise specified, local BSID is considered in this paper.

A BSID is associated with a candidate path. The BSID of an SR Policy refers to the BSID associated with the active path for the policy. Although different candidate paths can be assigned with different BSIDs in an SR Policy, a common use case is to let all paths use the same one [36]. In this way, the change of active path will not affect the BSID of SR Policy, enhancing the stability of BSID [23].

Figure 5 demonstrates an example of SR Policy implementing traffic steering, where incoming packets carrying a BSID. Suppose that the AS number of the IGP domain is 5, and the IPv4 addresses of R1 and R6 are 1.1.1.1/32, 1.1.6.6/32, respectively. Assume that network operator configures an SR Policy SRP_1 on R1, where the *endpoint* is R6 and *color* $= 10$ is associated with the intent to steer traffic along the IGP shortest path to R7 while avoiding link (R3, R6). The policy SRP_1 contains three candidate paths CP_1, CP_2 and CP_3, which share the same BSID. Assume that each candidate path is associated with one segment list. Following the rules described in Section 3.1.2, CP_2 is selected as the active path for its higher preference and discriminator. Therefore, segment list $< 16003, 16004, 16006 >$ will be installed in the FIB of R1 with a BSID 40171 as the key. Hence, upon receiving a packet with active SID 40171, R1 will swap the SID with segment list $< 16003, 16004, 16006 >$, instructing the packet to traverse through the specific path.

The color and endpoint of an SR Policy can also be used to steer traffic [36]. When headend learns

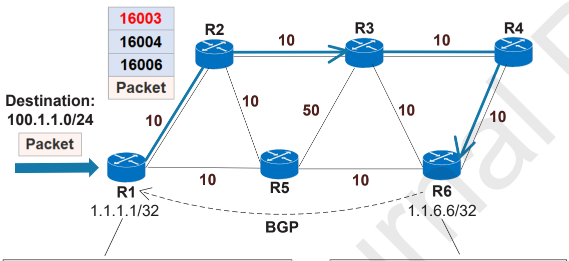


SR Policy Information on R1

```
SR Policy SRP1 <10, 1.1.6.6>
      Candidate Path CP1 <30, 5:1.1.1.1, 10>
            BSID 40171
            Preference 200
            w1   Segment List SL1 = < 16003, 24034, 16006 >
      Candidate Path CP2 <30, 5:1.1.1.1, 20> (Active Path)
            BSID 40171
            Preference 200
            w1'  Segment List SL1' = < 16003, 16004, 16006 >
      Candidate Path CP3 <30, 5:1.1.1.1, 10>
            BSID 40171
            Preference 100
            w1"  Segment List SL1" = < 16003, 24035, 16006 >
```

FIB on R1			
In	Out	Outgoing Interface	Fraction
40171	<16003, 16004, 16006>	R1→R2	100%

Fig. 5: Illustrations of SR Policy implementing traffic steering – remote steering. Packet classification is performed by a node upstream of headend and the outcome is encoded as a BSID in packet header. The number in black on the link represents the IGP link cost. Assume that the AS number of the IGP domain is 5.



SR Policy SRP2 on Headend R1:
Color: 20
Endpoint: 1.1.6.6/32
BSID: 40172
Segment List: <16003, 16004, 16006>

BGP Route:
Prefix: 100.1.1.0/24
Extended-community Color: 20
Next-hop: 1.1.6.6/32

FIB on R1	
SR Policy SRP2 In: 40172 Out: <16003, 16004, 16006>	**BGP** In: 100.1.1.0/24 Out: via 40172

Fig. 6: Illustrations of SR Policy implementing traffic steering – local steering, where packet classification is performed on the headend. The number in black on the link represents the IGP link cost.

a BGP route with extended-community color [40] and next-hop that match the color and endpoint of an SR Policy, it installs the BGP route resolving on the BSID of the policy [23]. Figure 6 provides an illustration. Suppose that a BGP route <*prefix*: 100.1.1.0/24, *next-hop*: 1.1.6.6/32> on R6 is attached to the *extended-community color* 20, which associates with the intent to forward the packets towards destination 100.1.1.0/24 along the IGP shortest path from R1 to R6 while avoiding link (R3, R6). R1 learns such route via BGP session with R6 and resolves it on the BSID 40172 of the SR Policy SRP_2. Hence, R1 will push the segment list of SRP_2 in the header upon receiving the packets towards destination 100.1.1.0/24.

3.2. SR-TE Algorithms

Segment lists of SR-TE are obtained through a series of SR-TE algorithms. Each algorithm is usually associated with an intent, which describes an objective to optimize a metric under a number of constraints when computing segment lists [41]. The computation of segment lists involves calculating the optimized paths and encoding them as a set of segment lists.

Two important types of SR-TE algorithms have been proposed from industry, i.e., *Flexible Algorithm* and *SR-native algorithm*, which are introduced in documents of both Internet Engineering Task Force (IETF) [42][43] and Cisco [23]. *Flexible Algorithm* enriches segments used to encode segment lists, while *SR-native algorithm* refers to the segment list computation algorithm that fully exploits the native of SR (e.g., ECMP support, shortest-path hops) [23].

3.2.1. Flexible Algorithm

Flexible Algorithm (Flex-Algorithm) is a type of algorithm associated with Prefix Segment. Flex-Algorithm can be customized by network operators in the IGP domain, and is assigned with a numeric identifier within [128, 255][3]. It enables a router to compute the shortest paths using a specific metric, even with constraints to exclude a number of link resources. To avoid looping, all routers participating in the same Flex-Algorithm must have a common knowledge of its definition. The definition of a Flex-Algorithm is composed of a *calculation-type*, a *metric-type* and a number of *constraints* [42]:

- *Calculation-type* specifies the type of path calculation, e.g., SPF or strict SPF [23]. If *calculation-type* is strict SPF, routers must override the local policies that may change the forwarding path.
- *Metric-type* specifies the type of metric to optimize during path calculation, which can be IGP metric, TE metric [44][45] or link delay [46][47].
- *Constraints* specify the particular extended administrative groups (commonly known as "link colors") [48] or Share Risk Link Groups (SRLGs) to exclude a set of links from path calculation.

Figure 7 illustrates the usage of Flex-Algorithm. Suppose that network operator customizes a Flex-Algorithm 128 <*calculation-type*: SPF, *metric-type*: link delay, *constraints*: exclude links of color *RED*>. All SR nodes are configured to support this algorithm and thus are capable of calculating shortest paths to each other consistently. Suppose that network operator would like to forward traffic from R1 to R7

[3]SR allows to associate the Prefix Segment with an algorithm represented by a numeric value in the range of [0, 255], as described in Section 2.2.

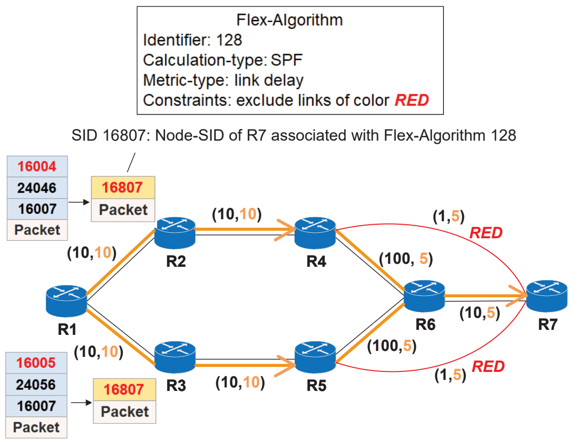


Fig. 7: Illustrations of Flex-Algorithm. The "link colors" are represented as names of colors for ease of understanding, which in practice are numeric values. The vector (x, y) along each link represents the link cost and link delay, respectively.

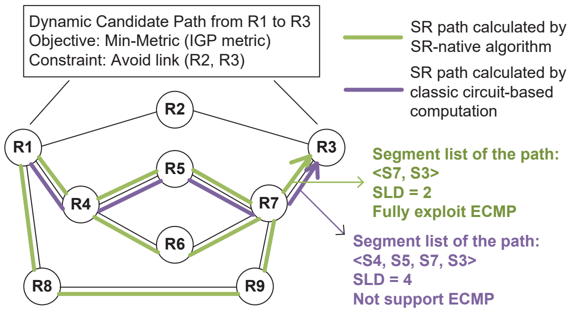

Dynamic Candidate Path from R1 to R3
Objective: Min-Metric (IGP metric)
Constraint: Avoid link (R2, R3)
SR path calculated by SR-native algorithm
SR path calculated by classic circuit-based computation
R1
R2
R3
R4
R5
R6
R7
R8
R9
Segment list of the path:
<S7, S3>
SLD = 2
Fully exploit ECMP
Segment list of the path:
<S4, S5, S7, S3>
SLD = 4
Not support ECMP

Fig. 8: Illustrations for describing SR-native algorithm [23]. SIDs S_3, S_4, S_5, S_7 represent the Prefix-SIDs of R3, R4, R5, R7, respectively.

along the low delay path without crossing the *RED* links. In this case, if leveraging Flex-Algorithm, R1 can encapsulate the packets with only one segment $< 16807 >$ and forward the packets along the low delay ECMP-aware paths. The SID 16807 represents the Node-SID of R7 associated with Flex-Algorithm 128. On the other hand, without Flex-Algorithm, R1 needs to use 2 segment lists $< 16004, 24046, 16007 >$ and $< 16005, 24056, 16007 >$ to implement low delay ECMP, each with 3 segments to achieve the intent, thus entailing more overhead.

As depicted in Figure 7, Flex-Algorithm extends the segments used to encode segment lists, which provides several key benefits in segment list encoding such as reducing segment overhead and enhancing ECMP support. Hence, many SR-native algorithms or intent-based SR-TE solutions (e.g., [49][50]) can leverage Flex-Algorithm Prefix Segment to optimize the encoding of segment lists.

3.2.2. SR-native Algorithm

SR-native algorithm, as described in [43], refers to the algorithm implemented on a headend or PCE for dynamic candidate path calculation. A dynamic path is specified as an objective and a set of constraints [43], which altogether describe an intent for an SR-native algorithm to optimize. Generally, two types of objectives are proposed for SR-native algorithms [43]:

- *Min-Metric*: the SR path computed by an SR-native algorithm should minimize a specific metric such as IGP metric, TE metric, or link delay.
- *Min-Metric with margin and maximal number of SIDs*: a modified version of *Min-Metric* with two changes: a margin of by which two paths with similar metrics would be considered equal, and a constraint on the maximum number of SIDs in the segment list.

Numerous constraints are also described in [43], which can be categorized into three types:

- Exclusion and/or inclusion of a set of network elements (e.g., SRLGs, IP addresses).
- Upper bounds for specific parameters (e.g., cumulative metric, number of SIDs in the segment list).
- Difference in terms of links, nodes, SRLGs, etc., from another service instance (e.g., SR path originating from another headend).

In addition to optimizing a specific intent, an SR-native algorithm also aims to maximize the use of all ECMP branches and minimize the SLD of the segment list [23]. Figure 8 illustrates the characteristics of SR-native algorithm. Assume that a dynamic path from R1 to R3 is specified as <*objective*: *Min-Metric* (IGP metric), *constraint*: avoid link (R2, R3)>. The classic circuit-based computation will calculate the shortest paths after pruning the link (R2, R3) from the graph, and select a single non-ECMP path from them, then encode it as the segment list $< S_4, S_5, S_7, S_3 >$. By contrast, an SR-native algorithm will seek an SR path that maximizes the use of ECMP branches and minimizes SLD, resulting in the segment list $< S_7, S_3 >$. Obviously, SR-native algorithm fully exploits ECMP and reduces segment overhead with respect to classic circuit-based computation.

3.3. Benefits and Challenges

After reviewing the architectural design of SR-TE, we summarize the key benefits of SR-TE as follows:

- *Scalability:* Due to the source routing mechanism, per-flow states are eliminated from intermediate nodes, reducing significant overhead and increasing the scalability of SR-TE.
- *Simplicity:* SR-TE does not require much intelligence on intermediate nodes, reducing the hardware requirements on them. Besides, the configurations of SR Policies or segment lists are only performed on headends, which simplifies and fastens the implementation of an SR-TE solution.

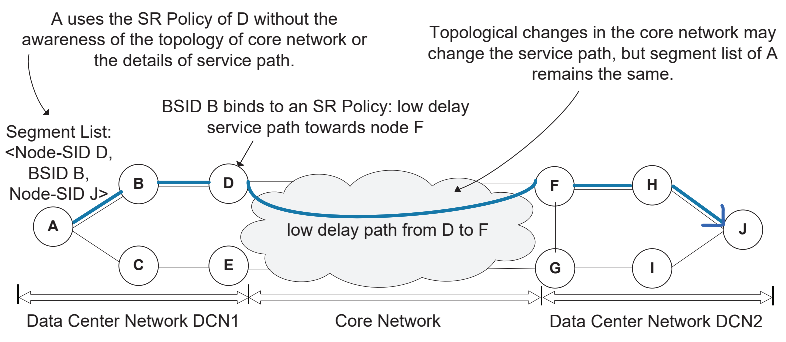

A uses the SR Policy of D without the awareness of the topology of core network or the details of service path.
Topological changes in the core network may change the service path, but segment list of A remains the same.
BSID B binds to an SR Policy: low delay service path towards node F
Segment List: <Node-SID D, BSID B, Node-SID J>
A
B
C
D
E
F
G
H
I
J
low delay path from D to F
Data Center Network DCN1
Core Network
Data Center Network DCN2

Fig. 9: Illustrations of BSID to provide network opacity and service independence, as well as isolate one domain from topological changes of another domain [23]. Suppose that all candidate paths of the SR Policy providing the low delay path from D to F use the same BSID *B*. Hence, changes of active path do not affect the BSID of the policy.

