UMinho | 2019 Vítor Manuel Sá Pereira **Intradomain Routing Optimization Based on Evolutionary Computation**

Universidade do Minho
Escola de Engenharia

Vítor Manuel Sá Pereira

Intradomain Routing Optimization Based on Evolutionary Computation

January 2019

Universidade do Minho
Escola de Engenharia

Vítor Manuel Sá Pereira

Intradomain Routing Optimization Based on Evolutionary Computation

Doctoral Thesis
Doctoral Degree in Informatics

Supervised by:
Professor Doutor Pedro Sousa
Professor Doutor Miguel Rocha

January 2019

STATEMENT OF INTEGRITY

I hereby declare having conducted my thesis with integrity. I confirm that I have not used plagiarism or any form of falsification of results in the process of the thesis elaboration.
I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

University of Minho, 28 January 2019

Full name: Vítor Manuel Sá Pereira

Signature: Vitor Manuel Sá Pereira

Acknowledgments

Firstly, I would like to express my sincere gratitude to my advisors, Professor Pedro Sousa and Professor Miguel Rocha, for the continuous support of my Ph.D study and related research, for their patience, motivation, and knowledge. Their guidance helped me in all the time of research and writing of this thesis.
I would like to thank my friends for accepting nothing less than excellence from me.I also would like to thank my colleagues and my students, for their patience, comprehension and support.
Last but not the least, I would like to thank my family: my parents and sisters for supporting me throughout writing this thesis and my life.

Thank you for supporting me in achieving this goal.

Abstract

The growing number of Internet-connected devices and the escalation of new Internet services, such as cloud services and video streaming, are some examples of factors that increase the volume and mutability of traffic in communication networks. The need to channel increasingly large volumes of traffic in network infrastructures with limited capacity, highlights the importance of Traffic Engineering mechanisms that aim to deliver an efficient use of network resources. Routing decisions play a key role as they define how traffic is distributed on the available paths and hence how networks resources are explored. Traditional routing protocols, such as Open Shortest Path First and Intermediate System to Intermediate System, have constraints which prevent an optimal network resources utilization. Some of these constraints are, for example, their inability to perform uneven splitting of traffic across multiple paths and their lack of a centralized control. These constraints impose additional difficulties when changes in network operating conditions need to be considered, such as significant variations in traffic necessities and link failures. When facing such changes, routing configurations need to adapt to the new conditions and ensure that the network continues to operate efficiently.

Communication technologies are constantly evolving. Recently, alternative routing solutions have emerged that enable new Traffic Engineering approaches. Network management problems and, in particular, the optimization of network resources utilization can be addressed using such alternative solutions. Software-Defined Networking and Segment Routing paradigms provide greater flexibility in the selection of routing paths, and although they can overcome many of the constraints of traditional routing protocols, it is necessary to find configurations, both scalable and manageable in real contexts, that optimize the distribution of available resources.

In this context, this research work intends to respond to the stated problems by proposing efficient mechanisms for optimizing the use of resources in networks configured with traditional Link State routing protocols, as well as in networks that implement the latest paradigms of Software Defined Networking and Segment Routing. In addition to enabling efficient resource utilization, the proposed optimization mechanisms are responsive to relevant changes in the network environment that may result from variations in traffic requirements or topology changes such as link failures. The nature of the problems, which besides not being solvable in polynomial time include more than one optimization objective in their formulation, are addressed using algorithms fostered in the field of Evolutionary Computation.

Distinct traffic requirements and different network states frequently require clashing configurations. Evolutionary Algorithms possess several characteristics that are desirable to solve problems with multiple conflicting goals and make them preferable to classical optimization methods. They provide a set of compromise solutions to problems for which there is no single optimal solution. The research work winded up in an autonomous optimization framework that integrates all the proposed Traffic Engineering methods, which is made publicly available to be freely used by researchers and network administrators.

Resumo

O surgimento e a proliferação de novos serviços de Internet, como os serviços de *cloud* e a transmissão de vídeo, bem como o aumento do número de dispositivos que se ligam diariamente à Internet, são alguns fatores que contribuem para um avolumar e uma mudança nos padrões do tráfego em redes de comunicação. A necessidade de canalizar grandes volumes de tráfego em infraestruturas de rede com capacidade limitada, enfatiza a importância da Engenharia de Tráfego que tem por objetivo proporcionar um uso eficiente dos recursos de rede. As decisões de encaminhamento desempenham um papel essencial neste contexto, pois definem como o tráfego é distribuído pelos caminhos disponíveis e, consequentemente, como os recursos de rede são utilizados. Os protocolos de encaminhamento tradicionais, como o *Open Shortest Path First* e o *Intermediate System to Intermediate System*, possuem restrições operacionais que impedem uma utilização ótima dos recursos. Algumas dessas restrições são, por exemplo, as opções limitadas de balanceamento de carga e a falta de uma gestão centralizada. Essas restrições impõem dificuldades acrescidas quando alterações nas condições de funcionamento da rede têm de ser contempladas como, por exemplo, variações significativas do volume e tráfego e falhas de ligações físicas. As configurações de encaminhamento precisam adaptar-se às novas condições operacionais e garantir que a rede continue a operar de forma eficiente.

As tecnologias de comunicação evoluem. Recentemente foram propostas soluções alternativas de encaminhamento de tráfego que permitem novas abordagens de Engenharia de Tráfego. Os problemas de gestão de redes e, em particular, a otimização da utilização de recursos podem ser abordados com recurso a essas novas soluções. Os paradigmas de *Software Defined Networking* e *Segment Routing* proporcionam uma maior flexibilidade na seleção de caminhos de tráfego, e embora sejam capazes de superar muitas das restrições dos protocolos de encaminhamento tradicionais, é necessário encontrar configurações, escaláveis e geríveis em contexto real, que otimizem a distribuição de tráfego nos recursos disponíveis.

Neste contexto, este trabalho de investigação procura responder aos problemas enunciados, propondo mecanismos eficientes para a otimização da utilização de recursos de redes configuradas com protocolos de encaminhamento *Link State* tradicionais, bem como em redes que implementam os mais recentes paradigmas de *Software Defined Networking* e *Segment Routing.* Para além de possibilitarem uma utilização eficiente dos recursos, os mecanismos de otimização propostos são responsivos a alterações relevantes no ambiente de rede que podem resultar de variações nos requisitos de tráfego ou alterações de topologia. A natureza dos problemas, que para além de não serem resolúveis em tempo polinomial incluem na sua formulação mais do que um objetivo de optimização, são abordados recorrendo a algoritmos da área da Computação Evolucionária.

Distintos requisitos de tráfego e diferentes estados de rede exigem frequentemente configurações conflituantes. Os Algoritmos Evolucionários possuem várias características que são desejáveis para resolver problemas com múltiplos objetivos e que os torna preferíveis a métodos clássicos de otimização. Eles fornecem um conjunto de soluções de compromisso em problemas para os quais não existe uma solução ótima única. O trabalho de investigação confluiu numa ferramenta de otimização que integra todos os métodos de engenharia de tráfego propostos, e que é disponibilizada para ser usada livremente por investigadores e administradores de rede.

Contents

List of Figures

List of Tables

List of Acronyms

ABR	Area Border Router
ALU	Average Link Utilization
API	Application Programming Interface
AS	Autonomous System
ASBR	Autonomous System Boundary Router
ATM	Asynchronous Transport Mode
BGP	Border Gateway Protocol
CR-LDP	Constraint-Based Label Distribution Protocol
DBR	Designated Backup Router
DR	Designated Router
DV	Distance Vector
EA	Evolutionary Algorithm
eBGP	Exterior Border Gateway Protocol
ECMP	Equal-Cost Multi-Path
EGP	Exterior Gateway Protocol
FEC	Forwarding Equivalence Class
FIB	Forwarding Information Base
FRR	Fast Reroute
iBGP	Interior Border Gateway Protocol
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
IGRP	Interior Gateway Routing Protocol
IP	Internet Protocol
IPv6	Internet Protocol version 6
IS-IS	Intermediate System to Intermediate System
ISP	Internet Service Provider
LAN	Local Area Network
LDP	Label Distribution Protocol
LED	Label Edge Router

LER	Label Edge Router
LFA	Loop-Free Alternate
LFIB	Label Forwarding Information Base
LS	Link State
LSA	Link State Advertisement
LSDB	Link State Database
LSP	Label Switching Path
LSR	Label Switched Router
MCFP	Multi-Commodity Flow Problem
MED	Multi Exit Discriminator
MLU	Maximum Link Utilization
MOEA	Multi-Objective Evolutionary Algorithm
MOP	Multi-objective Problem
MPLS	Multiprotocol Label Switching
NFV	Network Function Virtualization
NSGAII	Non-dominated Sorting Genetic Algorithm
OSPF	Open Shortest Path First
PCE	Path Computation Element
PF	Pareto Front
PSTN	Public Switched Telephone Network
QoS	Quality of Service
REH	Routing Extension Header
RIB	Routing Information Base
RIP	Routing Information Protocol
RSVP	Resource Reservation Protocol
SALP	Single Adjacency Label Path
SDN	Software Defined Networking
SID	Segment Identifier
SOEA	Single Objective Evolutionary Algorithm
SP	Shortest Path
SPEA2	Strength Pareto Evolutionary Algorithm
SPF	Shortest Path First
SPRING	Segment Packet Routing in Networking
SR	Segment Routing
STP	Spanning Tree Protocol
TE	Traffic Engineering
TI-LFA	Topology Independent Loop-Free Alternate
TOA	Trade-off Analysis

Chapter 1

Introduction

This chapter introduces today's challenges in intradomain routing and the research context that motivates this thesis. Furthermore, we point out the main objectives that we pursue throughout the work developed in this thesis in order to address the identified challenges. We then present the contributions of this work as the results of exploring the proposed concepts. This chapter ends with an overview of the thesis structure.

1.1 Introduction

The Internet is mostly a global system of interconnected store-and-forward packet-switching networks. An Internet Protocol (IP) packet arriving at a router needs to be temporarily stored in interim buffer space before being forwarded towards its destination. Network resources, specifically link bandwidth and routers buffer space, are however finite and whenever the demand for resources is close to or exceeds the network's capacity, the network becomes congested which leads to increased delivery delay and packet loss.

The problem of congestion cannot be solved only by increasing resources capacity. Although an increase in buffer size might help to mitigate congestion, it also leads to growing queues and an escalation in end-to-end delay. On the other hand, increasing links capacity or adding more links and routers can be financially prohibitive.

Congestion states are inherent in best-effort datagram networks [37] due to uncoordinated resource sharing. Best-effort was conceived with nondiscrimination in mind and with no guarantee that a data packet reaches its ultimate destination in a timely fashion. However, appropriate congestion control mechanisms can be used to provide differentiated services in such a context [17, 36]. It is possible to create classes with different Quality of Service (QoS) characteristics, by conducting traffic prioritization and resource reservation control mechanisms, pricing them accordingly. Service differentiation and best-effort service are conceptually orthogonal and although QoS provides a better service to selected network traffic over various underlying technologies, most Internet traffic is still, however, delivered using a best-effort model.

The "network neutrality" debate brought new importance to best-effort traffic forwarding [19]. The core of the argument relates to which applications, if any, should receive special treatment from the network, and whether special treatment is in the best interest of users. The concern is that Internet Service Providers (ISPs) and content providers (CPs) will decide which applications get the best service, pinching out new services, and creating an insurmountable barrier to entry for innovative new applications. Consequently, to protect users, network neutrality has devolved to a principle that all Internet traffic should be treated equally. This ongoing debate was already translated into government mandates, for example, in the United States and European Union.

When packets are dropped due to congestion, the sender host retransmits the lost packets, and the receiver reconstructs the ordered stream of data. The Transmission Control Protocol (TCP) is responsible for performing these functions [106]. By monitoring the success and failure of sending packets to the receiver, the TCP at the sending system side adjusts the rate of data transmission. If packets are lost, the sending host decreases the transmission rate to help relieve the apparent congestion. On the other hand, if packets are successfully delivered, the sending host optimistically increases the sending rate to benefit on the available bandwidth. Although the TCP congestion control guarantees a fair sharing of the links' bandwidth among the competing pairs of senders and receivers, it does not assure by itself a proper usage of the available infrastructures. While some links are congested others might be unused or lightly loaded.

Routing protocols have a major role in achieving an optimized distribution of traffic in networks' infrastructure and, consequently, in preventing congestion. Every network routing protocol performs three basic functions: to identify other routers on the network; to keep track of all the possible destinations; to make dynamic decisions for where to send each network message. The last function individually defines how traffic is distributed on the available paths and consequently has a major influence on resources utilization and congestion. Routing protocols and in particular their optimization are the motivation factor of this thesis, which will be further detailed.

1.2 Motivation

IP routing protocols have tunable parameters that operators can set to control the flow of traffic through their networks. In link-state routing protocols, those parameters are a set of non-negative weights assigned to each topology link. The routers flood the link weights throughout the network and compute shortest paths as the sum of the weights. Each router uses this information to construct a table that drives the forwarding of each IP packet to the next hop in its path to the destination. When more than one shortest path is available to the same destination, packet flows are (approximately) equally divided among the available paths, performing an Equal-Cost Multi-Path (ECMP) traffic load balancing [54].

In link-state routing, to optimize the distribution of traffic and make the best possible use of the available resources, while reducing the chance of congestion states, is to suitably set link weights. The selection of the link weights should depend on the offered estimated traffic, as captured by a matrix that represents the rate of traffic entering at router i that is destined to router j, and on the network topology. This multi-commodity flow problem, generally known as the weight setting problem, is NP-hard [39] and has been extensively studied, for example in [6, 30, 39, 40, 91].

Traditional link-state routing protocols, Open Shortest Path First (OSPF) [75] and Intermediate System to Intermediate System (IS-IS) [47], have constraints which prevent optimal use of resources: 1) the protocol limitation on even splitting of traffic across multiple shortest-path routes; 2) not all paths configurations are feasible by solely setting link weights; 3) the computational intractability of finding the best link weights using standard optimization tools; 4) the distributed nature of the routing protocol. Those constraints impose additional difficulties when changes in networks' conditions need to be contemplated, such as significant variations in traffic necessities and link failures. When facing such changes, the routing solution needs to adapt to the new conditions while assuring that the network continues performing in a satisfactory way and without congestion. In the context of a Master of Science Thesis we addressed these problems by transforming these multi-objective problems into single objective function optimization problems [80, 85].

Alternative routing solutions were recently proposed that enable new Traffic Engineering (TE) approaches to distinct networking problems, and in particular to the problem of congestion in intradomain networks. Software Defined Networking (SDN) [32] and Segment Routing (SR) [34] brought higher flexibility on the definition of forwarding paths. However, though they allow overcoming traditional routing protocol constraints, the problem of finding optimal configurations for congestion avoidance and optimized distribution of traffic in intradomain routing still needs to be addressed while contemplating scalability and complexity issues as well the mutability of network environments. Most of these problems are NP-hard and contemplate the optimization of more than one objective.

Multi-Objective Problems (MOPs) are often high-dimensional, discontinuous, multimodal and NP-Complete. Deterministic methods, because they are handicapped by their requirement for heuristics to direct or limit search, are often inefficient in the treatment of such problems [70]. Single Objective Evolutionary Algorithms (SOEAs) and Multi-Objective Evolutionary Algorithms (MOEAs) have been used successfully to solve problems with such characteristics [30, 30]. They in general provide good solutions to a wide range of optimization problems which traditional deterministic search methods find difficult.

Given the research scope and motivation of this thesis, the next section provides an insight on the plan to address the previously mentioned challenges and the goals that we pursue in the resulting work.

1.3 Objectives

Traffic engineering methods endeavor to optimize networks' performance while meeting user constraints and increase operators' benefits. This effort involves adapting the routing of traffic to the network conditions, with the goal of achieving optimal use of network resources. This general definition, although simple, encompasses NP-hard traffic engineering problems that differ in the context of routing technologies and with additional robustness constraints. We defined as a global objective to provide administrators with Evolutionary Algorithms (EAs) based methods and solutions able to optimize network resources utilization. This global objective is further stratified to address the following research questions:

- **Research Question 1: Which Multi-Objective Evolutionary Algorithm provides the best solutions for the multi-objective OSPF weight setting problem?**
  - Distinct Multi-Objective Evolutionary Algorithms have different characteristics that influence solutions quality, diversity, and convergence. Pursuing the research conducted in the master's thesis, it is important to evaluate

and compare solutions obtained with distinct MOEAs and select the one that provides better results for weights setting problems.

  - The MOEAs will be compared in various multi-objective problems and scenarios that address the pre-emptive optimization of OSPF weights for changing traffic and topology conditions.

- **Research Question 2: How to optimize resources utilization in hybrid IP/SDN networks?**
  - SDN decouples the network's control and data planes functionalities and allows to implement more flexible and versatile traffic forwarding decisions. These features allow to loosen some of the link-state constraints and explore the utilization of non-shortest paths to forward portions of traffic. The goal is to propose and evaluate a TE alternative to traditional link-state routing able to provide better usage of intradomain networks' resources.

- **Research Question 3: How to optimize resources utilization and minimize congestion that results from changes in traffic demands and link failures in hybrid IP/SDN networks?**
  - It is not sufficient to optimally accommodate known of foreseen traffic necessities. A network also needs to be able to respond to disruptive events, such as relevant changes in traffic demands (both in volume and pattern) or link failures, and such that congestion states are properly addressed. In this context, the aim is to propose and evaluate TE solutions able to minimize the impact of such events in the network congestion.

- **Research Question 4: How to optimize resources utilization in SR networks while minimizing label stack depth?**
  - Segment Routing implements a source routing model where edge nodes insert forwarding instructions, as a set of label or segment identifiers, in each packet header. Although it performs similarly to Multi-Protocol Label Switching (MPLS) [78], SR reduces the number of required protocols. SR also profits from the same flexibility of SDN but without the need to maintain a per-flow state in each router or switch. However, as forwarding paths are encoded in the packet headers, it is important to minimize the overhead by using the least number of labels or segment identifiers. In this context, the objective is to propose and evaluate a model for SR that allows optimizing networks

resources utilization using the fewest number of labels to configure each forwarding paths.

- **Research Question 5: How to optimize resources utilization and minimize congestion that results from changes in traffic demands and link failure in SR networks while minimizing label stack depth?**
  - The objective is to address the similar problem formulated for hybrid IP/SDN networks in an SR context, with the additional constraint of minimizing the label stack depth.
- **Research Question 6: Is it possible to develop an useful application framework able to assist network administrators in the optimization of routing configurations in OSPF, SDN and SR based scenarios?**
  - Such a framework, besides all including new optimization methods and proposals, should also offer tools to assist network administrators decision making.

1.4 Contributions

As the results of the evolved research work, we published the main scientific achievements as summarized in Table 1.1.

The scientific contributions, although interrelated, can be aggregated according to the routing technology in three groups:

1. **Link-state routing:** Robust congestion optimization with ECMP traffic load balancing [81, 82, 86, 99] and with unequal load balancing [82, 87];
2. **Hybrid IP/SDN routing:** The principal contributions regarding congestion optimization for hybrid IP/SDN routing were published in [82, 87]. However, relevant results which are integrated in this thesis, are yet to be submitted for publication. In particular, results for the optimization of congestion in incremental deployments of hybrid IP/SDN networks and for single link failures;
3. **Segment Routing:** The principal contributions were published in [83, 84]. Some additional results as well as a new proposal to address the optimization of resources utilization after a link failure, included in this thesis, are to be published.

Table 1.1: *Publication contribution*

Type	Year	Title	Venue
Conference	2015	Comparison of Single and Multi-Objective Evolutionary Algorithms for Robust Link-State Routing [86]	EMO 2015
		Automated Network Resilience Optimization Using Computational Intelligence Methods [81]	IDC 2015
	2016	Optimizing Load Balancing Routing Mechanisms with Evolutionary Computation [82]	IE 2016
	2017	Evolutionary Computation at Work for the Optimization of Link State Routing Protocols [87]	ACM GECCO 17
		Optimizing Segment Routing using Evolutionary Computation [83]	FNC 2017
	2018	Segment Routing Single Link Failure Congestion Optimization [84]	DCNET 2018
Journal	2016	A Framework for Improving Routing Configurations using Multi-Objective Optimization Mechanisms [99]	Journal of Communications Software and Systems
	2018	A Comparison of Multi-Objective Optimization Algorithms for Weight Setting Problems in Traffic Engineering	Submitted to international journal
	2019	Traffic Engineering with Three-Segments Routing	Submitted to international journal

1.5 Thesis Outline

The present thesis is organized as follows:

- **Chapter 2** provides an overview of the reference intradomain routing solutions, namely, Link-state routing protocols with particular emphasis on the Open Shortest Path First; Multi-protocol Label-switching (MPLS); Software Defined Networking (SDN), and Segment Routing (SR). We discuss some of their advantages and disadvantages in a Traffic Engineering perspective.

- **Chapter 3** introduces fundamental concepts on Multi-objective Optimization and describes three commonly used Evolutionary Algorithms, namely, a Single Objective EA with weighted-sum aggregation, and two Multi-Objective EAs, the Non-dominated Sorting Genetic Algorithm (NSGAII), and the Strength Pareto Evolutionary Algorithm (SPEA2). The EAs performance is compared in a set of multi-objective link weights setting optimization problems that reflect changing conditions on networks running the OSPF routing protocol.

- **Chapter 4** proposes a model to optimize network resources utilization in hybrid IP/SDN networks. The model is evaluated in single and multi-objective optimization problems, the last mirroring changes on traffic demands and single link failure scenarios. The suggested optimization model is also evaluated for an incremental deployment of hybrid IP/SDN networks.
- **Chapter 5** proposes and explores a new Traffic Engineering optimization model for Segment Routing networks. Similarly to the previous chapters, the validity of the model is appraised in the changing conditions of traffic necessities and single link failures.
- **Chapter 6** describes the optimization framework developed in the context of this work and highlights some of its main features.
- **Chapter 7** presents the conclusions of this thesis and the envisaged future work.

Chapter 2

Intradomain Routing Optimization: Concepts and Methods

This chapter main goal is to characterize the main intradomain routing solutions used in today's networks. It begins with an overview of essential routing concepts with particular emphasis on the problems that condition the proper functioning of a network. The main current intradomain routing solutions are described, namely, Link-state routing, with particular attention to the Open Shortest Path First (OSPF) routing protocol, and the more flexible solutions, Multiprotocol Label Switching (MPLS), Software-Defined Networking (SDN) and Segment Routing (SR). The chapter continues by formulating the Weight Setting Problem and explores some solutions already proposed to address it.

2.1 Introduction

The Internet consists of tens of thousands of structures called domains or Autonomous Systems (ASs), each typically administered by one single institution such as a university, corporation, or Internet Service Provider (ISP). Autonomous Systems are themselves composed of devices, routers, that forward traffic between hosts and define a connected group of one or more IP prefixes. Traffic forwarding within an AS is accomplished by following a set of clearly defined rules which establish a single routing policy [51]. Routing policies are typically imposed by running an Interior Gateway Protocol (IGP), a protocol that exchanges information between routers. This information is used to compute and enforce traffic forwarding rules.

A message sent by a computer, or device, commonly traverses various ASs before reaching its destination and, consequently, communication performance depends on how traffic is driven within and between the ASs along the path. Routing is the process of selecting a path for traffic between two points on a network or across multiple networks. Routing is run on many types of networks, from traditional circuit-switched networks, such as a Public Switched Telephone Networks (PSTN), to computer networks such as the Internet. Routing protocols and their decision-making process are, therefore, in large part responsible for the performance and reliability of the Internet.

Intradomain routing, within an AS, and interdomain routing, between ASs, rely on distinct protocols which address different problems. While intradomain routing determines paths inside a single administrative domain that traffic needs to take to reach its destination, interdomain routing treats the problem of calculating paths across domains that traffic needs to traverse to reach its destination. The two problems, being very different, present distinct challenges.