- *Flexibility:* Since a segment list allows the arbitrary use of segments, any path in the network can be expressed as a segment list composed of Prefix Segments and Adjacency Segments. Moreover, SR-TE allows operators to associate customized Flex-Algorithm with Prefix Segments, further enhancing the flexibility in path representation.

- *Robust Traffic Steering:* An explicit or dynamic candidate path can be associated with more than one segment lists, which allows SR-TE to implement weighted ECMP. In addition, a composite candidate path can be used for grouping SR Policies and thus, enables SR-TE to load balance traffic over different segment lists with various intents.

- *Multi-domain TE:* Leveraging segment lists, SR-TE provides the scalable concatenation of paths across different domains [51]. SR-TE overcomes the interoperability issues as it requires no signaling across multiple domains after the calculation of a new path (i.e., segment list) [51]. Furthermore, BSID enables SR-TE to provide network opacity and service independence, as well as isolate one domain from the topological changes of another domain [43] (See Figure 9).

- *Various Deployment Manners:* SR-TE can be deployed in various manners: distributed, centralized or hybrid [43]. In the case of distributed deployment, routers dynamically calculate the segment lists with specific algorithms. In a centralized environment, the controller takes control of all the calculation of segment lists. As for hybrid manner, routers preserve the capability of calculating segment lists and they are also allowed to request a PCE to assist their calculation.

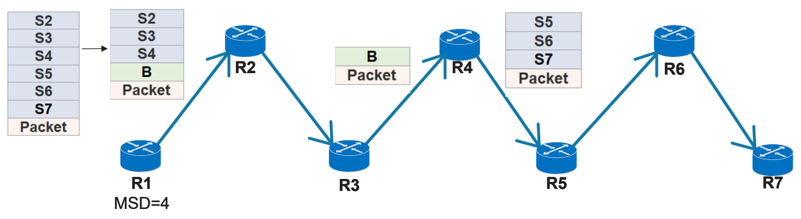
Yet despite much promise, SR-TE still faces two major challenges:

- *Overhead:* The main challenge of SR-TE is the extra overhead caused by inserting segments in each packet. SR-TE relies on segments to manipulate traffic, thereby inevitably occupying extra network resources (e.g., bandwidth) and entailing networking overhead.

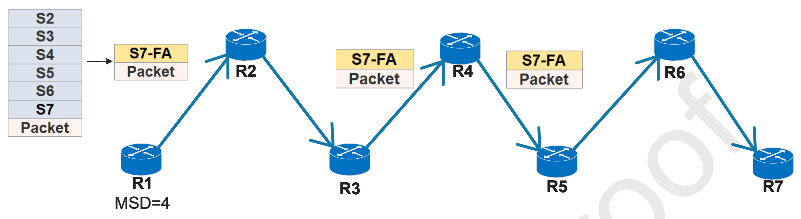
- *Maximum SID Depth (MSD) limitation:* Existing network devices only support limited MSD [52]. In other words, the segment list depth (SLD) allowing a headend to push in a packet should not exceed a value due to the hardware limitation. This restricts the number of segments to use and may decrease the effectiveness of SR-TE.

Two approaches are available to effectively address the challenges mentioned above: one is to exploit BSID, and the other is to associate Flex-Algorithm with Prefix Segment. Figure 10 demonstrates these two approaches. Assume that the MSD supported by ingress node R1 is 4 and R1 needs to forward packets along the low delay path to destination R7. With Prefix-SID only, R1 needs to use the segment list $< S_2, S_3, S_4, S_5, S_6, S_7 >$. The depth of this segment list is 6, which violates the MSD limitation and thereby can not be used to achieve the intent. One solution is to bind a BSID B to an SR Policy $< S_5, S_6, S_7 >$ on R4, resulting in the segment list $< S_2, S_3, S_4, B >$. This reduces the SLD on R1 to 4. Another solution is to associate a Flex-Algorithm FA with Prefix Segment in the SR domain. Suppose that Flex-Algorithm FA enables routers to compute the shortest paths with delay metric, and network operator configures a SID S_7–FA associated with algorithm FA on R7. In this way, the segment list becomes $< S_7$–$FA >$, reducing the SLD on R1 to 1.

In fact, previous studies have shown that only a small number of segments (less than 5) will be sufficient to implement TE in most cases [23], and network devices produced by major equipment manufacturers are capable to push up to 5 segments [52]. Hence, SR-TE generally will not cause too much overhead or violate the MSD limitation.



(a) SID B represents the BSID binding to $< S_5, S_6, S_7 >$ on R4. With BSID, R1 uses the segment list $< S_2, S_3, S_4, B >$ to achieve the intent, which reduces the SLD on R1 from 6 to 4.



(b) SID S_7–FA represents the Prefix-SID of R7 associated with Flex-Algorithm FA that enables routers to compute shortest paths with delay metric. With Flex-Algorithm, R1 uses the segment list $< S_7$–$FA >$ to achieve the intent, reducing the SLD on R1 from 6 to 1.

Fig. 10: Illustrations of BSID and Flex-Algorithm to reduce segments in the packet. SIDs S_2, S_3, S_4, S_5, S_6, S_7 represent the Prefix-SID of R2, R3, R4, R5, R6, R7, respectively. Segment list $< S_2, S_3, S_4, S_5, S_6, S_7 >$ violates the MSD limitation of R1 and thus is not available for implementing the low delay intent of operator.

4. Intent-based SR-TE Solutions

In this section, we conduct a survey on the state-of-the-art SR-TE solutions extensively studied in academia. Previous TE solutions aim for appropriate settings of IGP link weights (e.g., [7][8]), establishment of constrained MPLS tunnels (e.g., [10][12]) or installation of flow table entries on switches (e.g., [16][17]). By contrast, SR-TE solutions intend to efficiently compute a set of segment lists that can be programmed as SR Policies to implement the TE strategies. Unlike other routing schemes intended for wireless networks (e.g., secure QoS-aware routing schemes [53][54]), SR-TE solutions mainly apply in data center networks, backbone networks, WANs, etc.

Each of the following SR-TE solutions is designed with an intent expressed as an optimization objective with constraints. Thus, for simplicity and clear understanding, we categorize these solutions into four types based on the objectives of their intents[4]:

1. *Minimizing maximum link utilization*: The optimization of this objective involves an efficient flow allocation to balance the load of traffic across the network.
2. *Maximizing network throughput*: The optimization of this objective involves full utilize of link residual bandwidth to accommodate more service demands.
3. *Maximizing quality of experience (QoE)*: QoE is a parameter that measures the end-users' satisfaction of an application or service, influenced by various factors [55]. Therefore, this objective requires an optimization of a series of related network metrics, e.g., bandwidth, delay, and jitter [56].
4. *Minimizing energy consumption*: This objective can be achieved by reducing the amount of active network infrastructures. SR-TE is responsible to reroute traffic on the available resources.

Table 2 compares these four types of SR-TE solutions from different aspects. Note that the SR-TE solutions in Table 2 can exploit the Flex-Algorithm Prefix Segments for more efficient segment list encoding. Besides, some SR-TE solutions can be identified as SR-native algorithms, e.g., [49][57][58]. However, many of them are designed for applications or networks of specific properties and requirements, which may not be consistent with the characteristics of SR-native algorithm. For instance, an SR-TE solution proposed in [59] intends to find an SR path that will not cause any flow splitting, considering that routers may have limitation on splitting traffic (e.g., routers running JUNOS [60]).

[4]In this paper, we adopt a more flexible concept of intent compared to definition in [43].

Tab. 2: Comparison of SR-TE solutions of the four objectives.

Objective	Networks	Significance	Challenges	Ref.
Minimizing maximum link utilization	Mostly carrier networks [1]	(1) Avoid bottlenecks of network performance [49]. (2) Control congestion to ensure smooth network operations [61].	(1) Development of algorithms or models of scalability and computational efficiency [62][63]. (2) Satisfaction of constraints imposed from hardware limitation or practical requirements [57]. (3) Exploration on tradeoffs between benefits and costs from the maximum SLD allowed for each segment list [61].	[25][49] [57][58] [59][61] [62][63] [64][65] [66][67] [68]
Maximizing network throughput	Mostly data centers and WANs [1]	(1) Fully optimize bandwidth utilization [61]. (2) Accommodate more service demands without increasing costs [69].	(1) Calculation of SR paths of sufficient bandwidth [58]. (2) Reduction of the wastage of bandwidth [70]. (3) Prevention of the exhaustion of bandwidth to guarantee a consistently high level of network throughput [71].	[58][61] [70][71] [72][73] [74][75]
Maximizing QoE	Networks providing delivery and/or interactive applications or services [55]	Improve the experience and satisfaction of end-users [76].	(1) Identification of network metrics affecting specific types of applications [77]. (2) Allocation of SR paths that optimize specific network metrics to achieve the improvement of end-users' QoE [76].	[50][76] [77][78] [79][80]
Minimizing energy consumption	Backbone networks [81] and data centers [82]	Reduce the wastage of energy consumed by idle network infrastructures [83].	(1) Identification of network infrastructures to switch off and how to switch them back on [81][82]. (2) Calculation of SR paths that enables load balancing across the available network resources [81][82].	[81][82] [83]

4.1. *Minimizing Maximum Link Utilization*

Maximum link utilization (MLU) is a metric widely exploited to measure network congestion. Lower MLU usually indicates lower congestion, thus contributing to the smooth operations of networks. The minimization of MLU can be achieved via different optimization approaches. Table 3 summarizes the SR-TE solutions in this section.

4.1.1. *Linear Programming Based Optimization*

Linear Programming (LP) is a mathematical model widely exploited to solve optimization problems, including minimizing MLU in routing optimization. According to the assumptions on routing types, the LP based SR-TE solutions can be further categorized into *traffic aware routing* and *traffic oblivious routing*. The main difference between these two types of routing falls in the reliance of traffic matrix used for predictions of future traffic patterns. Traffic aware routing models rely on a specific traffic matrix for optimization, while traffic oblivious routing models optimize routing independently of traffic patterns.

Traffic Aware Routing

Traffic aware routing requires the knowledge of a traffic matrix to forecast the traffic distribution and bandwidth requests. It optimizes routing according to such matrix, and hence is capable of achieving near-optimal performance when traffic pattern is stable and similar to the optimized matrix. However, uncertain performance may happen with rapid and unpredictable traffic fluctuations [84].

Many LP models for traffic aware routing are proposed to minimize MLU under different constraints, e.g., SLD constraint, single-path constraint and candidate node constraint. Each constraint makes different contributions to the proposed SR-TE solutions, as shown in Table 4.

The SLD constraint imposes limitation on the length of each computed segment list, preventing the

Tab. 3: Summaries of SR-TE solutions for minimizing maximum link utilization.

Ref.	Optimization Approach	Summary
[25]	Linear Programming	Failure resilient model that minimizes MLU for predefined failure scenarios while satisfying practical requirements.
[49]	Hybrid Constraint Programming	Programmable and interactive optimizer with a built-in robust algorithm to compute segment lists.
[57]	Hybrid Constraint Programming	Scalable and flexible algorithm to accommodate practical requirements for segment list computation.
[58]	Linear Programming	Traffic oblivious routing model to provide solutions applicable for a broad range of traffic matrices.
[59]	Linear Programming	Operator-friendly model that minimizes the number of SR paths to simplify manual configurations while considering the characteristics of ISP networks.
[61]	Linear Programming	Computationally easy model that selects transit SR nodes from a limited set of critical candidate nodes.
[62]	Local Search	Extremely fast algorithm to control congestion caused by unexpected network events.
[63]	Column Generation	Scalable algorithm that exploits the K-segment scheme and Adjacency Segments for segment list computation.
[64]	Linear Programming	Traffic oblivious routing model that derives an estimated bound of traffic matrices for routing optimization.
[65]	Linear Programming	Effective model that exploits the K-segment scheme and benefits of Adjacency Segments for minimizing MLU.
[66]	Linear Programming	Hardware-friendly model that provides SR paths without violating traffic splitting limitation of network equipment.
[67]	Linear Programming	Hardware-friendly model that respects MSD and traffic splitting limitation of network devices.
	Local Search	Two-stage algorithm that initially allocates SR paths to reduce packet overhead, then re-optimizes the solution to further minimize MLU.
[68]	Linear Programming	Computationally easy model that selects one transit node for an SR path from the candidate nodes topologically near the source and destination.

Tab. 4: Constraints of the LP models for traffic aware routing for minimizing MLU and their contributions.

Constraints	Contributions	Ref.
SLD constraint	Respect hardware limitation	[25][59][61] [65][67][68]
Single-path constraint	Respect hardware limitation; Avoid the cost of splitting traffic	[25][59] [66][67]
Candidate node constraint	Reduce computational complexity	[25][59] [61][68]

models from violating the MSD limitation of network devices. In particular, Trimponias et al. [61] discover from experimental analysis that a list with 2 segments will be sufficient to achieve good enough performance for minimizing MLU, while allowing more segments will not produce significant improvements but increase the computational complexity. Hence, an LP model is proposed to exploit the 2-segment scheme which restricts the SLD of each segment list to 2. The same scheme is adopted by the LP models in [25][59][68].