The interconnection between different Autonomous Systems is traditionally achieved using a routing protocol, an Exterior Gateway Protocol (EGP), where the most widely used EGP on the Internet is the Border Gateway Protocol version 4 (BGP-4) [90]. A BGP router typically receives information on multiple paths to the same destination from its neighbors. To realize arbitrary routing policies that maximize local objectives (e.g., profit or performance), BGP needs to reconcile ISPs policy goals, which may be conflicting, and select the best route.

Three important needs motivated the design of BGP: scalability, policy and cooperation under competitive circumstances.

- Scalability: The growth in the number of networks requires that routers must be able to handle an increasing number of prefixes. BGP must ensure that the amount of advertisement traffic scales well with the network churning and parts of the network going down and coming up.

- Policy: Each AS implements and enforce various forms of routing policy. Route announcements in BGP should allow each AS to rank its available routes arbitrarily and perform route filtering.

- Cooperation under competitive circumstances: BGP was designed so that a single administrative entity would not manage the backbone Internet infrastructure. This structure implies that the routing protocol should allow ASs to make local decisions on how to route packets.

Table 2.1: *Simplified BGP decision process.*

#	Criteria
1	Highest Local Preference
2	Lowest AS Path Length
3	Lowest origin type
4	Lowest Multi-Exit Discriminator (MED)[1]
5	eBGP-learned over iBGP-learned
6	Lowest IGP cost to border router (hot-potato routing)
7	If both paths are external, prefer the path that was received first (i.e., the oldest path)
8	Lowest router ID (to break ties)

The described objectives lead to a step by step decision process that follows an ordered list of step criteria, which may involve some BGP attributes associated with each route exchanged by the protocol. An example of such a list is presented in Table 2.1.

Like interdomain routing, intradomain routing also has particular requirements. Besides sharing the same scaling concern of BGP, forwarding paths that deliver traffic to its destination should hold an essential set of properties:

- Low stretch: ISPs' customers increasingly demand delivery of their traffic with low latency. A low path stretch contributes to such goal.
- High diversity: To increase reliability and robustness of critical services in the face of temporary end-to-end path outages, it is desirable and beneficial to have path diversity.
- Efficient resources utilization: The utilization of network components has a direct impact on the performance of the network and its resilience to failure. It is desirable to avoid that while some links are congested others might be unused or lightly loaded. In such a context, networks depend on load balancing to achieve a good distribution of traffic.

Those properties are usually difficult to achieve as they impose additional constraints to the multi-commodity flow problem. Such optimization problems are frequently NP-hard, that is, they might not be computationally solvable in polynomial time.

[1]A MED is an optional attribute used to establish a preference order over possible entry paths on an AS with multiple entry points.

2.2 Intradomain Routing

2.2.1 Intradomain Routing Protocols

Intradomain routing protocols are based on a distributed approach where, typically, each network device cooperates with others to produce, accordingly with a specific algorithm, a set of rules that guide traffic forwarding in the network. A network device has two logical levels: a control plane and a data plane. The control plane is responsible for computing paths to each destination and makes decisions about where traffic should be sent. The data plane, on the other hand, implements the control plane decisions and has in charge the task of forwarding traffic relying on the rules computed by the control plane. Figure 2.1 illustrates how routing protocols relate with routers control plane and data plane.

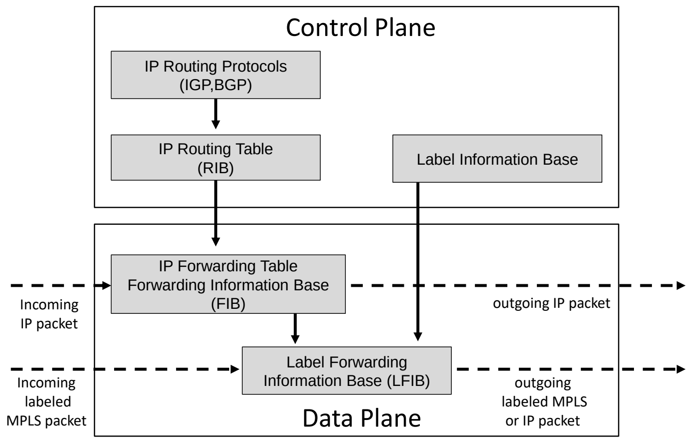

Control Plane
IP Routing Protocols (IGP,BGP)
IP Routing Table (RIB)
Label Information Base
IP Forwarding Table Forwarding Information Base (FIB)
Incoming IP packet
outgoing IP packet
Label Forwarding Information Base (LFIB)
Incoming labeled MPLS packet
outgoing labeled MPLS or IP packet
Data Plane

Figure 2.1: *Control and data planes.*

The selection of routing information learned via static definition or a dynamic routing protocol is inserted into Routing Information Base tables (RIBs). One or more RIBs programme each Forwarding Information Base (FIB) which is the actual information that a routing/switching device uses to forward traffic. The means by which multiple RIBs are programmed into a common set of FIBs vary depending on which administrative distance is used [4]. For example, when different protocols offer paths to a same destination, the routes obtained from one protocol can be preferred over others for FIB injection.

A router has numerous ways of learning the best paths toward individual IP prefixes: they might be directly connected, configured as static routes or learned through dynamic routing protocols. Dynamic protocols are aggregated into two distinct classes: Distance Vector (DV) and Link State (LS) [67]. Distance Vector routing protocols decide on the best path to a given destination by considering a distance metric. The distance is usually measured in hops, though in some Distance Vector protocols based routing the distance metric may include the latency to the destination, packets loss, or other factors that influence traffic on a given route. If the distance metric is the number of hops, then each time a packet goes through a router, a hop is considered to have been traversed. The route with the least number of hops to a given network is concluded to be the best route towards the destination. Routers regularly exchange information with neighboring routers by sending their route selection as a routing table. Examples of Distance Vector based routing protocols include Routing Information Protocol (RIP) [65] and Interior Gateway Routing Protocol[2] (IGRP)[52].

Distance Vector routing algorithms have some disadvantages. Routers running DV routing algorithms only share local information with neighboring routers and, as a consequence, a router only knows from which neighbor a route was learned, but it does not know where that neighbor learned the route from. A router cannot see beyond its neighbors which makes difficult to adapt and react to changes (e.g., a link failure). Link-state routing protocols, in contrast, require that all routers know about the paths reachable by all other routers in the network. Link-state information is flooded throughout the link-state domain to ensure all routers possess a synchronized copy of the area's link-state database. Link-state routing protocols have the advantage of robustness and a fast convergence time in comparison to DV based routing protocols. The most commonly used Link State routing protocols are Open Shortest Path First (OSPF) [75] and Intermediate System to Intermediate System (IS-IS)[47]. Link State routing protocols are discussed in more detail in the next section.

2.2.2 Link-State Routing

Link-state protocols are based on the existence of a same global map distributed on all routers running the protocol. This map is dynamically built and not imposed by an external source. The network map and all information about routers and links (as well as routes) are maintained in a state database on each router. The database is not a map in the usual sense of the word, but a set of records that represent the network topology as a list of links between routers.

Routers advertise an assortment of information through Link-state Advertisements (LSA) which include their neighbors, the networks to which they are connected, as well as accessibility information redistributed by other routing protocols. When a router

[2]IGRP is now absolete and has been replaced by the Enhanced-IGRP (EIRGP) [3]. EIGRP is a Hybrid routing protocol, and has properties of Distance Vector and Link State routing protocol.

boots, it gets a complete picture of its neighbors, routers that have access to the same network, and builds a routing table by calculating the best paths for each destination prefix. From that moment on, routers only receive LSA updates that reflect some change.

Paths computation is performed using the shortest path first (SPF) algorithm, also known as Dijkstra's algorithm, a greedy algorithm for solving the single-source shortest path problem in graphs with non-negative edge weights. The SPF algorithm is run, when necessary, by recalculating all paths to the various destinations (full SPF), or performing a partial path calculation, for example when a single external route is changed. Changes are propagated independently of the routers path calculation process, enabling link-state algorithms to adapt dynamically to changing conditions.

Shortest Path Algorithm

The shortest path (SP) algorithm was developed by Edsger Dijkstra in 1956 [28] and is the best-known algorithm for finding the shortest path between two vertices u and v on a weighted direct graph $G = (V, E)$. Each edge of the graph is associated with a weight w, represented by a real number.

The weight $w(p)$ of a path $p = \langle v_0, v_1, ..., v_i \rangle$ is the sum of the arc weights on the path. The Dijkstra algorithm is a greedy algorithm which, by making local choices, solves the shortest path problem with a single origin for non-negative weights. Given a vertex of origin $s \in V$, the intent is to find the shortest path from s to all remaining vertices $v \in V$. A pseudo-code of this algorithm is presented in Algorithm 1.

The algorithm encompass two main procedures: Initialization and Relaxation.

- Initialization:

  The Dijkstra's Algorithm partitions all vertices, or nodes, into two distinct sets: unsettled (Q) and settled (V). Initially, all vertex are in the unsettled set Q, i.e., they all are yet to be evaluated. A vertex v is moved from the unsettled to the settled set if a shortest path from the source s to this vertex has been found.

  For each vertex v, a variable d keeps track of the shortest distance from s to v, and a variable p is used to store the predecessor of each vertex v on the shortest path. When the algorithm starts, the distances $d(v)$ to all nodes $v \neq s$ are infinite. The predecessor of s is s and all other predecessors are null.

- Relaxation:

  The relaxation of an edge (u, v) consists of testing whether it is possible to improve the shortest path, found so far, to v passing by a node u. If a change in the path

Algorithm 1 Dijkstra's algorithm.

```
1: $S = NULL$
2: $Q = V$
3: $d(s) = 0$
4: $p(s) = s$
5: for all $v \in V, v \neq s$ do
6:     $d(v) = INF$
7:     $p(v) = NULL$
8: end for
9: while $Q \neq \emptyset$ do
10:     $u = EXTRACT\_MIN(Q)$
11:     $S \cup \{u\}$
12:     for all $v$ adjacent to $u$ do
13:         if $d(v) > d(u) + w(u,v)$ then
14:             $d(v) = d(u) + w(u,v)$
15:             $p(v) = u$
16:             $ADD\_TO\_UNSETTELD(v,Q)$
17:         end if
18:     end for
19: end while
```

represents an improvement, the values of d and p are updated and the node v is added to the nodes which need evaluation. At each iteration, the invariant $Q = V - S$ must be met. The algorithms runs until the unsettled set is empty and the final result is a loop-free shortest path tree from node s to all vertices.

The step by step description of Algorithm 1 is:

- 1-8: Initialization process. The distance from the initial vertex s to himself is null while the distance to the remaining nodes is infinite. All nodes in the link-state database are added to the unsettled nodes set Q.
- 9: Beginning of the main cycle which will end when there are no untreated nodes.
- 10-11: The node u, whose distance from the root is the smallest, is removed from Q and added to S. In the first iteration, node u coincides with s.
- 12-18: Relaxation process.

The shortest path tree is built from the list of predecessors. The complexity of the algorithm is delimited by $O((l + n\log(n))$, where $n = |V|$ is the number of vertices and $l = |E|$ is the number of arcs.

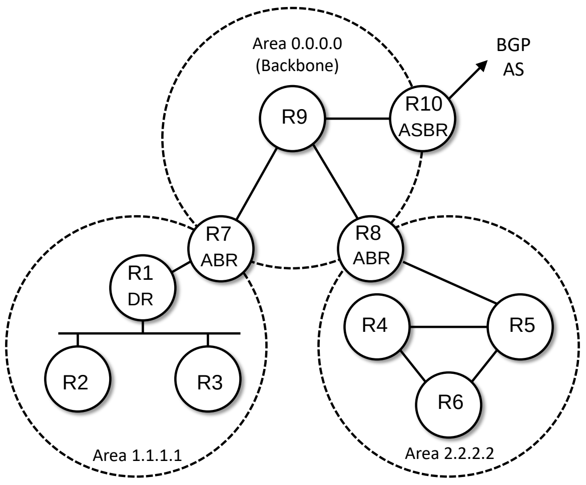


Figure 2.2: *OSPF network example.*

2.2.3 Open Shortest Path First

Open Shortest Path First is a widely used link state protocol, and therefore every router is required to know the entire network topology. For reasons of scalability, OSPF divides the routing domain or autonomous system into multiple areas. ASs OSPF areas are arranged around an area 0 or backbone, to which the remaining areas are connected. All OSPF routes with origin in one area and destination in another area need to pass through the backbone area. Routers with interfaces in various areas are known as Area Border Routers (ABRs) while routers which lie at the AS boundaries exchanging routing information with routers from other ASs are known as Autonomous System Boundary Routers (ASBRs), Figure 2.2. By splitting a routing domain into multiple areas, OSPF can reduce the amount of information that needs to be shared and synchronized.