Li et al. [65] present a Mixed Integer LP (MILP) model which exploits the K-segment scheme to guarantee the optimal solutions, where K can be customized. Due to the high complexity, a simplified model called K-sMILP is designed. Moreno et al. [67] also present an Integer LP (ILP) model for K-segment scheme, which, however, only serves as benchmarks for another proposed heuristic algorithm (see Section 4.1.2) in experiments due to its high complexity.

Existing routers have limitation on splitting traffic. For example, routers running JUNOS support splitting flows into a limited number of equal parts, such as 16, 32 or 64 [60]. In addition, traffic splitting triggered by ECMP could lead to undesired consequences, such as unbalanced traffic distribution or inadequate splitting [67]. To this end, LP based solutions in [25][59][66][67] introduce the single-path constraint, which enforces the computation of segment lists such that each flow will traverse a single path without splitting. This can be achieved by adapting the domain value of traffic splitting variables in the LP formulation to be binary. For instance, let f_{ij}^{sd} represents the proportion of the flow from ingress s to egress d on link (i, j). Such variable can be set to be floating point variable subject to $0 \leq f_{ij}^{sd} \leq 1$, if arbitrary flow splitting is allowed. Instead, to prohibit any splitting, i.e., compute single paths for each flow, f_{ij}^{sd} can be binary variable subject to $f_{ij}^{sd} \in \{0, 1\}$.

Taking all SR nodes in the SR domain as candidate transit nodes may lead to expensive computation, thereby decreasing the scalability of the algorithms [61]. Hence, by introducing the candidate node constraint, i.e., restricting the node set for transit node selection to a smaller one, the computational efficiency of the algorithms will be improved. Methodologies on candiate node generation have been studied. The works in [25][61][68] propose different methods to compute a small but representative set of candidate nodes, which is introduced as a constraint in the LP formulation. These methods will be discussed further in Section 5.2. The models in [25][59] explicitly specify a black list of nodes that should be excluded from transit node selection. Such a black list also respects the characteristics of ISP networks, as there are some nodes (e.g., edge routers), by designed, only allowed to be source or destination [59].

In addition to above constraints, some LP models are designed with other optimization aspects. For example, Li et al. [65] explore the benefits of Adjacency Segment in the LP model, which allows to consider more SR paths for minimizing MLU. The LP models in [25][59] also minimizes the number of computed SR paths to simplify network management, since each must be manually configured in the network environment without a centralized controller to enable automation. Besides, the LP model of Cianfrani et al. [66] solves the problem of SR node deployment to minimize MLU in addition to segment list computation (see Section 5.1).

The above LP models for traffic aware routing efficiently minimizes MLU under various constraints. In particular, the existence of a traffic matrix enables each model to achieve significant performance in general cases, where traffic shares similar pattern to the matrix. However, performance degradation may happen when real traffic deviates substantially from the optimized matrix [84].

Traffic Oblivious Routing

Traffic oblivious routing can efficiently resolve the challenge of traffic uncertainty, which computes segment lists with fairly limited knowledge of the traffic matrix [85]. This type of SR-TE solutions is capable of providing SR paths that work well for a wide range of traffic matrices. However, their potential drawback falls in the sub-optimal performance when handing normal traffic, which may exist in the network for a long period of time [86].

Traffic oblivious routing for minimizing MLU can be modeled as LP [58][64]. One approach to solve the problem is to consider all possible traffic matrices in the LP formulation, as demonstrated by Bhatia et al. [58]. At first, they propose an LP formulation for traffic aware routing for minimizing MLU. Based on this formulation, a game theoretic analysis is provided to derive the LP model for oblivious routing, which computes the solution of segment lists applicable to as many traffic matrices as possible. Since two segments are sufficient to achieve significant performance, the 2-segment routing scheme is exploited for segment list computation.

As traffic shares similar distribution over time, and recently observed traffic distribution would hold in the future with high probability [87], another approach is to utilize an estimated range of traffic matrices for oblivious routing. For example, Roomi et al. [64] propose an LP model that optimizes routing for an estimated range of matrices. To derive the lower and upper bound of traffic matrices, authors assume that an inital estimation of traffic matrix exists. Besides, another LP model is designed to find the traffic matrix that causes the utilization of a link to be maximal. Such model is applied to each link in the network to obtain a set of traffic matrices, from which the estimated range can be derived. The estimated range is then introduced as a constraint in the LP model for minimizing MLU, which exploits 3-segment scheme to guarantee the optimal SR paths.

The aforementioned models both have advantages and disadvantages. Bhatia's model [58] takes all the traffic matrices into account, which broadens the range of matrices that the model can cope with, but sacrifices performance for normal traffic. The 2-segment scheme simplifies the complexity of the model but may not be sufficient to achieve optimal solutions. On the other hand, Roomi's model [64] utilizes an estimated bound of matrices and exploits the 3-segment scheme for oblivious routing, which improves performance especially for the matrices in the estimated range. However, since an LP model is implemented on each link to derive the estimated range, scalability issue may arise notably in large scale networks with hundreds or thousands of links. Besides, with respect to 2-segment scheme, the 3-segment scheme increases the computational complexity and induces more segment overhead.

4.1.2. Local Search Based Optimization

Local Search (LS) is a heuristic algorithm to solve the optimization problems. The basic idea of LS is to take an initial solution as input and iteratively go from the solution to another by applying local changes called moves on the current solution, and select a new one from the neighborhood, i.e., the set of solutions generated by each move. The iteration of LS will stop until a criterion is met, such as a runtime limit or that the solution is qualified. Due to its nature, LS can be exploited for extremely fast optimization for MLU in

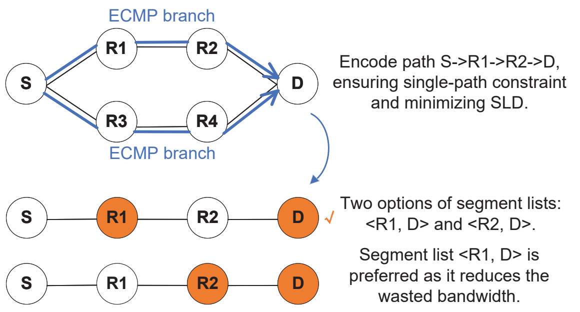

ECMP branch
S
R1
R2
R3
R4
D
ECMP branch
Encode path S->R1->R2->D, ensuring single-path constraint and minimizing SLD.
Two options of segment lists: <R1, D> and <R2, D>.
Segment list <R1, D> is preferred as it reduces the wasted bandwidth.

Fig. 11: Illustrations for explaining the encoding process of algorithm in [67]. There are two options for encoding path S→R1→R2→D, both subject to single-path constraint and minimizing SLD. If using $< R_2, D >$, SID R_2 will traverse through the subpath R1→R2, wasting more bandwidth. Hence, segment list $< R_1, D >$ is preferred.

SR-TE, as demonstrated in [62][67].

Unexpected events may arouse traffic surges in the network, which may lead to serious consequences on link utilization (e.g., congestion). In order to quickly respond to such events to ensure the smooth operations of networks, Gay et al. [62] propose an LS based SR-TE solution for minimizing MLU. In particular, moves of LS are designed to explore the neighborhood more efficiently and those with no contribution on decreasing MLU are eliminated. Moreover, to simplify the exploration of neighborhood, a stochastic heuristic algorithm is utilized to detect the parts of neighborhood that are most likely to improve the current solution. The intuition behind this algorithm is that changing the path using one of the maximal loaded links seems to result in a better solution. Experimental results indicate that the proposed solution achieves high performance in three scenarios: (i) minimize MLU under strict time constraint; (ii) minimize the computation time spent on decreasing MLU under a given threshold; (iii) quickly control congestion in 1 second when sudden traffic changes occur, even in large scale networks.

Moreno et al. [67] propose an LS based heuristic solution for MLU minimization. In the first stage, a heuristic algorithm encodes a set of paths between all source destination pairs as segment lists, with the single-path constraint and the objective to minimize packet overhead (see Figure 11). It also allocates SR paths to flows and returns the solution to LS algorithm for re-optimization. The LS algorithm first sorts the links in descending order in the light of their utilization. Afterwards, for each link, it seeks an alternative SR path that would decrease the utilization of the link, then replaces the old path with such one. Finally, it updates the link utilization, resorts the links and goes to the next iteration. Evaluation results show that the proposal efficiently minimizes MLU and the LS algorithm can re-optimize the solution from the heuristic algorithm in sub-second time.

The above solutions demonstrate that LS can be utilized as a fast optimization approach to minimize MLU under strict time constraint [62], or participate in the optimization process by quickly re-optimizing the solutions obtained from previous steps [67]. However, the challenge of LS falls in the possibility of sub-optimal performance as the global optimum may not be included in the exploring neighborhood.

4.1.3. Hybrid Constraint Programming Based Optimization

Hybrid Constraint Programming optimization approach combines both Constraint Programming (CP) [88] and Large Neighborhood Search (LNS) [89]. In CP, a set of variables each with finite domain of values is defined, and a set of constraints is also applied to those variables. Constraints are implemented through specific algorithms to preserve the consistency between variable domains, enabling CP to support independent and easily composable constraints [49]. Similar to Local Search, LNS is a scalable heuristic algorithm that iteratively improves the current solution by applying changes on it. In particular, the neighborhood of LNS is huge due to the large changes, which allows LNS to avoid or escape from local optima. Hybrid CP efficiently combines the strengths of two techniques. Therefore, it has been exploited to scalably compute SR paths for minimizing MLU under multiple constraints of practical significance.

Hartert et al. [57] propose a hybrid CP algorithm to minimize MLU while satisfying requirements of network operators. A scalable path representation *SR-path* variable is proposed, which corresponds to a sequence of Direct Acyclic Graphs (DAGs) that a flow will traverse through. Considering the requirements of network operators, a set of constraints are applied on the *SR-path* variables (see Figure 12). A hybrid CP algorithm is implemented to assign *SR-path* variables to each demand. At each iteration, the LNS algorithm sorts the set of demands which are routed over the most loaded links in non-increasing order according to their bandwidth requirements, followed by a randomized approach to select no more than k demands. For each selected demand, the CP algorithm heuristically searches for the *SR-path* variables that would significantly decrease MLU without violating the specific constraints.

As an optimizer implemented in the three-layer network architecture to simplify network management (see Figure 13), DEFO [49] adopts a similar hybrid CP algorithm for SR path computation. In particular, it exposes a high-level interface for network operators to directly declare their objectives (e.g., minimize MLU) and constraints (e.g., maximal delay experienced by specific demands), which will be optimized by the bulit-in hybrid CP algorithm.

Experimental analysis from [49][57] have demonstrated the efficiency of hybrid CP on minimizing

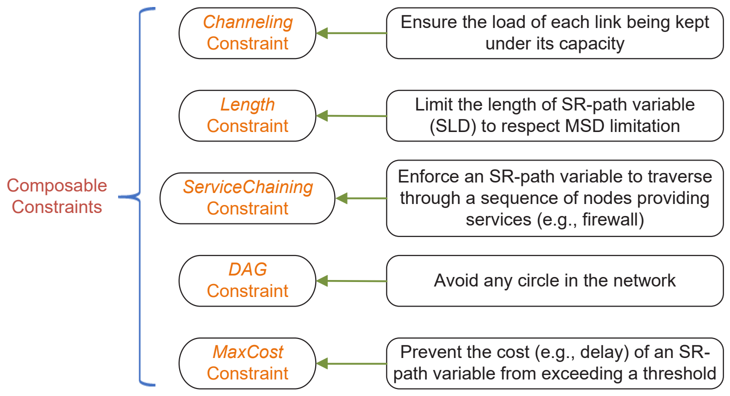

Composable Constraints
Channeling Constraint
Ensure the load of each link being kept under its capacity
Length Constraint
Limit the length of SR-path variable (SLD) to respect MSD limitation
ServiceChaining Constraint
Enforce an SR-path variable to traverse through a sequence of nodes providing services (e.g., firewall)
DAG Constraint
Avoid any circle in the network
MaxCost Constraint
Prevent the cost (e.g., delay) of an SR-path variable from exceeding a threshold

Fig. 12: Constraints associated with practical requirements proposed in [57].

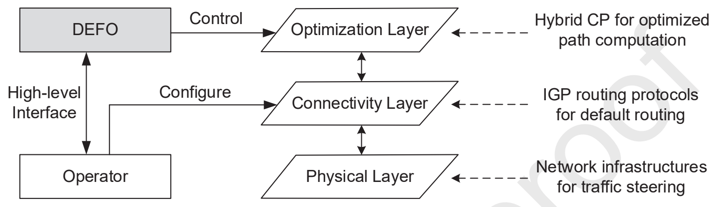

DEFO
Control
Optimization Layer
Hybrid CP for optimized path computation
High-level Interface
Configure
Connectivity Layer
IGP routing protocols for default routing
Operator
Physical Layer
Network infrastructures for traffic steering

Fig. 13: Three-layer network architecture of DEFO [49].

MLU under a set of constraints of practical significance. In fact, these constraints may greatly increase the computational complexity. It has been proved that the optimal solution of SR paths is NP-hard even if only the *channeling* constraint is respected [57].

4.1.4. Column Generation Based Optimization

Column Generation (CG) [90] defines the master problem to be the original Linear Programming problem with limited variables (i.e., an initial set of columns). It leverages the sub-problem to discover new variables (i.e., new columns) with reduced cost to the objective function. These variables tend to improve the current solutions. Hence, CG is rather efficient in solving the large linear programs, as it focuses on the sub-sets of variables that potentially contribute to the objective, instead of explicitly considering them all.

CG can be used to minimize MLU with good scalability. For example, Jadin et al. [63] propose a CG based algorithm Column Generation for Segment Rotuing (CG4SR), which exploits the K-segment scheme and supports Adjacency Segment. CG4SR first implements the CG algorithm to deal with routing optimization, where columns correspond to candidate SR paths. The master problem of CG is formulated as Linear Programming, which will maximize the volume of demands routed on the SR paths without exceeding a given threshold λ for link utilization. The reason for such definition is that directly minimizing MLU will result in the difficulty for the algorithm to converge, thus decreasing its scalability. In addition, the sub-problem is defined to seek the min-cost SR paths under the SLD constraint for each demand, which can be solved in polynomial time using a proposed dynamic programming algorithm. Note that some demands would be rejected if the value of λ is too tight. Therefore, CG4SR implements a binary search approach to determine the best value of λ, so that all demands can be routed. At last, for the SR paths returned from previous steps, CG4SR solves an Integer Linear Programming problem to seek an optimal one for each demand so that MLU can be minimized. Experimental analysis indicates that CG4SR is able to obtain near optimal solutions for MLU minimization in a short time.

In addition to scalability, CG is also flexible to deal with the optimization problem of MLU under different constraints. For instance, CG4SR [63] allows to accommodate other constraints like maximal delay of the computed SR paths. This can be achieved by only adapting the dynamic programming algorithm for solving the sub-problem.

4.2. Maximizing Network Throughput

Network throughput is a metric that measures the amount of datagrams passing through the network in a period of time. Higher throughput usually indicates that more bandwidth resources are utilized and more service demands are served, thus leading to a better network performance. The throughput maximization can be performed in online or offline. The main difference between online and offline routing is the requirement of a monitoring mechanism (e.g., a centralized controller). Table 5 provides summaries of the SR-TE solutions in this section.

4.2.1. Online Routing

Online routing dynamically computes the optimized SR paths for each incoming demand mainly

Tab. 5: Summaries of SR-TE solutions for maximizing network throughput.

Ref.	Routing Type	Summary
[58]	Online Routing	Exponentially set link weights according to link utilization for SR path computation, so as to fully utilize the available bandwidth.
[61]	Offline Routing	Solve a multicommodity flow like problem to acquire the solutions of 2-segment paths.
[70]	Offline Routing	Optimize routing for hybrid SR networks with a depth first search to find feasible paths and a heuristic algorithm for path allocation.
[71]	Online Routing	Leverage two functions to calculate link weights and node weights to construct multicast trees.
[72]	Online Routing	Compute min-weight SR paths with a parameter to make tradeoffs between routing performance and resource consumption.
	Offline Routing	Associate link length variables with links in the network to construct auxiliary graphs for min-weight SR path computation.
[73]	Offline Routing	Unequally split network traffic to utilize the available bandwidth resources.
[74]	Online Routing	Propose two metrics for link weight assignment to find min-weight SR paths, enabling load balancing across the network.
[75]	Online Routing	Combine SR-TE and SDN-based MPTCP for online establishment of MPTCP connections.

Tab. 6: Comparison of online SR-TE solutions for maximizing network throughput.

Ref.	Monitoring Technique	Monitoring Information
[58]	PCE	Link bandwidth, flow request
[71]	SDN controller	Link bandwidth, switch flow entry, multicast request
[72]	SDN controller	Link bandiwidth, flow request
[74]	SDN controller	Link bandwidth, traffic distribution, flow request
[75]	SDN controller	Link bandwidth, link delay, MPTCP request

based on the information of current network states (e.g., link residual bandwidth). Hence, an online monitoring mechanism for network information is necessary. One advantage of online routing is the potential to cope with dynamic network changes, as routing adjustment can be made promptly when traffic changes are detected.

Table 6 lists several SR-TE solutions for throughput maximization based on online routing. These solutions rely on a centralized controller to monitor the network for incoming requests and network states for path computation. For example, Bhatia et al. [58] propose an online routing algorithm that leverages a PCE to monitor flow requests and link bandwidth. For each incoming flow, links are weighted and a 2-segment min-weight path is computed. In particular, link weights are set exponentially with link utilization. Such setting prioritizes links with more residual bandwidth and prevents links from being easily overloaded. A similar setting on link weights is adopted in [72]. In addition, Zhang et al. [72] introduce a parameter in the update of link weights to make tradeoffs between routing performance and bandwidth consumption - a smaller parameter leads to a higher network throughput, but also a higher risk for link congestion because of more occupied bandwidth.

In light of link weight settings for SR path computation, Lee et al. [74] propose two important metrics: *link congestion* that grows rapidly with link utilization, and *link criticality* that describes the potential of a link to be congested in the near future. The *link criticality* is calculated based on a traffic matrix updated by the SDN controller, preventing a link from being quickly exhasuted. Based on the two metrics, link weights are calculated and the min-weight SR-paths are determined for incoming flows.

Online SR-TE solutions for throughput maximization can be combined with techniques like Multipath Transmission Control Protocol (MPTCP) [91] or multicast routing [92]. For instance, Pang et al. [75] provide an online routing scheme that combines SDN-based MPTCP and SR-TE. The proposal first sorts the available paths between the source and destination by path delay in ascending order. This is because of an observation that the smaller variance in path delay of the subflows of the MPTCP flow, the lower probability that packets will be out of order. Then, it identifies the required bandwidth of the MPTCP connection using Hedera algorithm [93], and iteratively selects the sorted paths one by one until the bandwidth requirement is satisfied. The selected paths are mapped to the corresponding segment lists and programmed on the source node to establish the MPTCP connection.

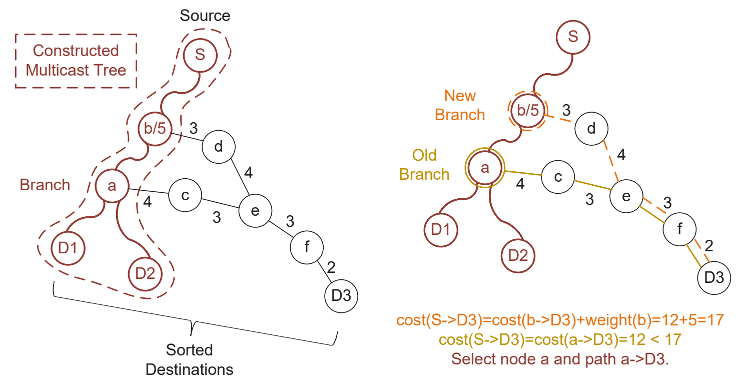

Source
Constructed Multicast Tree
S
b/5
3
d
4
Branch
a
4
c
3
e
3
f
2
D1
D2
D3
Sorted Destinations
New Branch
Old Branch
cost(S->D3)=cost(b->D3)+weight(b)=12+5=17
cost(S->D3)=cost(a->D3)=12 < 17
Select node a and path a->D3.

Fig. 14: Illustrations of selection of branch nodes and paths among the first *k* (=2) shortest paths for a destination [71].

Sheu et al. [71] propose an online SR-TE solution that collaborates with multicast to maximize throughput. Two important functions are proposed for link/node weight calculation: *betweenness function* and *congestion function*. The *betweenness function* quantifies the criticality of a link/node and prevents a link/node from continuously being selected and quickly exhausted. The *congestion function* measures the loaded state of a link/node using information of link residual bandwidth or node residual flow entries. It avoids congestion and facilitates the selection of links with more bandwidth. Note that node weight is to minimize the number of branch nodes for scalability and load balance among SDN switches in terms of flow table entries. Based on link weights, the first *k* shortest paths from source to each destination in the multicast group are calculated. Destinations are sorted according to the cost of their shortest paths and added into the multicast tree in ascending order. To select the least cost paths and branch nodes of the multicast tree, a cost function defines the path cost as the cost of the subpath from the selected branch node (already in the multicast tree) to the destination, plus the weight of the branch node if it is a new branch node (see Figure 14). The SDN controller will implement the multicast tree by programming the source node and branch nodes: each time a branch node receives a packet, it duplicates the packet, modifies the SIDs in the header and forwards the packet along the corresponding path.

The global view of a centralized controller simplifies the collection of network information, and enables the aforementioned solutions to compute segment lists in accordance with recently collected information, thus adapting to dynamic network states. However, if in-band channel is used, frequent communication with controller would add additional networking overhead.

4.2.2. *Offline Routing*

In *offline routing*, segment lists are computed according to a given traffic matrix used as a forecast of future network states. Segment lists are programmed in the network as a long-term strategy to optimize routing. Hence, an online monitoring mechanism is not required. Compared to online routing, one advantage of offline routing is the ability to serve in the scenario where a monitoring mechanism is difficult to implement, typically in networks without centralized controllers.

Tab. 7: Comparison of offline SR-TE solutions for maximizing network throughput.

Ref.	**Advantages**	**Network Type**
[61]	Low computational complexity	Full SR network
[72]	Flexible and highly parameterized	Full SR network
[73]	Unequally split traffic	Full SR network
[70]	Reduce wasted bandwidth	Hybrid SR network

Several offline SR-TE solutions for maximizing throughput have been proposed (see Table 7). Trimponias et al. [61] present a model TE_{MF} which formulates the problem of throughput maximization as Linear Programming. TE_{MF} takes a traffic matrix as input and acquires the solution of SR paths by solving a multicommodity-flow like problem, in which the SLD of each segment list is restricted to 2. In particular, TE_{MF} selects transit nodes only from a small and representative set of candidate nodes, reducing its computational complexity.

To adapt to various network topologies and traffic matrices, Zhang et al. [72] propose a flexible algorithm FPTAS which parameterizes segment number, candidate SR node set, link-state routing policy, etc. Each link in the network is associated with a link length variable, and an auxiliary graph is constructed for each flow for path computation, where edges between nodes can be represented as segments. For a given flow, FPTAS iteratively calculates edge weights of the auxiliary graph based on link lengths, and allocate a min-weight SR path to the flow, until its required bandwidth is satisfied. Link lengths are updated accordingly each time an SR path is computed, Tab. 8: Summaries of SR-TE solutions for maximizing QoE.

Ref.	Optimized Metrics	Application Examples	Summary
[50]	Bandwidth, delay	Interactive services like Web browsing or conversations	Minimize end-to-end delay for a better QoE and minimize SLD to reduce segment overhead.
[76]	Bandwidth, delay, jitter, packet loss rate	Multimedia services based on MPTCP transportation	Compute segment lists for MPTCP connections while considering QoE-related network metrics.
[77]	Bandwidth, jitter, packet loss rate	Media applications and interactive services like Web browsing	Leverage a comprehensive evaluation model to select the best paths from candidate paths.
[78]	Bandwidth, delay, jitter, packet loss rate	Multimedia services of multiple end-users	Construct multicast trees for QoE optimization based on inaccurate network information.
[79]	Bandwidth, delay	Instantaneous services like real-time video streaming	Implement an algorithm to seek optimal delay-constrained SR paths based on a novel structure SR graph.
[80]	Bandwidth, delay, packet loss rate	Multimedia services based on QUIC transportation	Identify different service flows and compute segment lists for specific services using reinforcement learning.

in which an important parameter is introduced to make tradeoffs between algorithm performance and computational cost.

Pereira et al. [73] propose an offline optimization model SALP-SR that fully exploits the weighted load balancing mechanism of SR-TE to achieve unequal traffic splitting. SALP-SR first solves the problem of IGP link weight configuration using the Evolutionary Algorithm [94]. Afterwards, segment lists are computed to implement traffic distribution, in which the amount of traffic that should be routed on each segment list is also properly computed based on a flow splitting function. Such function assigns flows to a next-hop with a probability that decreases exponentially with the extra path length (compared to the shortest path). In particular, a traffic splitting parameter is introduced in the function to cope with traffic variation better, which can be manipulated to adjust the proportion of flows routed on each segment list without changing the configuration of link weights and segment lists. Experimental results demonstrate the effectiveness of the model to well distribute the traffic and fully utilize all available bandwidth, increasing network throughput.

While the aforementioned solutions are designed for full SR networks, Gang et al. [70] propose a throughput optimized solution for hybrid SR networks, where SR and non-SR nodes coexist. For each flow in the traffic matrix, if the corresponding IGP shortest paths cross the SR domain, the subpaths in the SR domain will be rearranged for optimization. In this case, a depth first search is implemented to find out all feasible paths for the flows without violating the routing rules in the IP domain[5], as well as the path length (i.e., hop counts) and cost constraints. The purpose of introducing these constraints is to reduce wasted bandwidth, since a path with longer length or higher cost will occupy more bandwidth and thus may decrease throughput. Each flow will be allocated with a path with sufficient bandwidth without violating the link capacity constraint.

In order to adapt to changes of traffic pattern, above solutions require periodical update on the traffic matrix. This may not work well with sudden traffic changes such as traffic burst. One approach is to combine both offline and online schemes, in which online scheme serves as a more adaptive paradigm for local adjustment that tackles events that are not forecast by offline scheme.