OSPF routers, with interfaces in broadcast Local Area Networks (LANs) or point-to-point links, discover other routers in their immediate vicinity through a periodic exchange of "hello" messages. Each router sends a "hello", a multicast message, through all its interfaces in specific intervals of time (*HelloInterval*). A "hello" message includes a list of all nodes from which it has recently received a hello. If a router A finds that it is listed in the hello message of neighbor B, then its adjacency with the neighbor is bidirectional. After a neighborhood relation is established, if router A has router B as its neighbor, it synchronizes its Link-state Database (LSDB) with the LSDB of the neighbor. Router A then generates a new LSA listing the adjacency state of all its interfaces that belong to the same area (such as the connection between it and neighbor B) and sends the LSA through its interfaces. When a neighbor router receives the LSA, it resends it through all its interfaces in the area, except for the interface through which the LSA was received. Thus, the LSA is transmitted throughout the entire area. When a router does not receive an LSA acknowledgment from a neighbor within a certain configured time interval (*RxmtInterval*), it retransmits the same LSA to the neighbor. Each router in the area thus receives the LSA, initially transmitted by router A, and becomes aware of the neighbors with witch the router A has established a complete adjacency.

Two routers remain adjacent while they continue to exchange hello messages periodically. The adjacency is considered to be broken when a router does not receive any hello message from its neighbor for a defined period (*RouterDeadInterval*). There are various possible reasons for the loss of connectivity. It occurs if the link between the router and the neighbor fails or if the neighboring router is no longer functional. In some cases, the link layer protocol may signal a link failure event and allow routers to terminate an adjacency without having to wait for the *RouterDeadInterval* to expire. The loss of an adjacency causes the generation of a new LSA on the router. This LSA is sent to the entire area, informing the remaining routers of the adjacency failure. When a router receives a new LSA, it recalculates and updates its routing table.

In summary, the convergence of the OSPF routing algorithm, after a topology change, encompasses the following steps:

1. Detection of a topology change by neighboring routers.
2. Establishment or loss of adjacencies by the routers affected by the change of topology.
3. Generation of new LSAs and consequent flooding throughout the area.
4. Routing table calculation by each router upon the reception of the new LSAs.

The overall convergence time depends on the time needed to complete each of the steps mentioned.

To minimize the amount of information to be shared and to make the protocol more scalable, OSPF chooses a router to become a Designated Router (DR) as well as a router to be a Designated Backup Router (BDR). Instead of each router exchanging update information with the remaining routers in the area, all routers exchange adjacency information with the DR and the BDR, which become responsible for generating and sending topology updates to the remaining routers. This procedure also makes easier the synchronization of Link-state databases.

2.2.4 Multi-Protocol Label Switching

The IP routing process is destination based, that is, when a packet arrives at a router, the IP address of the destination is looked up and mapped against the forwarding table to identify the appropriate outgoing network interface and next-hop IP address. The underlying principle is to operate on a packet-by-packet basis and forward packets from the same flow on the same path. In IP Traffic Engineering (TE), all packets for a given destination follow the same path due to a destination-oriented routing paradigm and if equal cost paths are found, traffic can take two or more different paths to the destination. However, there are times when controlling the flow of packets beyond IP traffic engineering based on link weight settings is desirable. For example, depending on the type of packet or class of traffic, it might be required that packets be forwarded along distinct routing paths. Therefore, a mechanism was required able to define a specific path and force packets from a particular traffic stream to take this path independently of the designated IP shortest path. Multiprotocol Label Switching (MPLS) is such a mechanism [78].

In the late 1990's, the limitations and inflexibility in IP routing and forwarding, coupled with the ongoing quest to improve switching performance, led to the development of MPLS. Label switching was envisioned and designed as a simple mechanism that would operate between layers 2 and 3 in the standard Internet protocol stack and enables efficient lookups at each hop on a designated path. In traditional IP routing, hop-by-hop forwarding decisions are independent from router to router. Each router makes its forwarding decisions based locally on the packet's header and the result of the routing algorithm. However, in MPLS every packet arriving at a Label Edge Router (LER) is assigned to a Forwarding Equivalence Class (FEC). A label which identifies the FEC is then inserted at the packet's header, Figure, and all packets which belong to a particular FEC, and sourced at a particular node, will follow the same path, a label switching path (LSP). Routers on the path must be label-switched routers (LSRs) to understand and implement the label instruction and do not require additional IP lookup. Each LSR maintains a valid mapping from the label of an incoming packet to a label to be attached to the packet before being sent out, Figure 2.1. LSRs maintain states in terms of input/output labels associated with a particular path which may be designated for a particular class of traffic flows. At the final destination router of the MPLS domain, the label is removed, and the packet is delivered via standard IP routing.

The label switching technique is not new. Frame Relay and Asynchronous Transport Mode (ATM) [50] use it to move frames or cells throughout a network. However, the basic concept of MPLS is to speed up the delivery of packets by assigning them to a particular FEC just once. These labels' values are distributed to the other LSRs using at least one of Label Distribution Protocol (LDP) [111], Resource Reservation Protocol with Traffic Engineering Extensions (RSVP-TE) [13], Constraint-Based Routed LDP (CR-LDP) or Multiprotocol BGP by LERs.

LDP and RSVP-TE are the two most commonly used label distribution protocols and are two different means of telling neighboring routers which label to use. Initially, the RSVP protocol [16] was developed to support the IntServ [48] QoS model resource reservations for each flow that demands specific QoS requirements as it traverses the network. RSVP was later extended to support the creation and maintenance of LSPs, as well as being able to make bandwidth reservations for the signaled LSPs. Using its signaling element, RSVP-TE sets up an LSP end-to-end (ingress-to-egress). Therefore, label distribution is coordinated among all the LSRs along a path. LDP, on the other hand, has no signaling element. It sets up LSPs hop-by-hop, and labels can be distributed between neighbors independently of other LSRs along the path. Because it has no signaling element, LDP depends on the network's IGP to determine the path an LSP must take, whereas RSVP-TE can set up paths independently from what the IGP determines to be the optimal path to a destination.

MPLS Traffic Engineering (MPLS-TE) not only can control where and how traffic is routed on networks, manage capacity, prioritize different services, and prevent congestion, but also improves convergence during a link or node failure. In a typical IP network, running a link state protocol solely, the best path calculation happens on-demand when a failure is detected. As discussed in the previous section, it can take several seconds to recalculate the best paths and push those changes to the router hardware, particularly on a busy router. Furthermore, transient routing loops may also occur while every router in the network learns about the topology change. MPLS, on the other hand, implements a fast reroute mechanism (FRR)[10] where the backup paths calculation happens before the failure occurs. Backup paths are pre-programmed into the router FIB awaiting activation, which can happen in milliseconds following failure detection. MPLS FRR has a response time of under 50 milliseconds. Because the entire path is set within the LSP, routing loops do not occur during IGP convergence.

There are two different ways to provide LSP protection: One-to-One Protection and Many-to-One Protection also known as Facility Backup [10]. In One-to-One Protection an individual backup path is fully signaled through RSVP for every LSP, at every point where protection is provided (i.e., every node). To fully protect a LSP that crosses N nodes, there could be as many as (N - 1) detours, i.e., partial one-to-one backup tunnels.

it is therefore desirable to merge detours back to the main LSP, wherever possible, to minimize processing overhead. Many-to-One Protection, on the other hand, creates a single bypass LSP between two nodes to be protected. During a failure, multiple LSPs are rerouted over the bypass LSP.

2.2.5 Software-Defined Networking

Despite the promise of a more flexible forwarding scheme provided by MPLS, most IP networks continue to heavily rely on shortest-path rules. It has been acknowledged for a long time that the indirect control of forwarding paths by setting administrative weights makes the TE tasks very difficult to execute [40, 88]. On the contrary, MPLS-based mechanisms allow network administrators to deploy almost any possible routing pattern. However, the introduction of such a powerful tool shifts the problem from how to set weights so that traffic uses more or less the desired routes to a new type of problem. Though with MPLS traffic between any origin-destination pair can be forwarded along any path configuration, deciding the routes for all such pairs at the same time can be a very difficult task and requires complex configurations.

Dynamic IP routing protocols and MPLS couple the control plane with the data plane, as shown in Figure 2.1. The control plane on each device exchanges information with others to decide how packets should be processed. Since the control plane is distributed on the devices, it does not have a global view of the network and cannot make good network-wide decisions. Furthermore, the interface between the control plane and the data plane is closed, and operators can not only change the control plane or data plane without changing the other. These problems make it complicated to implement network management routing tasks both at layers 2 and 3 of the protocol stack. Operators cannot flexibly customize the control plane for new routing protocols which are obligated to use vendors protocols implementation (e.g., Spanning Tree Protocol (STP) for layer two routing, and OSPF or IS-IS for layer three routing). All the configurations are done on a per-device basis which represents a difficult challenge for operators to reason about the interactions between different devices in the network. By breaking this vertical integration, that is, with the separation of the control and data planes, the recent concept of Software Defined Networking (SDN) [32] opens new opportunities for devising new TE optimizing mechanisms. The control plane is logically centralized at a controller. The controller gathers information from the data plane and provides a global operational view. Applications running on top of the controller make packet processing decisions, based on the global view, and distribute the decisions to the data plane via the controller.

The open application programming interface (API), provided by SDN implementations such as OpenFlow [66], enables controllers to directly interact with the forwarding plane of network devices such as switches and routers. Instructions are provided by SDN controllers instead of multiple, vendor-specific devices and protocols which simplifies network design and operation through a standard API. A dedicated OpenFlow Switch is a dumb datapath element that forwards packets between ports, as defined by a remote control process. OpenFlow can, therefore, be considered as a forwarding abstraction, which makes use of the already existing flow tables. The OpenFlow forwarding abstraction provides a match plus action mechanism. For each flow-entry, that matches on any existing header, or new header, an action is deployed. There are three basic actions: forward a flow's packets to a given port; encapsulate and forward a flow's packets to a controller; drop the flow's packets.

SDN match-action tables model network devices as general packet processing devices, regardless of whether a device acts as a layer two switch, a layer three router, a load balancer or a firewall. A match-action table contains a list of rules with multiple components. The most critical components are priority, match and action. The match component specifies the header pattern of packets, the action component specifies the processing of packets, and the priority specifies the order when a packet matches multiple rules. Typically, only the first packet in a new flow is forwarded to the controller, so that the controller can decide if the flow should be added to the flow table and what action should be deployed. SDN also enables finer grain traffic forwarding schemes and more expedite and flexible configurations. The routing application can use custom routing algorithms based on the global view provided by the control plane and can easily make packet forwarding decisions based on different header fields. It is also possible to explore novel networking configurations assuming as an example: i) mixed approaches involving traditional routing protocols along with SDN specific forwarding schemes or ii) entirely new/disruptive traffic forwarding schemes configured by centralized control entities using the OpenFlow protocol.

2.2.6 Segment Routing

Segment routing (SR) [33] is a Software-Defined Networking (SDN) [32] technology proposed by the IETF Segment Packet Routing in Networking (SPRING) working group in 2013. SR was initially proposed to address several drawbacks of existing IP/MPLS networks concerning scalability, simplicity and ease of operation. The key idea in segment routing is to break up the routing path into smaller parts, called segments, enabling a better network utilization and improving traffic engineering tasks.

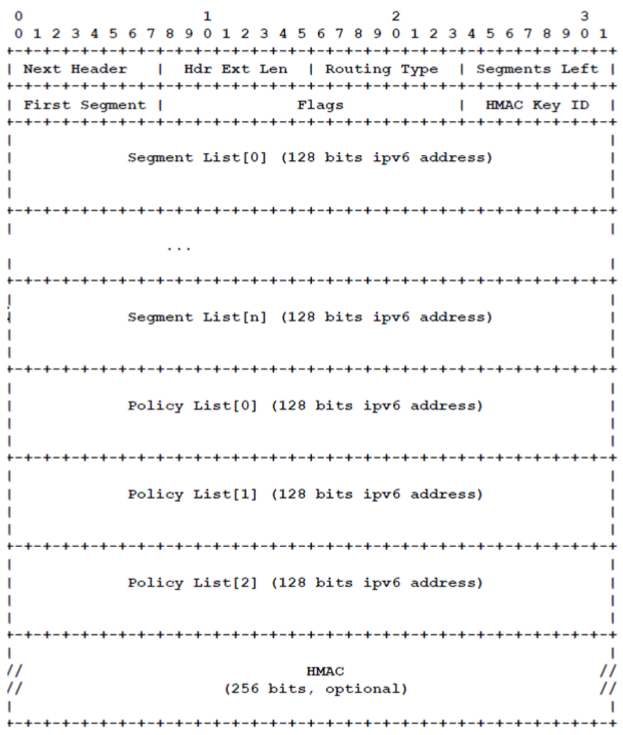
Segment Routing implements the source routing paradigm where a source node directs incoming packet flow through the network by specifying a list of intermediate destinations that a packet must visit on its way to the final destination. Segment Routing is enabled by a small number of extensions to already existing intradomain routing protocols and BGP, and it can be applied both in MPLS and IPv6 (SRv6) architectures with minimal changes to both forwarding planes.

Segment Routing decouples edge to edge forwarding paths into a set of instructions, segments, that nodes execute on the incoming packet. A segment can describe a topological or a service instruction. While a service instruction pinpoints a service or a Network Function Virtualization (NFV) [71] where a packet should be delivered, a topological instruction determines a path along which a packet should be forwarded. A segment is identified with a Segment Identifier (SID). The terms segment and segment identifier will be used interchangeably. The Segment Routing concept is the same both in IP and MPLS environments. However, some implementation details and protocol extensions differ between Segment Routing in MPLS and Segment Routing in IPv6 networks. A segment is encoded as a 32 bits MPLS label and list of segments are equivalent to MPLS label stack. In IPv6 architecture, a segment is represented as an IPv6 address, Figure 2.3. This is enabled by introducing Segment Routing Header (SRH) [35] that allows multiple IPv6 addresses to be encoded in source router so that multiple intermediate hops can be specified.

Segments have a global or a local significance within the network. A global segment is an instruction that is supported by all nodes in the domain and must be unique within the same domain. Any node must have in its FIB all global segments. The values of global segment identifiers are taken from the Segment Routing Global Block (SRGB). A local segment, on the other hand, is an instruction that is only supported by the node originating it and take a value outside the SRGB range. Since it has only local significance, a local segment value is related only to local router FIB. A router is not aware of local segments of other routers in a domain. Global Segments, similarly to labels in MPLS, need to be advertised to all network nodes. However, SR does not require SRVP or LDP to perform such a task. Segments are advertised using IGP and BGP routing protocols. For both protocols, Segment Routing extensions are defined to include SR information that enables segments' signaling. The distribution of segments may also be assured by a controller or a Path Computation Element (PCE).

SR is built over already existing IGP and takes advantage of their features.