4.3. Maximizing Quality of Experience

Quality of Experience (QoE) is a parameter describing the satisfaction or annoyance of end-users for an application or service [55]. It is influenced by a vast variety of factors that can be interrelated, including those related to the network performance.

Numerous network metrics, such as *bandwidth*, *delay*, *jitter* and *packet loss rate*, can affect QoE of specific applications or services [56]. For instance, video streaming services have certain requirements on high transmission bandwidth and low delay or jitter [95]. To this end, many SR-TE solutions have been proposed to optimize several interrelated network metrics for QoE maximization, as shown in Table 8.

Davoli et al. [50] present an open source heuristic SR-TE algorithm [96] to optimize path delay, which contributes to a better QoE. The algorithm initially performs a Constrained Shortest Path First (CSPF) algorithm to allocate TE paths to flows, satisfying bandwidth requirements of flows while ensuring the link

[5]The subpaths in the IP domain must be IGP shortest paths [70].

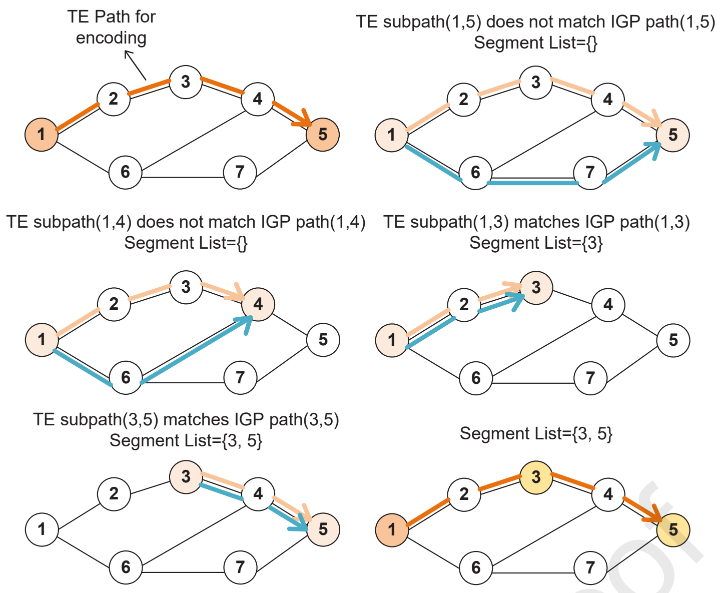

TE Path for encoding
TE subpath(1,5) does not match IGP path(1,5)
Segment List={}
TE subpath(1,4) does not match IGP path(1,4)
Segment List={}
TE subpath(1,3) matches IGP path(1,3)
Segment List={3}
TE subpath(3,5) matches IGP path(3,5)
Segment List={3, 5}
Segment List={3, 5}

Fig. 15: Illustrations of SR assignment algorithm in [50].

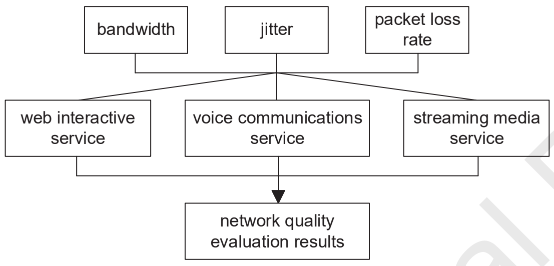


Fig. 16: FAHP-based comprehensive path evaluation model [77].

capacity constraint. The allocated paths are repeatedly optimized to minimize the average path delay until no improvement can be achieved, then delivered to an SR assignment algorithm to be encoded as segment lists with minimal length. The main idea of SR assignment algorithm is to concatenate the minimal number of IGP shorest paths (each represented by an IGP global SID) into the corresponding TE path, as depicted in Figure 15.

Considering the growing demands of instantaneous services, Luttringer et al. [79] also focus on the provision of delay-constrained SR paths. In particular, a novel structure SR graph is derived from the original network graph to enable efficient SR path computation, in which edges are represented by segments. A Node Segment encoding a set of ECMP-aware shortest paths represents an edge in the SR graph, whose IGP metric is the shortest-path cost and delay is the maximum delay of the paths specified by the segment. An Adjacency Segment corresponding to an edge in the original graph remains an edge of the same IGP metric and delay in the SR graph. Given the SR graph as input, an open source algorithm BEST2COP [97] is implemented for path computation. The main idea is to, for a specific source node, seek the best paths towards each destination node under the MSD and delay constraints while optimizing the IGP metrics, by iteratively expanding the current best paths by one segment (i.e., one edge in the SR graph) in the Bellman-Ford fashion.

Hou et al. [77] propose an SR-TE solution to optimize QoE for specific applications. Based on fuzzy analytic hierarchy process (FAHP) [98], a comprehensive evaluation model is designed to measure the quality of candidate paths. The model selects bandwidth, jitter and packet loss rate as key performance indexes (KPIs) to evaluate the performance of a path on three services: web interactive service, voice communications service and streaming media service (see Figure 16). It allows the weight vector of services to be determined by operators in the light of users' preference, while the weight vector of KPIs is obtained by establishing the fuzzy judgement matrix. As a result, the model is able to evaluate the comprehensive performance of candidate paths on the three services, so that the best paths can be selected for each source destination pair to optimize QoE. The candidate path set of the model is generated by calculating the first *k* min-weight paths for each source destination pair, where link weights are properly computed using a multi-objective particle swarm optimization (MOPSO) algorithm [99].

To tackle the challenge of the explosive growth of multimedia traffic, several works take bandwidth, delay, jitter and packet loss rate into consideration when computing SR paths. Barakabitze et al. [76] focus on multimedia services based on MPTCP transportation, and propose a QoE-centric Multipath Routing Algorithm (QoMRA) for establishing MPTCP connections. In addition to optimize the network metrics, QoMRA also utilizes the concepts of *link criticality* and *link congestion* in [74] (see Section 4.2.1) for path computation to predict the future load of a link and mea-

Tab. 9: Summaries of SR-TE solutions for minimizing energy consumption.

Ref.	Switch-off Devices	Energy-efficient Strategy
[81]	Links	Switch off underutilized links and obtain SR paths for rerouting traffic by solving a multicommodity flow problem.
[82]	Links	Redistribute traffic on SR paths with sufficient bandwidth, and switch off links not used by any SR path.
[83]	Links, SDN switches	Aggregate flows on a subset of network infrastructures using segment lists, and switch off the idle links and SDN switches.

sure its current loaded state. On the other hand, Yang et al. [78] target multimedia services of multiple end-users, and propose an SR-TE solution for constructing multicast trees for multicast requests. In particular, as collected network information is often inaccurate due to propagation delay, information aggregation and so forth, a probability method is implemented to compute optimal paths for the multicast group based on inaccurate information. The multicast tree is constructed from the optimal paths with the objective to minimize branch nodes for scalablity.

While optimizing a series of network metrics is significant to QoE optimization, manipulating flows at a per-flow basis can lead to a performance enhancement, compared with per-destination based routing. In fact, per-flow routing provides more flexibility in resource allocation due to the ability to steer flows belonging to the same source and destination but different services along different optimized SR paths. An example is provided in [80], where a per-flow based SR-TE solution is proposed. A classifier is implemented to identify different kinds of services for QUIC (Quick UDP Internet Connection) [100] traffic. Based on the classification, a specific routing strategy (i.e., segment lists) is determined using reinforcement learning to meet the strict QoE requirements (bandwidth, delay, packet loss) of end-users, thus leading to the enhancement of end-users' QoE.

4.4. *Minimizing Energy Consumption*

Energy consumption of large scale networks has become a primary concern in networking [101]. As a matter of fact, networks are designed with redundancy of network infrastructures as so to support services and applications in peak period [102]. However, compared to their full capacities, network infrastructures are underutilized most of the time, while they consume almost the same amount of energy regardless of their utilization [103], thus entailing the wastage of a great amount of energy. Hence, energy consumption minimization is also an important TE objective with practical significance.

The minimization of energy consumption can be achieved by switching off a subset of network devices (e.g., links, SDN switches). To ensure correct operations of networks, SR-TE is responsible to reroute the existing traffic on available resources, where the benefits of SR-TE can be exploited. Table 9 provides summaries of the solutions in this section.

STREETE [81] is an online SR-TE solution for minimizing energy consumption in backbone networks. It intends to switch off links that transmit less data and reroute traffic on the available links. Obviously, the connectivity of network must be guaranteed whenever a link is switched off. In order to reroute traffic, STREETE leverages the global view of SDN controller to solve a multicommodity flow problem to obtain a set of SR paths, which are implemented by the controller before links are actually switched off. An enhanced algorithm is proposed in [104], which allows STREETE to selectively turn on links that are previously switched off when detecting a network state close to congestion. This enables STREETE to rapidly react to the risk of congestion while keeping as many links inactive as possible, thus improving the robustness and energy efficiency of STREETE.

Ghuman et al. [82] propose an online SR-TE solution to optimize energy efficiency in data center networks. Different from STREETE, new SR paths with sufficient bandwidth are computed in the first place to carry the load of the network, and links that are not utilized by any SR path are then switched off. In particular, traffic manipulation is performed at a per-packet level to achieve better performance in load balancing and congestion avoidance. Controller will keep monitoring the utilization of each active link and if any exceeds a pre-defined threshold, the previously closed links will be switched back on.

Another energy-efficient solution is provided by Jia et al. [83], which intends to aggregate flows on a subset of network infrastructures and turn off the idle links and SDN switches. It leverages a critical algorithm to calculate the rerouting paths and identify whether a link/switch should be turned off. For each flow traversing through a specific link, the algorithm computes a shortest path that does not contain such link. This path will be implemented as a segment list for rerouting the flow if it ensures the link capacity constraint and delay constraint of the flow. Next, a link will be switched off if all the flows previously routed over it are rerouted, and an SDN switch will be turned off if all the links incident to it are also switched off.

Experimental analysis of above solutions have demonstrated their capability of saving a large amount of energy. However, since switching off network devices will inevitably affect network services (e.g., causing unexpected events like micro-loops [105]), it is inevitable to make tradeoffs between energy saving and network performance. Hence, this issue still needs more in-depth investigation and exploration to reduce energy consumption while guaranteeing good network performance.

5. Node Management for SR-TE

The number and locations of SR enabled nodes as well as the scope of candidate transit SR nodes greatly impact the efficiency and effectiveness of SR-TE solutions. For example, more SR enabled nodes and larger scope of candidate nodes provide more opportunities for optimization but increase cost and complexity. This section will discuss several solutions on node management for SR-TE, including SR node deployment and candidate node selection.

5.1. SR Node Deployment

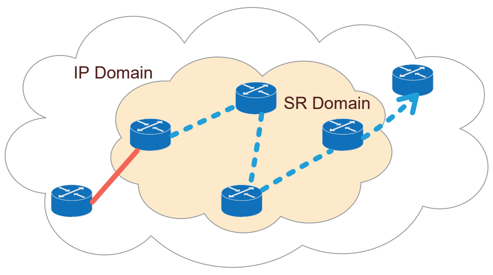
Based on economical and technical consideration, an incremental deployment of SR nodes is often more practical and stable when implementing SR in networks [24]. Besides, since SR-TE must be implemented in the SR domain, the deployment of SR nodes also plays an important role in ensuring high efficiency of SR-TE. Different strategies for SR node deployment may lead to different SR-TE performance in hybrid networks [66]. Therefore, the objective is to achieve almost the same or similar performance as the full SR network on the one with a limited set of SR nodes [24].

Several schemes for deploying SR nodes have been investigated (see Figure 17) [66]:

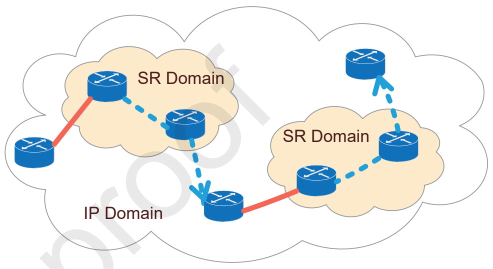
- *Single SR domain (S-SRD)*: All SR enabled nodes constitute a single connected SR domain.
- *Multiple SR domains (M-SRD):* The set of SR enabled nodes are divided into multiple disjoint SR domains.

When the number of SR nodes to be deployed is relatively small, the M-SRD scheme generally leads to a performance enhancement compared to S-SRD scheme [66], due to the fact that M-SRD increases the possibility to accommodate nodes that are critical to the improvement of SR-TE performance of the network but may not form a connected set. In particular, this advantage will diminish when the number of SR nodes deployed in the network is sufficiently large, as the disjoint SR domains will converge to a single connected one [66].

Linear Programming can be exploited to solve the problem of SR node deployment. For example, Cianfrani et al. [66] propose an MILP model that determines the locations of SR nodes with the objective to



(a) Single SR Domain (S-SRD)



(b) Multiple SR Domains (M-SRD)

Fig. 17: Illustrations of SR domain design schemes [66]. The red solid lines represent IGP shortest-path routing. The blue dotted lines represent SR-path routing.

minimize MLU. The model takes two important variables as input, i.e., N_{max} and K_{max}, representing the maximal number of SR nodes and SR domains, respectively. Obviously, by adjusting the value of K_{max}, it is able to provide solutions for S-SRD ($K_{max} = 1$) or M-SRD schemes ($K_{max} > 1$). Given the two variables along with a traffic matrix, the model returns effective solutions for SR node deployment for minimizing MLU.