- A **Prefix-SID** is a segment that refers to a specific network prefix. Prefix-SIDs are always global within an IGP domain and refer to the shortest path computed



```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Header   |  Hdr Ext Len  | Routing Type  | Segments Left |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| First Segment |              Flags            |  HMAC Key ID  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|            Segment List[0] (128 bits ipv6 address)            |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                             ...                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|            Segment List[n] (128 bits ipv6 address)            |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|            Policy List[0] (128 bits ipv6 address)             |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|            Policy List[1] (128 bits ipv6 address)             |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|            Policy List[2] (128 bits ipv6 address)             |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
//                             HMAC                            //
//                   (256 bits, optional)                      //
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2.3: *Segment Routing in IPv6 (Adapted from [35]).*

by IGP to the related prefixes. A packet that enters an IGP area with an active Prefix-SID will be forwarded along the ECMP-aware shortest path to the prefix. Since a prefix could represent a node or a group of nodes within an IGP domain, Prefix-SIDs are further divided into Node-SIDs and Anycast-SIDs:

  - **Node-SID** refers to a specific node. A Node-SID has a global significance, and it identifies exactly the prefix of the node's loopback interface.
  - **Anycast-SID** identifies a set of routers. A packet with Anycast-SID will be forwarded towards the closest node of the anycast set. The Anycast-SID is an interesting tool for traffic engineering, as it eases expressing macro traffic-engineering policies.

- An **Adjacency-SID** represents a local segment (interface) to a specific SR node. Each router assigns a locally significant segment ID for each of its IGP adjacencies.

Any SR path can be represented by a combination of node SIDs and adjacency SIDs. Furthermore, SR supports Equal Cost Multi-Path (ECMP) [54] by design. Paths identified by node segments are IGP shortest-paths, and intrinsically include all the ECMP paths to the destination node, which contributes to a tremendous gain in network performance and scalability. Another essential IGP feature on which SR relies is the automatic rerouting of connections after a link failure. Upon a link failure, the IGP protocol recomputes all shortest-paths, and segments are automatically repaired without any additional intervention. The time required to detect a link failure, propagate the fault and recompute the shortest-paths, however, can be excessively long, and therefore recovery paths need no be pre-computed and installed in the data plane. Although fast reroute (FRR) with loop-free alternates (LFA) [11] follows this strategy and provides sub 50msec loss of connectivity to IGPs, it offers an incomplete coverage and is topology dependent. With SR, those limitations no longer exist. Topology-Independent LFA (TI-LFA) [14] provides local protection for IGP SIDs in any topology. Backup paths can be computed on a per IGP SID basis along the post-convergence path. In the vast majority of cases, a single segment is enough to encode the post-convergence path in a loop-free manner.

2.3 The Weight Setting Problem

The Internet is compounded by store and forward packet-switched networks where packets introduced by various hosts traverse communication links, briefly wait in router queues and reach their respective destinations. Routers, that forward packets without any priority or warranty of Quality of Service (QoS), offer a best-effort service model. One of the critical aspects of best-effort traffic forwarding is network congestion. In general, congestion occurs whenever the demand for network resources is greater than its available capacity, and packets are dropped due to overmuch queuing in the network. The underlying reasons for congestion are inappropriate resources provisioning or unbalanced distribution of traffic. Traffic load balancing is then a crucial traffic engineering concern as, if not operating correctly, it can cause some areas to become congested while others are underutilized. When congestion in the network increases beyond a certain threshold, the network performance degrades which translates into an increasing delivery delay and packets loss. Distinct sources compete for the same resources while being unaware of the network current state and other sources necessities leading in many situations to congestion collapse, that is, to a decrease in throughput. In order to cooperatively use the network to its full capacity, congestion avoidance mechanisms may pre-allocate resources while congestion control mechanisms manage the pace of traffic. However, and regardless of the installed congestion avoidance and control mechanisms, routing protocols which are responsible for the distribution of traffic on the available resources play a crucial role in assuring an appropriate network performance.

Link state routing protocols make traffic forwarding decisions based on the shortest paths defined from the installed link weights configuration. It would be expected that, to achieve a better usage of the available resources, links with a higher capacity should accommodate a higher volume of traffic. Such a result can be achieved by assigning to each link a weight inversely proportional to its capacity. This weight configuration strategy, *InvCap*, is adopted as a default by many vendors such as Cisco. However, though it might seem like a good configuration strategy, it is not the case. *InvCap* does not assure proper distribution of traffic over the network resources or provide a good traffic load balancing on multiple shortest paths. As a consequence, while some shortest path links may become congested other links may not be used at all. In this context, different strategies try to obtain link weights configurations that optimize particular objective functions.

2.3.1 Multi-commodity Flow Problem

The assignment of commodity flows from source to destination in a network is generally known as a multi-commodity flow problem (MCFP). The problems arise when several commodities share edges in a network and compete for the capacity on these edges [58]. Each commodity has a unique set of characteristics and the commodities are not interchangeable. That is, it is not possible to satisfy demand for one commodity with another commodity. A comprehensive survey of linear multi-commodity flow models and solution procedures are presented in [1].

All MCFPs must consider two necessary constraints. The first is the travel demand, which means that all the commodities need to be transported to their destinations. The second is the edge capacity constraint, that is the flow on each edge cannot exceed flow capacity. The first constraint is essentially the sum of a set of single-commodity flow problems. However, the second constraint needs to consider all commodities together, and it causes interactions between them.

There are several possible formulations for the MCFP, and we here adopt the *node-edge* or *conventional* formulation. A set of commodity flows K is to be assigned over a network represented as a directed graph $G\,(N, E)$ where N and E are the set of nodes and edges respectively. A MCFP contains decision variables x, where x_{ij}^k is the fraction of the total quantity of commodity $k \in K$, q^k, assigned to the edge ij. The cost of assigning such a fraction of the total commodity k to an edge ij equals $q^k x_{ij}^k$ times the unit flow cost for the edge ij, denoted as c_{ij}^k. All edges ij have, however, a limited capacity d_{ij} which can not be exceeded. The objective is to minimize the total cost

(Equation 2.1) while attending the defined constraints, that is,

$$\text{Minimize} \sum_{k \in K} \sum_{ij \in E} c_{ij}^k q^k x_{ij}^k \tag{2.1}$$

such that

$$\sum_{k \in K} q^k x_{ij}^k \leq d_{ij}, \forall ij \in E \tag{2.2}$$

$$x_{ij}^k \geq 0, \forall ij \in E \tag{2.3}$$

$$\sum_{ij \in E} x_{ij}^k - \sum_{ji \in E} x_{ij}^k = b_i^k, \forall i \in N, \forall k \in K \tag{2.4}$$

Equations 2.2 and 2.3 are, respectively, the edges capacity and non-negativity of commodity fractions constraints. Equation 2.4 is a flow conservation constraint. For each commodity flow k, if i is the supply node of commodity k, then b_i^k is equal to 1; if i is the destination node, then b_i^k is equal to -1. On remaining nodes, commodity k is neither produced or consumed and b_i^k is equal to 0.

The decision variables x define both paths and load balancing fractions for the commodity flow problem. In the particular case where x_{ij}^k variables are restricted to binary values,i.e., 0 or 1, commodities may use only a unique path from origin to destination. If arbitrary splitting of traffic flows is allowed, the multi-commodity flow problem can be solved optimally in polynomial time [30].

2.3.2 Link Utilization Cost Function

The weight setting problem for Link State routing protocols is a particular case of the MCFP. The problem can be formulated as: finding a link weights configuration that optimizes the network resources utilization, assuming a specific set of traffic demands.

Different approaches to the weight setting problem use distinct objective functions to evaluate the network resources utilization. Even for simple objective functions, the problem of optimizing the link weights is NP-hard [39]. The computational challenge arises because of the limited control over the splitting of traffic over multiple shortest path routes. A commonly used and intuitive objective function is the minimization of the Maximum Link Utilization (MLU) values. However, this metric is sensitive to individual bottleneck links, it suffers from being local and emphases on a single link, and such a bottleneck may be severe to avoid under any routing solution. A better approach would be to consider a network-wide objective of minimizing all links utilization and apply a cost function that penalizes solutions that have heavily-loaded links.

Bernard Fortz and Mikkel Thorup in [39] proposed such function which maps links' utilization into non-negative real values. This piece-wise linear convex function, Φ, penalizes links which are over-utilized and grows exponentially with the link's utilization, as can be observed by its plot for a generic link a (Figure 2.4). Such behavior models packets re-transmission cost due to packets' loss. When compared with other cost functions such as MLU, the cost function Φ has the advantage of reflecting in a single value the congestion on all network links rather than take as cost a single maximum value (MLU) and inflicts a heavy penalization on undesired links' utilization. Next, a formal definition of this function is introduced adopting the notation presented in the previous section.

Demands are modeled as a matrix D of aggregated traffic requirements between each source s and destination t, $D(s,t)$. The load of a link a, represented as $\ell(a)$, is the sum of all traffic flows $f_a^{(s,t)}$, with source and destination pair (s,t), which travel over a, that is,

$$\ell(a) = \sum_{(s,t)\in N\times N} f_a^{(s,t)} \tag{2.5}$$

The cost function $\Phi(\ell(a))$ evaluates, for each link $a \in E$, the utilization $u(a)$ of the link, where $u(a) = \ell(a)/c(a)$. For each link $a \in E$, Φ_a is a continuous function such that $\Phi_a(0) = 0$, that is, no penalization is applied on unused links, and which derivative is presented in equation 2.6.

$$\Phi_a' = \begin{cases} 1 & for \quad 0 \le u_a < \frac{1}{3} \\ 3 & for \quad \frac{1}{3} \le u_a < \frac{2}{3} \\ 10 & for \quad \frac{2}{3} \le u_a < \frac{9}{10} \\ 70 & for \quad \frac{9}{10} \le u_a < 1 \\ 500 & for \quad 1 \le u_a < \frac{11}{10} \\ 5000 & for \quad u_a \ge \frac{11}{10} \end{cases} \tag{2.6}$$

The overall network congestion cost Φ is the sum of all link costs, for all network

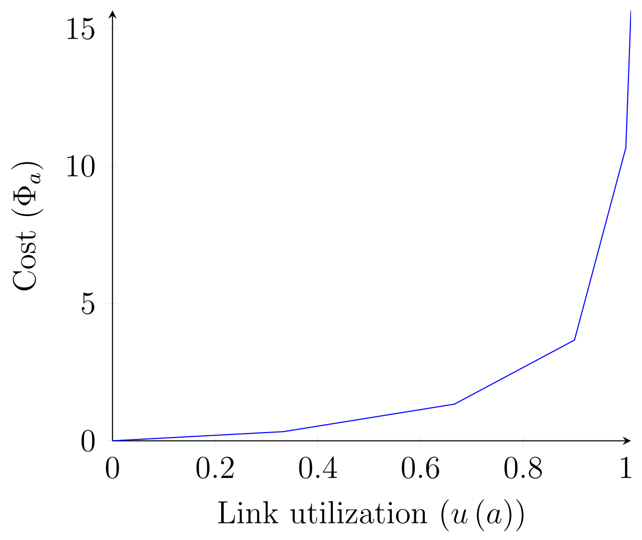


Figure 2.4: Φ_a *function plot.*

links a, Equation 2.7.

$$\Phi = \sum_{a \in A} \Phi_a (c(a)) \tag{2.7}$$

The values defined in Equation 2.6 have no other significance than to penalize increasing links utilization massively. Any continuous convex penalizing function with similar characteristics could be used instead of Φ_a. This cost function is widely accepted, enabling results comparison, which is a strong argument to its usage.

Integer Linear Programming Problem

The weight setting problem can be defined as an integer linear programming problem by performing some changes to the general MCFP definition.

- Objective Function:

$$\min \Phi = \sum_{a \in A} \Phi_a \tag{2.8}$$

- subject to:

$$\sum_{u:(u,v)\in A} f^{(s,t)}_{(u,v)} - \sum_{u:(v,u)\in E} f^{(s,t)}_{(v,u)} = \begin{cases} -D\left(s,t\right) & \text{if } v = s, \\ D\left(s,t\right) & \text{if } v = t, \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

$$\ell\left(a\right) = \sum_{(s,t)\in N\times N} f^{(s,t)}_{a} \tag{2.10}$$

$$\Phi_a \geq \ell\left(a\right) \tag{2.11}$$

$$\Phi_a \geq 3\ell\left(a\right) - \frac{2}{3}c\left(a\right) \tag{2.12}$$

$$\Phi_a \geq 10\ell\left(a\right) - \frac{16}{3}c\left(a\right) \tag{2.13}$$

$$\Phi_a \geq 70\ell\left(a\right) - \frac{178}{3}c\left(a\right) \tag{2.14}$$

$$\Phi_a \geq 500\ell\left(a\right) - \frac{1468}{3}c\left(a\right) \tag{2.15}$$

$$\Phi_a \geq 5000\ell\left(a\right) - \frac{19468}{3}c\left(a\right) \tag{2.16}$$

$$f^{(s,t)}_{a} \geq 0 \qquad a \in E; u, v, s, t \in N. \tag{2.17}$$

where Equation 2.9 is the flow conservation constraint, Equation 2.10 defines the load of link a, conditions Equations 2.11 to 2.16 are the cost applied to each link and Equation 2.17 assures that the amount of traffic in link a is non-negative.

Normalization of the Cost Function Φ

The cost function Φ when applied to different networks varying in size and links' capacity produces values on different scales. Consequently, to enable results comparison, the obtained values need to be normalized to a common scale. A normalization factor Φ_{Uncap} that enables the use of a common scale is defined by:

$$\Phi_{Uncap} = \sum_{(s,t)\in N\times N} \left(D\left(s,t\right) \times dist_1\left(s,t\right)\right) \tag{2.18}$$

where $dist_1$ is the shortest path distance between nodes when using unit link weight. In practice, the distance is the minimum number of hops between two nodes. Defining $\Phi_{unitOSPF}$ as the cost function Φ when all weights are unit, the following properties arise:

(i) Φ_{Uncap} is the total load when traffic traffic is forwarded across unit weight shortest paths.

(ii) $\Phi_{Uncap} = \Phi_{UnitOSPF}$ if all links possess unlimited capacity.

(iii) Φ_{Uncap} is the minimum network total load.

(iv) $\Phi_{Uncap} \leq \Phi_{OPT}$

(v) $\Phi_{UnitOSPF} \leq 5000.\Phi_{Uncap}$

If we denote the optimal solution of the problem by Φ_{OPT} and normalization as

$$\Phi^* = \Phi/\Phi_{Uncap} \tag{2.19}$$

these properties allow to obtain the following order relation:

$$1 \leq \Phi^*_{OPT} \leq \Phi^*_{OptOSPF} \leq \Phi^*_{UnitOSPF} \leq 5000 \tag{2.20}$$

When $\Phi^* = 1$, it can be concluded that all links loads are under 1/3 of their capacity. On the other hand, when all links have a load equal to the limit of their capacities, Φ^* is equal to 10 2/3. This latter value is considered as the limit of the acceptable operating region of the network. The advantage of the Φ^* cost function when compared to others congestion measures, such as MLU, is that the utilization ratio of all individual links is reflected in a unique real value.

2.4 Traffic Matrices