Based on the analysis of node properties, efficient heuristic algorithms can also be designed for SR node deployment. For instance, Gang et al. [70] investigate the topological properties of nodes for SR domain design and introduce a metric $R(v) = \theta{\cdot}D_v{+}\beta{\cdot}SP_v$, where θ and β are weight parameters subject to $\theta + \beta = 1$. Metric D_v corresponds to the degree of node v. The higher D_v, the more adjacencies of node v and thereby more paths can be used to forward traffic when v is upgraded to SR node. Metric SP_v associates with the number of shortest paths traversing through node v, which quantifies the criticality of a node. Based on $R(v)$, a greedy heuristic algorithm is proposed for S-SRD scheme, which circulates the process to select the node with the highest $R(v)$ and seeks the next one from the adjacent nodes of the current SR domain, until the number of SR nodes reaches a pre-defined upper bound.

Cianfrani et al. [66] also analyze the properties of

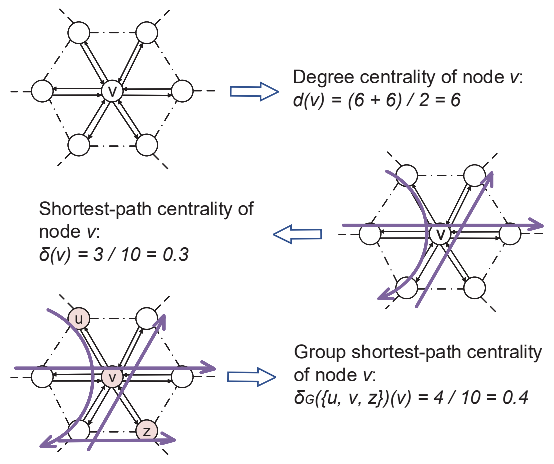

Degree centrality of node v:
d(v) = (6 + 6) / 2 = 6
Shortest-path centrality of node v:
δ(v) = 3 / 10 = 0.3
Group shortest-path centrality of node v:
δG({u, v, z})(v) = 4 / 10 = 0.4
v
u
z

Fig. 18: Illustrations of topological centralities proposed in [61]. Assume that network is a directed graph and the total number of shortest paths is 10.

SR nodes in order to simplify the design of SR domains. They propose a traffic based parameter *Most Loaded Link (MLL)*, which refers to the maximum utilization of the outgoing links of a node when peak traffic is routed through IGP link-state protocols. Preliminary analysis shows that MLL can be used as a criterion to efficiently design SR domains. Tian et al. [24] confirm the efficiency of MLL in the deployment of SRv6 nodes. In particular, the SRv6 nodes may not constitute any connected sets, as SRv6 allows intermediate nodes on an SR path to be unaware of SRv6 [5], thus providing more flexibility.

In conclusion, efficient solutions for SR node deployment can be obtained through Linear Programming or heuristic algorithms. In fact, according to the requirements and properties of a network, there are many other important factors that should be considered when determining locations of SR nodes, such as practical requirements (e.g., Service Function Chaining [106]), link attributes (e.g., link capacity or delay) and node attributes (e.g., hardware capability).

5.2. Candidate Node Selection

The majority of existing SR-TE solutions take all nodes in the entire SR domain as candidate transit nodes. Although this offers more choices on SR path selection, it may reduce the scalability of these solutions due to the expensive computation. Fortunately, studies have shown that a small and representative set of candidate nodes for transit node selection is sufficient to achieve satisfactory TE performance, while computational complexity is significantly reduced [61]. Hence, selecting appropriate candidate nodes can be an elegant approach to increase the scalability of SR-TE solutions.

Methodologies on candidate node selection have been studied in [25][61][68], in which two different ways to generate candidate nodes are provided. Trimponias et al. [61] present a method to generate ex-

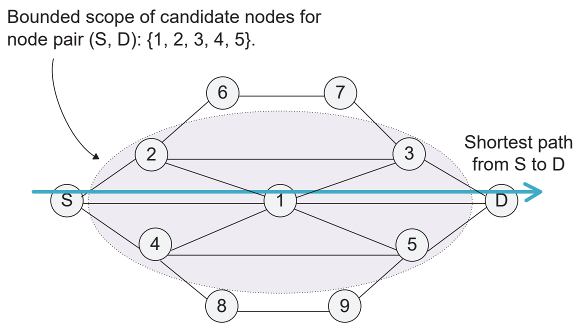

Bounded scope of candidate nodes for node pair (S, D): {1, 2, 3, 4, 5}.
Shortest path from S to D
S
1
2
3
4
5
6
7
8
9
D

Fig. 19: Illustrations of stretch bounding in [68]. Suppose stretch $\alpha = 1.5$. The length of shortest path form S to D is 2. Hence, the path length from S to D through a candidate node should not exceed: $1.5 \times 2 = 3$, resulting in the candidate node set $\{1, 2, 3, 4, 5\}$.

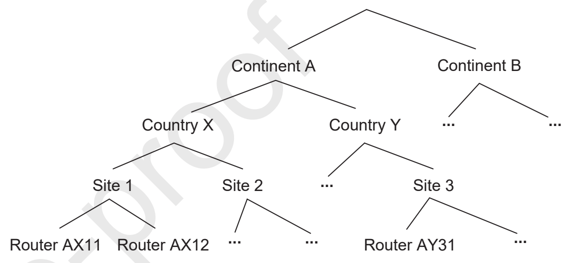


Fig. 20: Illustrations of hierarchy tree of geographical information of routers [25].

actly one set of candidate nodes for all source destination pairs. Three topological centralities for candidate node selection are proposed: (i) degree centrality $d(v)$: the average of in-degree and out-degree of v; (ii) shortest-path (SP) centrality $\delta(v)$: the proportion of the number of shortest paths that go through v; (iii) group shortest-path (GSP) centrality $\delta_G(C)(v)$: the proportion of the number of shortest paths that go through *any node* in C. Figure 18 provides an example for the three centralities. These centralities treat all links equally, while in practice links are further characterized by link capacities. Therefore, link cost (reciprocal of link capacity) is introduced in above centralities: degree is described as the sum of link costs incident to the node, while shortest paths of SP/GSP centrality are calculated with link costs. Experimental results show that GSP centrality is the most effective and robust criterion for candidate node selection. In particular, it is demonstrated that only about 2.48% ~ 7% of all nodes is sufficient to obtain good solutions of SR paths.

A *stretch bounding* method is proposed by Settawatcharawanit et al. [68] to construct candidate node sets for each source destination pair. It is based on the idea that a node should not be considered as candidate node if its location is too far from the source to destination. In other words, only when the length of a path from source to destination through an intermediate node does not exceeds the length of the corresponding shortest path by α times, can this intermediate node participate in the transit node selection (see Figure 19). The *stretch* then corresponds to the aforementioned α. An enhanced version of this method is provided in [107], where a randomized sampling approach is implemented to optimize the results of stretch bounding, so that traffic can load balance across the candidate nodes.

Unlike stretch bounding, the method in [25] selects candidate nodes for each source destination pair by considering the geographical locations of routers rather than their topological locations. Each router is associated with three geographical labels: continent, country and site. Based on the geographical labels, a hierarchy tree is constructed, where every router is a leaf node (see Figure 20). The candidate nodes are then restricted to the leaf nodes of the sub-tree with root as the lowest common ancestor (LCA) of source and destination. For instance, in Figure 20, the LCA of router $AX11$ and $AY31$ corresponds to Continent A. Hence, every leaf node of Continent A may be used as a transit node (e.g., router $AX12$), while other nodes must be excluded in transit node selection. One major advantage of this method is that the propagation delay between routers is taken into account in candidate node selection.

The aforementioned methodologies for candidate node selection are independent of specific SR-TE solutions and can be performed in offline. Thus, the originated set of candidate nodes can be introduced as a constraint in the SR-TE solutions described in previous sections. Although restricting the scope of candidate nodes will sacrifice TE performance to gain computational efficiency, this approach will be of great practical significance when scalability becomes a bottleneck and the sacrifice of appropriate performance can be tolerated.

6. Open Issues and Future Research

SR-TE has been extensively studied in academia, with segment list computation as the main stream. However, there are still many issues that need to be further investigated. Several research directions worth of future exploration are summarized as follows.

1. **SR-TE solutions for multi-domain TE.** Previous studies on multi-domain SR-TE range from mechanism for multi-domain SR-TE (e.g., [108]), architectural design for multi-domain SR networks (e.g., [109][110]), to methodology for exchanging SID information across multi-domain (e.g., [51][109]). However, although network service providers have certain requirements on multi-domain TE paths [111], solutions for segment list computation still remain to be further explored. In this research area, the SR Policies in each domain should be fully utilized when computing segment lists. Hence, one possible approach for multi-domain TE optimization is to identify the best concatenation of BSID of SR Policies. This provides several benefits: simplify segment list computation, fully exploit the benefits of BSID (see Section 3.3) and produce less state information stored on headends.
2. **Combine offline and online SR-TE solutions.** It is possible to combine offline and online SR-TE solutions to implement more sophisticated TE strategies. In normal circumstances, traffic are routed over SR paths provided by offline solutions, while those unexpected situations (e.g., constant and rapid growth of traffic) are tackled by online solutions, which make routing adjustment over a subset of existing SR paths. Several important issues should be considered in this area:
    - Exploration on tradeoffs between routing performance and computational complexity. More complex offline solutions could be developed to obtain more optimal SR paths that represent a long term strategy for routing optimization. On the other hand, the development of online solutions may place more emphasis on computational efficiency, as unexpected events should be tackled promptly.
    - Investigation on appropriate criteria to launch online solutions, e.g, how long the maximum link utilization has been above a threshold. Strict criteria may cause network instability due to frequent adjustment, while loose ones may entail congestion due to unresponsive actions.
    - Study on the efficient selection of SR paths to make routing adjustment. In particular, the number of SR paths to be adjusted should be minimized to avoid traffic flapping and oscillation.
3. **Hybrid SR network planning.** In the process of upgrading networks to be SR capable, heterogeneous networks are often inevitable, especially for large scale networks. Hence, an efficient planning for a hybrid SR network is necessary, so as to enable a smooth transition from a pure IP network to a full SR one, as well as guarantee high performance of SR-TE. Concerns regarding this area include:
    - SR node deployment: More efficient algorithms for SR node deployment should be developed, which consider various important factors such as SFC and link/node attributes.
    - Link weight configuration: By properly setting link weights, it is possible to maximize the number of IGP shortest paths crossing the SR domains. In this case, more flows can be rerouted for optimization when en-

tering the SR domains, leading to an enhancement of TE performance.

- Segment list computation: The salient features of Adjacency Segment provides several key benefits for TE optimization, such as better utilization of parallel links in the network [63] and more flexibility in path selection [65]. However, Adjacency Segment still remains to be explored for TE optimization in hybrid SR networks. For example, how to utilize Adjacency Segment to determine appropriate egress/ingress nodes for flows across the SR and non-SR domains to optimize a specific intent of TE.

4. **Development of a programmable SR-TE optimizer.** A programmable SR-TE optimizer allows network operators to directly declare their intents [49]. Such optimizer is responsible to transfer the descriptions of an intent into an objective and a set of constraints for the built-in algorithms to compute optimized SR paths. Hence, it simplifies network management as well as achieves high flexibility to cope with the diverse requirements of operators. The major challenges in this area include:

   - How to design an interactive interface? Such interface should be able to transfer the requirements of operators into the intents to optimize, and possibly report the results of path computation for network management.
   - How to develop robust algorithms for segment list computation? To cope with various intents, scalability and flexibility should be the key in the development of routing algorithms, in which optimization approaches such as hybrid Constraint Programming and Column Generation may be exploited.

   An example is provided in [49], where an optimizer DEFO exposes a high-level interface to enable intuitive declaration of intents via a small Domain Specific Language (DSL) [112]. However, it is necessary to continue research on optimizers with more effective and flexible programmability.

5. **Implement machine learning algorithms in SR-TE.** The next generation networks are expected to be self-driven, where automatic configuration, optimization and adaptation are enabled [113]. This imposes requirements for effective and adaptive SR-TE solutions for routing optimization, which can be achieved by machine learning (ML) algorithms that learn the regularity of the network from the data sets collected in the past, and adaptively make decisions for future events. Several issues in SR-TE that may be addressed by ML algorithms include:

   - Segment list computation: One possible scheme can be leveraging a powerful routing algorithm to produce optimal SR paths as data sets to train the ML algorithm [114]. After the training process, the ML algorithm will substitute for the routing algorithm as a more adaptive paradigm for routing optimization [114].
   - Candidate node selection: Another possible area is to train the ML algorithm to learn the regularity of the distribution of critical candidate nodes, with massive optimal paths for different traffic matrices as data sets. In this case, the ML algorithm adaptively derives a set of critical candidate nodes for transit node selection according to the traffic pattern. This will accelerate the process of path computation and improve the scalability of SR-TE solutions.
   - Traffic matrix prediction: Recently, researchers have explored the application of ML algorithms in traffic matrix prediction [115][116]. This may open opportunities for providing more accurate and representative traffic matrix for routing optimization, ensuring excellent performance of many SR-TE solutions (e.g., [61][59][66]).

7. Conclusions

In this paper, a comprehensive survey on SR-TE is conducted, with the focus on SR-TE architecture and recent SR-TE research on routing optimization. We introduce a critical framework SR Policy for instantiating segment lists in the network, as well as significant SR-TE algorithms, i.e., Flexible Algorithm and SR-native algorithm, for segment list computation. The key benefits and major challenges of SR-TE are also summarized, along with effective measures to resolve the challenges. Moreover, we investigate many state-of-the-art SR-TE solutions for segment list computation, covering their intents, optimization approaches, routing types, strengths and weaknesses, etc. Node management important to SR-TE including SR node deployment and candidate node selection is discussed in detail as well. At last, we give an end to this paper by presenting a set of future research directions, so as to complement the research fields on SR-TE.

8. Acknowledgment

This work has been partially supported by Chinese National Research Fund (NSFC) No. 62172189 and 61772235; Natural Science Foundation of Guangdong Province No. 2020A1515010771; Science and Technology Program of Guangzhou No. 202002030372.

References

[1] G. Trimponias, Y. Xiao, X. Wu, H. Xu, Y. Geng, Node-constrained traffic engineering: Theory and applications, IEEE/ACM Transactions on Networking 27 (4) (2019) 1344–1358.

[2] A. Farrel, Overview and principles of Internet traffic engineering, Internet-Draft draft-ietf-teas-rfc3272bis-12, Internet Engineering Task Force, work in Progress (2021). URL `https://datatracker.ietf.org/doc/html/draft-ietf-teas-rfc3272bis-12`

[3] Y. Zuo, Y. Wu, G. Min, L. Cui, Learning-based network path planning for traffic engineering, Future Generation Computer Systems 92 (2019) 59–67.

[4] Z. N. Abdullah, I. Ahmad, I. Hussain, Segment routing in software defined networks: A survey, IEEE Communications Surveys & Tutorials 21 (1) (2018) 464–486.

[5] P. L. Ventre, S. Salsano, M. Polverini, A. Cianfrani, A. Abdelsalam, C. Filsfils, P. Camarillo, F. Clad, Segment routing: A comprehensive survey of research activities, standardization efforts, and implementation results, IEEE Communications Surveys & Tutorials 23 (1) (2021) 182–221.

[6] M. A. Mohiuddin, S. A. Khan, A. P. Engelbrecht, Fuzzy particle swarm optimization algorithms for the open shortest path first weight setting problem, Applied Intelligence 45 (3) (2016) 598–621.

[7] D. B. Magnani, I. A. Carvalho, T. F. Noronha, Robust optimization for OSPF routing, IFAC-PapersOnLine 49 (12) (2016) 461–466.

[8] L. Liu, J. Zhou, X. Guo, R. Qi, A method for calculating link weight dynamically by entropy of information in SDN, in: IEEE 22nd International Conference on Computer Supported Cooperative Work in Design (CSCWD), IEEE, 2018, pp. 535–540.

[9] V. Pereira, P. Sousa, M. Rocha, A comparison of multi-objective optimization algorithms for weight setting problems in traffic engineering, Natural Computing (2020) 1–16.

[10] O. Lemeshko, O. Yeremenko, Linear optimization model of MPLS traffic engineering fast reroute for link, node, and bandwidth protection, in: 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET), IEEE, 2018, pp. 1009–1013.

[11] M. A. Panhwar, K. A. Memon, A. Abro, D. Zhongliang, S. A. Khuhro, Z. Ali, Efficient approach for optimization in traffic engineering for multiprotocol label switching, in: IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC), IEEE, 2019, pp. 1–7.

[12] S. Kumaran, S. Prasad, N. S. Kumar, et al., Implementation and performance analysis of traffic engineered multiprotocol label switching network for IPv6 clients, in: International Conference on Electronics and Sustainable Communication Systems (ICESC), IEEE, 2020, pp. 766–773.

[13] Z. Ali, G. Swallow, F. Zhang, D. Beller, RSVP-TE path diversity using exclude route, RFC 8390 (2018). `doi:10.17487/RFC8390`.

[14] T. Schüller, N. Aschenbruck, M. Chimani, M. Horneffer, Failure resilient traffic engineering using segment routing, in: IEEE 44th Conference on Local Computer Networks (LCN), IEEE, 2019, pp. 422–429.

[15] V. Sridharan, P. M. Mohan, M. Gurusamy, QoC-aware control traffic engineering in software defined networks, IEEE Transactions on Network and Service Management 17 (1) (2019) 280–293.

[16] J. Zhang, M. Ye, Z. Guo, C.-Y. Yen, H. J. Chao, CFR-RL: Traffic engineering with reinforcement learning in SDN, IEEE Journal on Selected Areas in Communications 38 (10) (2020) 2249–2259.

[17] M. I. Salman, B. Wang, Boosting performance for software defined networks from traffic engineering perspective, Computer Communications 167 (2021) 55–62.

[18] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, A. Vahdat, B4: Experience with a globally-deployed software defined WAN, in: Proceedings of ACM SIGCOMM 2013, ACM, 2013, pp. 3–14.

[19] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, R. Wattenhofer, Achieving high utilization with software-driven WAN, in: Proceedings of ACM SIGCOMM 2013, ACM, 2013, pp. 15–26.

[20] Z. Li, Y. Hu, PASR: An efficient flow forwarding scheme based on segment routing in software-defined networking, IEEE Access 8 (2020) 10907–10914.

[21] Z. Li, Y. Hu, T. Hu, R. Ma, PARS-SR: A scalable flow forwarding scheme based on segment routing for massive giant connections in 5G networks, Computer Communications 159 (2020) 206–214.

[22] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, R. Shakir, Segment routing architecture, RFC 8402 (2018). `doi:10.17487/RFC8402`.

[23] C. Filsfils, K. Michielsen, F. Clad, D. Voyer, Segment routing part II: Traffic engineering, 2nd Edition, Independently Published, 2019.

[24] Y. Tian, Z. Wang, X. Yin, X. Shi, Y. Guo, H. Geng, J. Yang, Traffic engineering in partially deployed segment routing over IPv6 network with deep reinforcement learning, IEEE/ACM Transactions on Networking 28 (4) (2020) 1573–1586.

[25] T. Schüller, N. Aschenbruck, M. Chimani, M. Horneffer, Failure resiliency with only a few tunnels–Enabling segment routing for traffic engineering, IEEE/ACM Transactions on Networking 29 (01) (2021) 262–274.

[26] C. Filsfils, K. Michielsen, K. Talaulikar, Segment routing part I, 1st Edition, CreateSpace Independent Publishing Platform, USA, 2017.

[27] A. Bashandy, C. Filsfils, S. Previdi, B. Decraene, S. Litkowski, R. Shakir, Segment routing with the MPLS data plane, RFC 8660 (2019). `doi:10.17487/RFC8660`.

[28] C. Filsfils, D. Dukes, S. Previdi, J. Leddy, S. Matsushima, D. Voyer, IPv6 segment routing header (SRH), RFC 8754 (2020). `doi:10.17487/RFC8754`.

[29] S. Previdi, L. Ginsberg, C. Filsfils, A. Bashandy, H. Gredler, B. Decraene, IS-IS extensions for segment routing, RFC 8667 (2019). `doi:10.17487/RFC8667`.

[30] P. Psenak, C. Filsfils, A. Bashandy, B. Decraene, Z. Hu, IS-IS extension to support segment routing over IPv6 dataplane, Internet-Draft draft-ietf-lsr-isis-srv6-extensions-15, Internet Engineering Task Force (2021). URL `https://datatracker.ietf.org/doc/html/draft-ietf-lsr-isis-srv6-extensions-15`

[31] P. Psenak, S. Previdi, C. Filsfils, H. Gredler, R. Shakir, W. Henderickx, J. Tantsura, OSPF extensions for segment routing, RFC 8665 (2019). `doi:10.17487/RFC8665`.

[32] P. Psenak, S. Previdi, OSPFv3 extensions for segment routing, RFC 8666 (2019). `doi:10.17487/RFC8666`.

[33] Z. Li, Z. Hu, D. Cheng, K. Talaulikar, P. Psenak, OSPFv3 extensions for SRv6, Internet-Draft draft-ietf-lsr-ospfv3-srv6-extensions-02, Internet Engineering Task Force (2021). URL `https://datatracker.ietf.org/doc/html/draft-ietf-lsr-ospfv3-srv6-extensions-02`

[34] S. Previdi, C. Filsfils, A. Lindem, A. Sreekantiah, H. Gredler, Segment routing prefix segment identifier extensions for BGP, RFC 8669 (2019). `doi:10.17487/RFC8669`.

[35] C. Filsfils, S. Previdi, G. Dawra, E. Aries, D. Afanasiev, Segment routing centralized BGP egress peer engineering, Internet-Draft draft-ietf-spring-segment-routing-central-epe-10, Internet Engineering Task Force (2017). URL `https://datatracker.ietf.org/doc/html/draft-ietf-spring-segment-routing-central-epe-10`

[36] C. Filsfils, K. Talaulikar, D. Voyer, A. Bogdanov, P. Mattes, Segment routing policy architecture, Internet-Draft draft-ietf-spring-segment-routing-policy-11, Internet Engineering Task Force (2021). URL `https://datatracker.ietf.org/doc/html/draft-ietf-spring-segment-routing-policy-11`

[37] S. Sivabalan, C. Filsfils, J. Tantsura, W. Henderickx, J. Hard-

wick, Path computation element communication protocol (PCEP) extensions for segment routing, RFC 8664 (2019). `doi:10.17487/RFC8664`.

[38] M. Koldychev, S. Sivabalan, C. Barth, S. Peng, H. Bidgoli, PCEP extension to support segment routing policy candidate paths, Internet-Draft draft-ietf-pce-segment-routing-policy-cp-04, Internet Engineering Task Force, work in Progress (2021).
URL `https://datatracker.ietf.org/doc/html/draft-ietf-pce-segment-routing-policy-cp-04`

[39] S. Previdi, C. Filsfils, K. Talaulikar, P. Mattes, E. C. Rosen, D. Jain, S. Lin, Advertising segment routing policies in BGP, Internet-Draft draft-ietf-idr-segment-routing-te-policy-12, Internet Engineering Task Force (2021).
URL `https://datatracker.ietf.org/doc/html/draft-ietf-idr-segment-routing-te-policy-12`

[40] K. Patel, G. V. de Velde, S. R. Sangli, J. Scudder, The BGP tunnel encapsulation attribute, RFC 9012 (2021). `doi:10.17487/RFC9012`.

[41] C. Filsfils, Network programming with SRv6, Ph.D. thesis, Université de Liège, Liege, Belgique (2020).

[42] P. Psenak, S. Hegde, C. Filsfils, K. Talaulikar, A. Gulko, IGP flexible algorithm, Internet-Draft draft-ietf-lsr-flex-algo-15, Internet Engineering Task Force (2021).
URL `https://datatracker.ietf.org/doc/html/draft-ietf-lsr-flex-algo-15`

[43] C. Filsfils, K. Talaulikar, P. G. Król, M. Horneffer, P. Mattes, SR policy implementation and deployment considerations, Internet-Draft draft-filsfils-spring-sr-policy-considerations-07, Internet Engineering Task Force (2021).
URL `https://datatracker.ietf.org/doc/html/draft-filsfils-spring-sr-policy-considerations-07`

[44] D. M. Yeung, D. Katz, K. Kompella, Traffic engineering (TE) extensions to OSPF version 2, RFC 3630 (2003). `doi:10.17487/RFC3630`.

[45] T. Li, H. Smit, IS-IS extensions for traffic engineering, RFC 5305 (2008). `doi:10.17487/RFC5305`.

[46] S. Giacalone, D. Ward, J. Drake, A. Atlas, S. Previdi, OSPF traffic engineering (TE) metric extensions, RFC 7471 (2015). `doi:10.17487/RFC7471`.

[47] L. Ginsberg, S. Previdi, S. Giacalone, D. Ward, J. Drake, Q. Wu, IS-IS traffic engineering (TE) metric extensions, RFC 8570 (2019). `doi:10.17487/RFC8570`.

[48] E. Osborne, Extended administrative groups in MPLS traffic engineering (MPLS-TE), RFC 7308 (2014). `doi:10.17487/RFC7308`.

[49] R. Hartert, S. Vissicchio, P. Schaus, O. Bonaventure, C. Filsfils, T. Telkamp, P. Francois, A declarative and expressive approach to control forwarding paths in carrier-grade networks, in: Proceedings of ACM SIGCOMM 2015, ACM, 2015, pp. 15–28.

[50] L. Davoli, L. Veltri, P. L. Ventre, G. Siracusano, S. Salsano, Traffic engineering with segment routing: SDN-based architectural design and open source implementation, in: 4th European Workshop on Software Defined Networks, IEEE, 2015, pp. 111–112.

[51] A. Giorgetti, A. Sgambelluri, F. Paolucci, F. Cugini, P. Castoldi, Segment routing for effective recovery and multi-domain traffic engineering, Journal of Optical Communications and Networking 9 (2) (2017) A223–A232.

[52] R. Guedrez, O. Dugeon, S. Lahoud, G. Texier, A new method for encoding MPLS segment routing TE paths, in: 8th International Conference on the Network of the Future (NOF), IEEE, 2017, pp. 58–65.

[53] Y. Xu, J. Liu, Y. Shen, J. Liu, X. Jiang, T. Taleb, Incentive jamming-based secure routing in decentralized internet of things, IEEE Internet of Things Journal 8 (4) (2020) 3000–3013.