A Traffic Matrix (TM) is an abstract representation of the traffic volume flowing between sets of source and destination pairs in a network. Estimating the edge-to-edge TM in a network is an essential part of many network design and operation tasks such as capacity planning, routing protocol configuration, network provisioning, load balancing, and anomaly detection. The cost function Φ, presented in the previous section, is one example of how TMs are used to improve networks functioning conditions. However, a direct and precise measurement of TMs in large IP networks is extremely hard, if not unattainable, due to the large number of source-destination pairs, the high volume of traffic at each link, and the lack of a measurement infrastructure. To counter this limitation, estimated or generated traffic matrices are often used.

2.4.1 Traffic Matrix Estimation

The data gathered from direct measurements is sufficient, in most cases, to populate a traffic matrix using estimation techniques [69, 124]. Two main strategies are commonly used to infer TM: (1) indirectly from link loads [76]; (2) directly from sampled flow statistics [79]. Some other approaches take advantage of both strategies [126]. Indirect methods are sensitive to the statistical assumptions made in their models and are shown to have large errors. Direct methods can be quite attractive due to their high accuracy levels. However, the lack of required measurement infrastructure and the prohibitively large overhead imposed on the network components are two main drawbacks of direct measurements.

In this work we are particularly interested in flow-based methods to measure traffic statistics in SDN. OpenFlow switches, unlike commodity switches, provide a permissive query API that allow traffic measurements with a low overhead. Upon the arrival of a new flow or upon the expiration of a flow entry, *PacketIn* and *FlowRemoved* messages, respectively, are sent by OpenFlow switches to the controller, and thereby enable to compute the links utilization between switches. Exploration works of these features permitted new proposals for TM estimation, resulting in, as example, the OpenTM [114] and FlowSense [122] traffic estimation proposals for networks with OpenFlow capabilities.

2.4.2 Traffic Matrix Synthesis

Motivated by the lack of availability of real world traffic matrices, due to the proprietary nature of most traffic data, the synthesis of the traffic matrix has become an important area. Suprisingly, there is however a shortage of work in this subject. One of the fiew models to synthesise traffic matrices proposed in the literature is the Gravity model [77, 92]. It inherits its name from the Newton's law of gravitation which is commonly used by social scientists to model the movement of people, goods or information between geographic areas. There is also a simpler spatial model proposed in [39] which emphasis in geographical characteristics of traffic and its relationship with the topology.

In the present work we adopted this last proposal.

Two concepts characterize the model: first, close pairs of nodes exchange larger volumes of traffic, and second, the definition of an expected mean of congestion α in the the network. For each pair of nodes (s,t), $s \neq t$, the amount of traffic from s to t is modelled by Equations 2.21 and 2.22, where R is a random number in range $[0,1]$, $d_{s,t}$ is the Euclidean distance between both nodes, $\overline{c_a}$ is the average capacity of all links, $|E|$ is the number of links in the topology and $H_{s,t}$ the minimum number of hops between s and t. The use of the Euclidean distance in the formulation $D\left(s,t\right)$ enforces that close pairs of nodes have relatively more demand.

$$D\left(s,t\right) = \frac{R \times \delta}{d_{s,t}} \tag{2.21}$$

$$\delta = 2 \times \alpha \times \overline{c_a} \times |E| \times \sum\nolimits_{(s,t) \in N^2} \frac{H_{s,t}}{d_{s,t}} \tag{2.22}$$

2.5 Previous Contributions

The simplicity and popularity of link-state intradomain routing protocols has motivated the appearance of seminal research work involving TE efforts aiming to attain near optimal weight setting configurations (e.g. [30, 39]) for a given set of traffic demands, usually represented as a demand matrix. The results of such preliminary efforts have motivated several researchers to explore and improve those TE approaches. Furthermore, advances in traffic estimation techniques and the availability of tools within such purposes opened the opportunity for such theoretical approaches to be effectively applied to some real network environments. Several studies highlighted the advantages of enhanced TE aware configurations over traditional heuristics usually adopted by administrators [101], and their use in multi-constrained TE optimization contexts involving several QoS related constraints [91, 98].

Despite proving the efficiency of the optimization processes, many proposals still presented some limitations, usually assuming static optimization conditions, often not considering possible changes in the demand matrix assumptions, and not attaining sufficiently resilient aware configurations to be used in real networking scenarios. Even thought changing the weight configuration could respond to those changes, network administrators do not consider this solution, at least on a regular basis, as such changes could result in network instability due the distributed nature of the routing protocols and required convergence time. Moreover, the performance of some transport level protocol connections may be degraded due to the arrival of packets in disorder while the network remains unstable. Therefore any solution able to tackle this problem should change the least possible weights, trying to preserve, for as long as possible, previous installed configurations and flow paths. Minimizing the number of weight changes has been initially studied in [40] employing local search. In [80] we showed that by seeding Evolutionary Algorithm initial population with the installed weight setting, new optimized solutions would maintain more than 85% of the shortest paths.

Instead of changing link weights settings to respond to changes, some alternative approaches consider setting resilient link weights configurations. Traffic follows quite regular periods with a peak in the day and the evening [7] and different traffic requirements. More generally let us assume that the uncertainty in the network traffic can be approximated by a set of demand matrices, where each matrix represents a possible scenario of traffic. A link weights configuration that simultaneously minimizes the congestion on all scenarios will provide the network with the necessary resilience to traffic changes. This idea was initially proposed in [40] where weights are optimized using a local search algorithm considering two distinct traffic matrices. Additionally, the authors showed that the obtained solutions were also adequate for noisy traffic demand matrices derived from the traffic matrices used during the optimization process. In [80] the same problem was explored using multi-objective evolutionary algorithms. Instead of making available a single solution to network administrators, the last has the advantage to provide a set of possible solutions which embody different trade-offs for the network temporal moment congestion. In [6] the authors proposed a mixed integer programming model to optimize the link weight metric for polyhedral demands, that is, a single polyhedral demand matrix is used to model a set of k different traffic necessities. With such approach, however, there is a significant compromise on congestion when the network resources utilization is evaluated for each k different traffic necessities.

In addition to traffic demands variations, topology changes, such as links failure, alter the conditions for which the link weights were optimized. After a link failure, all traffic that was traveling over the failing link is rerouted to new IGP pos-convergence shortest paths and, consequently, congestion may occur in parts of the network. To address this problem, and aiming to find a link weights configuration that allows the network to continue operating properly after a link failure, the authors in [41] proposed an objective function which aggregates the network congestion in a normal state and after the failure of a single link. To accommodate all possible single link failures, the last is the average of the network congestion values for the failure of all links, one at a time. As the failure of each link does not reflect equally on the network congestion, for example, the failure of unused links has no impact, such approach can lead to solutions which do not perform well as the impact on the objective function of more significant links is diluted. Furthermore, a link failure may not occur, and the use of an aggregated function gives equal importance to the network congestion values in both states, in a normal and in failing state, which penalizes the fully functional network performance. A weighting scheme may be used to leverage the importance of each objective, but such a trade-off is not always possible to identify before the optimization. To address the identified issues, we proposed in [80] a multi-objective approach that only considers the failure of the most relevant links and offers a set of solutions with distinct trade-offs between the objectives.

The link-state weight setting problem has produced several distinct approaches over the years, but they all consider, in some way, predictable changes on the networking environment. Unforeseen changes continue to undermine the operational performance of IP networks and must be addressed with different tools and methods. The advent of more complex networking infrastructures (e.g. with ISPs also incorporating value-added services, Data Center storage points, etc.) has fostered the need and opportunity for exploring new TE methods. The Software-Defined Networking and Segment Routing paradigms provide an abstraction and modularity which can be taken as a base-ground to develop new tools and methods to address those type of problems.

2.6 Conclusions

Intradomain routing is at the heart of today's Internet and is in large part responsible for its performance and quality of service. Intradomain routing deals with two main objectives: firstly, make domain forwarding paths decisions so that traffic is delivered to its destination while meeting users constraints and improving ISP benefits; secondly, allow to quickly recover from link and router failures and adapt to new traffic necessities. To accomplish those objectives, an intradomain routing solution is required to hold some fundamentals properties: scalability, efficient resource utilization, easy setup and management, robustness with enough flexibility to adapt to changes and provide low stretch and diversified paths.

In this chapter, we discussed the properties of the leading solutions currently used for intradomain routing: OSPF/IS-IS, MPLS, SDN, and the more recent SR approach. Each of these solutions checks for some of these properties but fails on others. Link-state protocols, such as OSPF and IS-IS, although scalable and easy to set up, are not sufficiently flexible. MPLS, on the other hand, is highly flexible but can be very complicated to set up. SDN, with a centralized control of intradomain routing, is even more flexible than MPLS but presents scalability and over complexity issues. SR aims to combine the advantages of link-state routing with the flexibility of MPLS with an SDN centralized control. In such a context, the role of TE is to couple the available solutions to meet all desired properties.

Chapter 3

Multi-Objective Optimization of Link State Routing

Proper use of resources in networks that perform traditional link state routing protocols requires the optimization of a set of weights associated with each topology link. When traffic requirements are known, the optimization problem is NP-hard and thus meta-heuristics such as Evolutionary Algorithms (EAs) can be used to obtain optimized configurations. In this context, this chapter addresses the evaluation of three distinct EAs, a single and two multi-objective EAs, in two tasks related to weights setting optimization towards optimal intradomain routing. The chapter starts by introducing the basic principles of multi-objective EAs and describes the characteristics of the three algorithms, namely, the Non-dominated Sorting Genetic Algorithm (NSGA-II), the Strength Pareto Evolutionary Algorithm (SPEA2) and a Single-Objective Evolutionary Algorithm (SOEA) using weighted-sum aggregation. The chapter continues with a comparison of the three optimization algorithms in problems that consider dynamic alterations to the network state. The first problem considers variations in traffic requirements and the latter considers the possibility of link failures. The optimization tasks need to simultaneously optimize for both conditions, the normal and the altered one, following a preventive Traffic Engineering approach towards robust configurations. Since this can be formulated as a bi-objective function, the use of multi-objective EAs, such as SPEA2 and NSGA-II, came naturally, being those compared to a single-objective EA.

3.1 Introduction

Changes in traffic demands and link failures are dynamic conditions that undermine the operational performance of a network. Traffic demands undergo periodic changes during specific periods of time, such as night and day, that impact the congestion levels of the network. To address those changes, network administrators can perform alterations to the installed weights to redefine the distribution of traffic. However, such changes can introduce temporary problems. Weight changes create a transient instability on the traffic flow due to the distributed nature and convergence time of commonly used routing protocols. Furthermore, changes in traffic paths disrupt the performance of higher level protocols, such as the Transmission Control Protocol (TCP), whose connections may become degraded by out of order packet delivery.

There are also considerations to be made when reconfiguring weights in response to link failures. The majority of link failures are single link failures, and, usually, they last for a short amount of time [55]. Frequent reconfiguration of link weights is thereby not considered a proper approach to the problem.

A more appealing solution to these problems consists in finding a single weight setting that would allow the network to maintain a good performance level against such events, that is, find a weight configuration that continues to provide a good distribution of traffic after a link failure or in case of foreseen changes of traffic demands. These constitute multi-objective optimization problems for which meta-heuristics can provide good solutions.

Evolutionary Computation (EC) algorithms are nature inspired methods. Mainly used to solve optimization problems, they maintain a population of solutions and the interaction between these points drives the optimization process across a search space. Although the underlying mechanisms are simple, Evolutionary Computation algorithms have proven themselves as a general, robust and powerful search mechanisms that can offer solutions to problems which can not be treated, or with difficulty, by other optimization methods. Most closed optimization methods depend on asserted assumptions. For example, gradient descent based methods require the ability to compute the derivative of objective functions. Other methods, such as Linear Programming, are only able to address single objective problems and often require a relaxation by removing the integrality constraint of each variable. Furthermore, many real problems are non linear and approximating non linear functions with linear functions may not be really satisfying. Evolutionary Computation algorithms possess several characteristics that are desirable for problem solving. They are capable of handling multiple conflicting objectives in intractably large and highly complex search spaces. In combinatorial optimization, in problems where the search space is vast, Evolutionary Algorithms (EAs) [22] try to find an optimal solution from a finite set of solutions. Solutions are evaluated via a fitness function, which determines how well they solve the particular problem instance. In analogy with the Darwinian natural selection, the fittest individuals survive and evolve to produce more adapted solutions. By focusing on the best members of the population, and introducing small variations (mutation) and mating (crossover) operations, it is expected that the population evolves toward good, or even optimal, solutions within a reasonable time.

Real-world optimization problems are often characterized not by a single objective, but by a set of criteria against which candidate solutions must be assessed. While in an uni-objective context to "optimize" is a well-understood task, meaning to find the extremum of the objective' function, the same cannot be said of the multi-objective problems. Objectives on a multi-objective problem are frequently conflicting, that is, a solution that improves one of the objectives will eventually degrade at least one of the others. Consequently, there is no single global solution, and it is necessary to determine a set of optimal points, a Pareto Set, which populate a Pareto Front (PF), Figure 3.1. Generating the Pareto set can be computationally expensive and is often unfeasible because the complexity of the underlying application prevents exact methods from being applied. For this reason, a number of stochastic search strategies such as Evolutionary Algorithms, Tabu Search [44], Simulated Annealing [61], and Ant Colony Optimization [29] have been developed: they usually do not guarantee to identify optimal trade-offs but try to find a good approximation, a set of solutions whose objective vectors are, hopefully, not too far away from the optimal objective vectors.

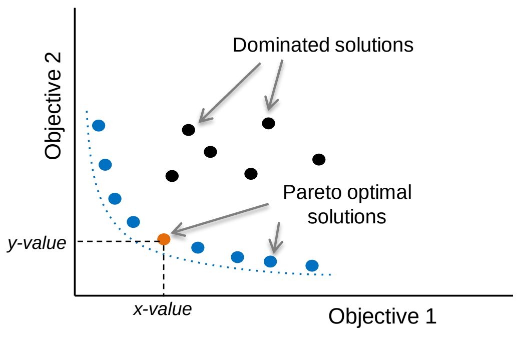


Figure 3.1: *Illustration of the Pareto Front concept.*

3.2 Basic Principles of Multi-Objective Evolutionary Algorithms

Suggested in the late 1980s, Multi-objective Evolutionary Algorithms (MOEAs) [97] are now commonly used to solve optimization problems that need to consider multiple objectives simultaneously and seek to optimize the components of a vector-valued cost function. Typically, MOEAs use the concept of dominance in the fitness assignment. This idea, initially introduced into EAs by Goldberg [45], has the main advantage of not requiring the transformation of the multi-objective problem into a single objective one. Furthermore, they can generate in a single run a set of diversified solutions which lie in the Pareto optimal, Figure 3.1. Each point in this surface is optimal in the sense that no improvement can be achieved in one cost vector component that does not lead to degradation in at least one of the remaining components. This gives rise to the concept of Pareto dominance where a solution is said to dominate another if it is not worse than the other in all objectives and, simultaneously, it is strictly better than the other in at least one objective.

There are two important goals, possibly conflicting, in dominance-based approaches, convergence, and diversity. Convergence refers to the ability to find a (finite) set of solutions lying on the Pareto-optimal front, while diversity refers to the heterogeneity of such solutions, which should cover the entire range of the Pareto-optimal front. Most MOEAs try to maintain diversity within the current Pareto set approximation by incorporating density information into the selection process. The probability of selecting a particular individual decreases with a higher density of individuals in its neighborhood. This issue is closely related to the estimation of the probability density functions in statistics.

While the distance to the optimal front is to be minimized, the diversity of the generated solutions is to be maximized. Different MOEAs however, provide non-dominated solutions with distinct convergence and diversity for distinct MOPs. To improve MOEAs diversity and convergence, an additional concept of elitism became very popular since the 1990s. Elitism addresses the problem of losing good solutions during the optimization process due to random effects. One way to deal with this problem consists in maintaining an external set, called archive, that allows storing all the non-dominated or the most preferred solutions found during the search. This archive mainly aims at preventing these solutions from being lost during the optimization process. Alternatively, the old population and the offspring, that is, the mating pool after variation, are combined by applying a deterministic selection procedure instead of replacing the old population by the modified mating pool.

Since Schaffer et al. introduced the Vector Evaluated Genetic Algorithm (VEGA) to solve multi-objective problems [96], several other MOEAs emerged. Some representative examples include algorithms as the PAES by Knowles and Corne [62], SPEA [128] and SPEA2 by Zitzler et al. [127], NSGA [104] and NSGA-II by Deb et al. [104], MOPSO by Coello, Pulido, and Lechuga [23], PESA-II by Corne, Jerram, Knowles and Oates [24], among many more. They all propose distinct strategies to maintain elitism by defining variants criteria to which the solutions to be kept are selected (e.g., dominance criterion and density estimation). As a consequence, distinct MOEAs have different merits and demerits and, consequently, may not offer equally good solutions for distinct problems.

3.2.1 Multi-objective Generic Problem

Before discussing particular problems and algorithms for multi-objective congestion optimization, we define a generic problem that involves multiple conflicting objectives. A multi-objective optimization problem involves more than one objective function that need to be either minimized or maximized, subject to constraints and variable bounds. A MOP, that involves m optimization objectives, can, thus, be formalized as:

$$\begin{array}{rll} \text{Minimize } \mathbf{F}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_m(\mathbf{x})]^T; & & \\ \text{subject to} \quad g_j(\mathbf{x}) \leq 0, & j = 1, ..., J; & \\ x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1, .., n. & \end{array} \tag{3.1}$$

where a feasible solution $\mathbf{x} \in X^n$ is a vector of n decision variables, $\mathbf{x} = (x_1, x_2, ..., x_n)$. The feasible set of solutions in the decision space is defined by the constrains $g_j(x) \leq 0$, and by the variable bounds. The multi-objective function vector $\mathbf{F} \in X^n \times Y^m$, where $f_i : X^n \to Y$, $i = 1..m$, evaluates the quality of a specific solution by assigning it an objective vector $(y_1, y_2, ..., y_m)$ in the objective space Y^m, Figure 3.2.

As an example, let us suppose that the objective space is a subset of the real numbers, that is, $Y \subset \mathbb{R}$ and that the goal of the optimization is to minimize a single objective $(m = 1)$. In such a single-objective optimization problem, a solution $x1 \in X^n$ is better than another solution $x2 \in X^n$ if $y1 < y2$ where $y1 = F(x1)$ and $y2 = F(x2)$. Although several optimal solutions may exist in decision space, they are all mapped to the same objective value, i.e., there exists only a single optimum in objective space.

In the case of a vector-valued evaluation function F with $Y^m \subset \mathbb{R}^m$ and $m > 1$, the situation of comparing two solutions $x1$ and $x2$ is more complex. Following the concept

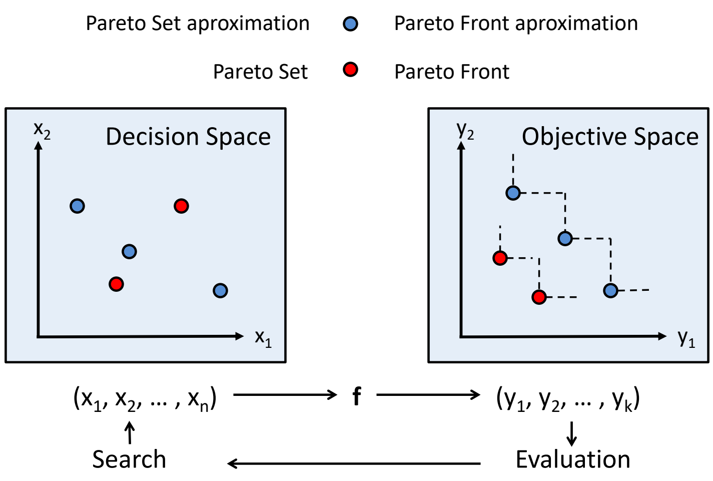


Figure 3.2: *Illustration of a generic multi-objective problem.*

of Pareto dominance, an objective vector $y1$ is said to dominate another objective vector $y2$ ($y1 \prec y2$) if no component of $y1$ is greater than the corresponding component of $y2$ and at least one its components is smaller. Accordingly, we can say that a solution $x1$ is better than another solution $x2$, i.e., $x1$ dominates $x2$ ($x1 \prec x2$), if $y1 = F(x1)$ dominates $y2 = F(x2)$, that is ($y1 \prec y2$). Here, optimal solutions, i.e., solutions not dominated by any other solution, may be mapped to different objective vectors. In other words, there may exist several optimal objective vectors representing different trade-offs between the objectives.

A general stochastic search algorithm consists of three parts: i) a working memory that contains the currently considered solution candidates, ii) a selection module, and iii) a variation module. Selection procedures are classified as mating and environmental selection. Mating selection aims at picking promising solutions for variation and usually is performed in a stochastic fashion. In contrast, environmental selection determines which of the previously stored solutions and the newly created ones are kept in the internal memory. The variation module takes a set of solutions and systematically or randomly modifies these solutions in order to generate new solutions. One iteration of a stochastic optimizer includes the consecutive steps of mating selection, variation, and environmental selection; this cycle is repeated until a certain stopping criterion is fulfilled (Figure 3.3).

3.2.2 Representations

In 1886, Gregor Mendel realized that nature distinguishes between an individual's genetic code and its outward appearance. The genotype represents all the information

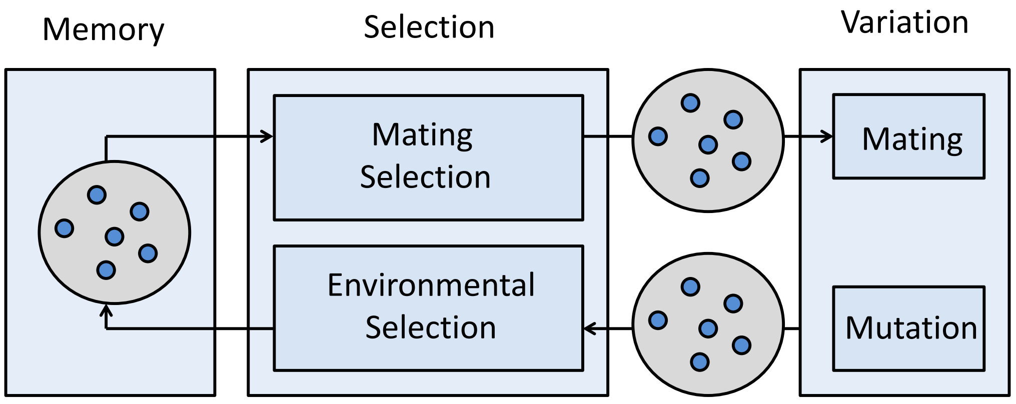


Figure 3.3: *Components of a generic multi-objective problem.*

stored in the genome and allows to describe an individual at the level of the genes. The phenotype describes the external appearance of an individual and all physical features. There is a mapping, or representation, genotype-phenotype, which uses the genotypic information to construct the phenotype. The genotypic information is not stored inside the alleles, but in the genetic allele sequence. By interpreting the allele sequence, nature can encode a large number of different phenotypic expressions using only a few different types of alleles.

When speaking of individuals in a population, we must carefully distinguish between genotypes and phenotypes. The phenotype of an individual determines its success in life. Thus, when comparing the capacity of different individuals, it is necessary to evaluate at the level of the phenotype. However, when it comes to reproduction, we should look at individuals at the genotype level.

Identifying the appropriate method for encoding solutions in chromosomes is a vital issue in the use of EAs. This problem has been investigated in many respects such as character mapping from genotypic space to phenotypic space when individuals are decoded into solutions as well as metamorphosis properties when individuals are manipulated by genetic operators. Binary coding was one of the first methods used. However, nowadays it is known to have serious drawbacks due to the existence of *Hamming cliffs*, encoding pairs that have a large Hamming distance[1] but which belong to points of minimum distance in space phenotype. For example, pair 0111111111 and 1000000000 belong to neighboring points in the phenotype space (minimum Euclidean distance) but have a maximum Hamming distance in the genotype space. In recent years, various coding methods have been created to achieve effective implementations of evolution-

[1]The Hamming distance between two strings of equal length is the number of positions where the corresponding symbols are different.

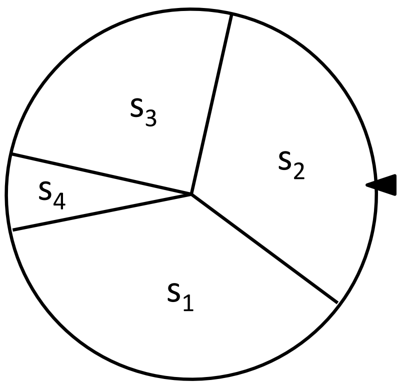
ary algorithms, and that adapt to the specificity of the problems. According to the notation used as alleles of a gene, the encoding methods can be classified as Binary Encoding, Real or Floating-point Encoding, Integer Encoding, Literal Permutation Encoding, Structure Encoding (for example in a tree), Hybrid Encoding among many others.

The application of EAs to solving any optimization problem always starts with the definition of the coding or representation method. In the context of this work, we used a representation by integers, with direct correspondence to the OSPF weights, being maintained the lower and upper limits.

3.2.3 Selection Mechanism

A selection mechanism in EAs is simply a process that favors the selection of better individuals in the population for the mating pool. They are the EAs' driving force and are applied with different pressure during the evolutionary process. The selection pressure is the degree to which the better individuals are favored: the higher the select ion pressure, the more the better individuals are favored. Typically, at the beginning of a genetic search, a lower selection pressure is applied. Individuals with poorer fitness can be selected enabling a spreader exploration of the search space. At the end of the search process, it is recommended to reduce the search space by gradually reducing the fitness dispersion values of selected individuals. The selection mechanism should direct the search to more promising regions. The most common types of selection are:

- Roulette wheel selection: The process consists of selecting individuals stochastically from one generation to create the foundation of the next generation. Individuals with better fitness are more likely to be selected. This process reproduces a natural selection in which fitter individuals are more likely to contribute to the mating pool for a next generation. The probability of selection of an individual is proportional to his fitness (see Figure 3.4). Individuals with poorer fitness can still be selected as they may be useful to improve future generations.
- Deterministic selection: These are deterministic procedures that select the best individual.
- Tournament selection: This method randomly chooses a set of individuals, and from it, the best individual is selected.



Individual	Fitness	Probability
s_1	1.4	37%
s_2	3.1	30%
s_3	4.3	25%
s_4	8.2	8%

Figure 3.4: *Roulette wheel selection.*

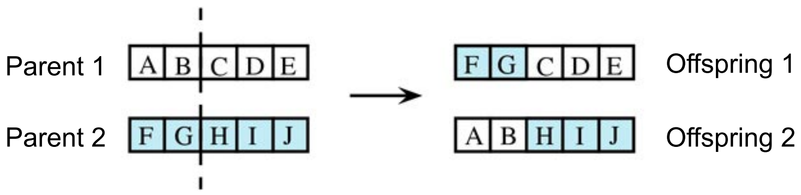


Figure 3.5: *One-Point crossover.*

3.2.4 Operators

Crossover or recombination operators are n-arity operators that receive two or more individuals and produce a new offspring with a combination of the parents genetic material. These operators have underpinned the idea that a new individual may be better than his relatives if he receives the best characteristics of their parents. Crossovers occur during the evolutionary process according to a user-defined probability and vary according to the representation. Some crossing operators for integer representation used in the devised work are described next:

- One-Point crossover: This operator randomly selects a crossover point and the heads (or tails) of its two parents are swapped to get new off-springs (see Figure 3.5).
- Two-Point crossover: Two crossover points are selected at random and alternating segments are swapped to get new off-springs (see Figure 3.6).
- Uniform crossover: Two parents are used to generate two offspring. For each position in the genome, a random binary variable is generated:
  - if the value of this variable is 1, the first descendant receives the gene of the first parent in that position, while the second parent receives the gene from the second parent in that position.

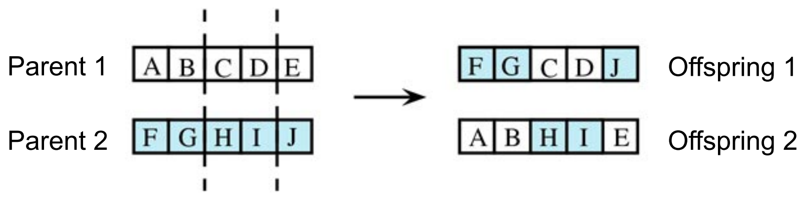


Figure 3.6: *Two-Point crossover.*

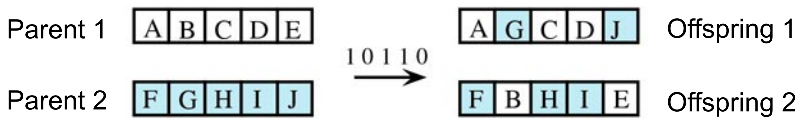


Figure 3.7: *Uniform crossover.*

– if the value of this variable is 0, the role of the parents is reversed. This operator is shown in Figure 3.7.

Some genetic operators have a single arity and are known as mutation operators. They take a single individual and apply a transformation over it and are used to maintain diversity from one generation to the next. Mutations are applied in the evolution process according to a defined probability, which is generally low. Through the use of mutations, one or more genes of an individual are changed stochastically, helping to avoid the stagnation of the population at a local optimum. Some examples of integer representation mutation operators are:

- Random