[54] Y. Xu, J. Liu, Y. Shen, X. Jiang, Y. Ji, N. Shiratori, Qos-aware secure routing design for wireless networks with selfish jammers, IEEE Transactions on Wireless Communications (2021) 1–15.

[55] P. Le Callet, S. Möller, A. Perkis, et al., Qualinet white paper on definitions of quality of experience, European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003) (2012).

[56] A. Canovas, A. Rego, O. Romero, J. Lloret, A robust multimedia traffic SDN-based management system using patterns and models of QoE estimation with BRNN, Journal of Network and Computer Applications 150 (2020) 102498.

[57] R. Hartert, P. Schaus, S. Vissicchio, O. Bonaventure, Solving segment routing problems with hybrid constraint programming techniques, in: International Conference on Principles and Practice of Constraint Programming, Springer, 2015, pp. 592–608.

[58] R. Bhatia, F. Hao, M. Kodialam, T. V. Lakshman, Optimized network traffic engineering using segment routing, in: IEEE INFOCOM 2015, IEEE, 2015, pp. 657–665.

[59] T. Schüller, N. Aschenbruck, M. Chimani, M. Horneffer, S. Schnitter, Traffic engineering using segment routing and considering requirements of a carrier IP network, IEEE/ACM Transactions on Networking 26 (4) (2018) 1851–1864.

[60] Juniper, Junos OS administration library for routing devices: Maximum-ECMP, Website (2013).
URL `http://www.juniper.net/documentation/en_US/junos15.1/topics/reference/configuration-statement/maximum-ecmp-edit-chassis.html`

[61] G. Trimponias, Y. Xiao, H. Xu, X. Wu, Y. Geng, Centrality-based middlepoint selection for traffic engineering with segment routing, arXiv preprint arXiv:1703.05907 (2017).

[62] S. Gay, R. Hartert, S. Vissicchio, Expect the unexpected: Sub-second optimization for segment routing, in: IEEE INFOCOM 2017, IEEE, 2017, pp. 1–9.

[63] M. Jadin, F. Aubry, P. Schaus, O. Bonaventure, CG4SR: Near optimal traffic engineering for segment routing with column generation, in: IEEE INFOCOM 2019, IEEE, 2019, pp. 1333–1341.

[64] H. Roomi, S. Khorsandi, Semi-oblivious segment routing with bounded traffic fluctuations, in: Iranian Conference on Electrical Engineering (ICEE), IEEE, 2018, pp. 1670–1675.

[65] X. Li, K. L. Yeung, Traffic engineering in segment routing networks using MILP, IEEE Transactions on Network and Service Management 17 (3) (2020) 1941–1953.

[66] A. Cianfrani, M. Listanti, M. Polverini, Incremental deployment of segment routing into an ISP network: A traffic engineering perspective, IEEE/ACM Transactions on Networking 25 (5) (2017) 3146–3160.

[67] E. Moreno, A. Beghelli, F. Cugini, Traffic engineering in segment routing networks, Computer Networks 114 (2017) 23–31.

[68] T. Settawatcharawanit, V. Suppakitpaisarn, S. Yamada, Y. Ji, Segment routed traffic engineering with bounded stretch in software-defined networks, in: IEEE 43rd Conference on Local Computer Networks (LCN), IEEE, 2018, pp. 477–480.

[69] A. Mendiola, J. Astorga, E. Jacob, M. Higuero, A survey on the contributions of software-defined networking to traffic engineering, IEEE Communications Surveys & Tutorials 19 (2) (2017) 918–953.

[70] Y. Gang, P. Zhang, X. Huang, T. Yang, Throughput maximization routing in the hybrid segment routing network, in: Proceedings of the 2nd International Conference on Telecommunications and Communication Engineering, 2018, pp. 262–267.

[71] J.-P. Sheu, Y.-C. Chen, A scalable and bandwidth-efficient multicast algorithm based on segment routing in software-defined networking, in: IEEE International Conference on Communications (ICC), IEEE, 2017, pp. 1–6.

[72] J. Zhang, Q-SR: An extensible optimization framework for segment routing, arXiv preprint arXiv:2012.13171 (2020).

[73] V. Pereira, M. Rocha, P. Sousa, Traffic engineering with three-segments routing, IEEE Transactions on Network and Service Management 17 (3) (2020) 1896–1909.

[74] M.-C. Lee, J.-P. Sheu, An efficient routing algorithm based on segment routing in software-defined networking, Com-

puter Networks 103 (2016) 44–55.

[75] J. Pang, G. Xu, X. Fu, SDN-based data center networking with collaboration of multipath TCP and segment routing, IEEE Access 5 (2017) 9764–9773.

[76] A. A. Barakabitze, L. Sun, I.-H. Mkwawa, E. Ifeachor, A novel QoE-centric SDN-based multipath routing approach for multimedia services over 5G networks, in: IEEE International Conference on Communications (ICC), IEEE, 2018, pp. 1–7.

[77] X. Hou, M. Wu, M. Zhao, An optimization routing algorithm based on segment routing in software-defined networks, Sensors 19 (1) (2019) 49.

[78] S. Yang, C. Xu, L. Zhong, J. Shen, G.-M. Muntean, A QoE-driven multicast strategy with segment routing–A novel multimedia traffic engineering paradigm, IEEE Transactions on Broadcasting 66 (1) (2019) 34–46.

[79] J.-R. Luttringer, T. Alfroy, P. Mérindol, Q. Bramas, F. Clad, C. Pelsser, Computing delay-constrained least-cost paths for segment routing is easier than you think, in: IEEE 19th International Symposium on Network Computing and Applications (NCA), IEEE, 2020, pp. 1–8.

[80] V. Tong, S. Souihi, H. A. Tran, A. Mellouk, Service-centric segment routing mechanism using reinforcement learning for encrypted traffic, in: 16th International Conference on Network and Service Management (CNSM), IEEE, 2020, pp. 1–5.

[81] R. Carpa, O. Glück, L. Lefevre, Segment routing based traffic engineering for energy efficient backbone networks, in: IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS), IEEE, 2014, pp. 1–6.

[82] K. S. Ghuman, A. Nayak, Per-packet based energy aware segment routing approach for data center networks with SDN, in: 24th International Conference on Telecommunications (ICT), IEEE, 2017, pp. 1–6.

[83] X. Jia, Y. Jiang, Z. Guo, G. Shen, L. Wang, Intelligent path control for energy-saving in hybrid SDN networks, Computer networks 131 (2018) 65–76.

[84] Y. Guo, Z. Wang, X. Yin, X. Shi, J. Wu, Traffic engineering in hybrid SDN networks with multiple traffic matrices, Computer Networks 126 (2017) 187–199.

[85] M. Chiesa, G. Rétvári, M. Schapira, Oblivious routing in IP networks, IEEE/ACM Transactions on Networking 26 (3) (2018) 1292–1305.

[86] N. Geng, M. Xu, Y. Yang, E. Dong, C. Liu, Adaptive and low-cost traffic engineering based on traffic matrix classification, in: 29th International Conference on Computer Communications and Networks (ICCCN), IEEE, 2020, pp. 1–9.

[87] S. Deng, H. Balakrishnan, Traffic-aware techniques to reduce 3G/LTE wireless energy consumption, in: Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, 2012, pp. 181–192.

[88] P. Baptiste, C. Le Pape, W. Nuijten, Constraint-based scheduling: Applying constraint programming to scheduling problems, Vol. 39, Springer Science & Business Media, 2012.

[89] D. Pisinger, S. Ropke, Large neighborhood search, in: Handbook of Metaheuristics, Springer, 2019, pp. 99–127.

[90] J. Gondzio, P. González-Brevis, P. Munari, New developments in the primal–dual column generation technique, European Journal of Operational Research 224 (1) (2013) 41–51.

[91] A. Ford, C. Raiciu, M. J. Handley, O. Bonaventure, C. Paasch, TCP extensions for multipath operation with multiple addresses, RFC 8684 (2020). `doi:10.17487/RFC8684`.

[92] Z. AlSaeed, I. Ahmad, I. Hussain, Multicasting in software defined networks: A comprehensive survey, Journal of Network and Computer Applications 104 (2018) 61–77.

[93] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, A. Vahdat, et al., Hedera: Dynamic flow scheduling for data center networks., in: Proceedings of USENIX NSDI 2010, USENIX, 2010, pp. 89–92.

[94] V. Pereira, M. Rocha, P. Cortez, M. Rio, P. Sousa, A framework for robust traffic engineering using evolutionary computation, in: IFIP International Conference on Autonomous Infrastructure, Management and Security, Springer, 2013, pp. 1–12.

[95] S.-S. Zhu, Y.-N. Dong, C. Xu, A statistical QoE-QoS model of video streaming services, in: Proceedings of the 6th International Conference on Computing and Data Engineering, 2020, pp. 195–199.

[96] SDN based traffic engineering and segment routing.
URL `https://github.com/netgroup/SDN-TE-SR`

[97] BEST2COP.
URL `https://github.com/talfroy/BEST2COP`

[98] A. Emrouznejad, W. Ho, Fuzzy analytic hierarchy process, CRC Press, 2017.

[99] S. Lalwani, S. Singhal, R. Kumar, N. Gupta, A comprehensive survey: Applications of multi-objective particle swarm optimization (MOPSO) algorithm, Transactions on Combinatorics 2 (1) (2013) 39–101.

[100] J. Iyengar, M. Thomson, QUIC: a UDP-based multiplexed and secure transport, Internet-Draft draft-ietf-quic-transport-34, Internet Engineering Task Force, work in Progress (2021).
URL `https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-34`

[101] R. Maaloul, L. Chaari, B. Cousin, Energy saving in carrier-grade networks: A survey, Computer Standards & Interfaces 55 (2018) 8–26.

[102] J. A. Manjate, M. Hidell, P. Sjödin, Can energy-aware routing improve the energy savings of energy-efficient ethernet?, IEEE Transactions on Green Communications and Networking 2 (3) (2018) 787–794.

[103] B. G. Assefa, Ö. Özkasap, RESDN: A novel metric and method for energy efficient routing in software defined networks, IEEE Transactions on Network and Service Management 17 (2) (2020) 736–749.

[104] R. Cârpa, M. D. de ASSUNÇÃO, O. Glück, L. Lefèvre, J.-C. Mignot, Responsive algorithms for handling load surges and switching links on in green networks, in: IEEE International Conference on Communications (ICC), IEEE, 2016, pp. 1–7.

[105] A. Bashandy, C. Filsfils, S. Litkowski, B. Decraene, P. Francois, P. Psenak, Loop avoidance using segment routing, Internet-Draft draft-bashandy-rtgwg-segment-routing-uloop-10, Internet Engineering Task Force, work in Progress (2020).
URL `https://datatracker.ietf.org/doc/html/draft-bashandy-rtgwg-segment-routing-uloop-10`

[106] J. M. Halpern, C. Pignataro, Service function chaining (SFC) architecture, RFC 7665 (2015). `doi:10.17487/RFC7665`.

[107] T. Settawatcharawanit, Y.-H. Chiang, V. Suppakitpaisarn, Y. Ji, A computation-efficient approach for segment routing traffic engineering, IEEE Access 7 (2019) 160408–160417.

[108] C. Filsfils, S. Previdi, G. Dawra, W. Henderickx, D. Cooper, Interconnecting millions of endpoints with segment routing, RFC 8604 (2019). `doi:10.17487/RFC8604`.

[109] N. Kukreja, R. Alvizu, A. Kos, G. Maier, R. Morro, A. Capello, C. Cavazzoni, Demonstration of SDN-based orchestration for multi-domain segment routing networks, in: 18th International Conference on Transparent Optical Networks (ICTON), IEEE, 2016, pp. 1–4.

[110] R. Guerzoni, D. Perez-Caparros, P. Monti, G. Giuliani, J. Melian, G. Biczók, Multi-domain orchestration and management of software defined infrastructures: A bottom-up approach, in: 3th Europe Conference on Networks and Communications (EUCNC), IEEE, 2016, pp. 1–6.

[111] J. Sun, S. Sun, K. Li, D. Liao, V. Chang, Efficient algorithm for traffic engineering in multi-domain networks, in: International Conference on Smart Computing and Communication, Springer, 2017, pp. 365–374.

[112] D. Ghosh, DSLs in action, Manning Publications Co., 2010.

[113] M. Usama, J. Qadir, A. Raza, H. Arif, K.-L. A. Yau, Y. Elkhatib, A. Hussain, A. Al-Fuqaha, Unsupervised machine learning for networking: Techniques, applications and research challenges, IEEE Access 7 (2019) 65579–65615.

[114] M. K. Awad, M. H. H. Ahmed, A. F. Almutairi, I. Ahmad, Machine learning-based multipath routing for software defined networks, Journal of Network and Systems Management 29 (2) (2021) 1–30.

[115] K. Gao, D. Li, L. Chen, J. Geng, F. Gui, Y. Cheng, Y. Gu, Incorporating intra-flow dependencies and inter-flow correlations for traffic matrix prediction, in: IEEE/ACM 28th International Symposium on Quality of Service (IWQoS), IEEE, 2020, pp. 1–10.

[116] L. Nie, X. Wang, S. Wang, Z. Ning, M. Obaidat, B. Sadoun, S. Li, Network traffic prediction in industrial Internet of Things backbone networks: A multi-task learning mechanism, IEEE Transactions on Industrial Informatics (2021) 1–10.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: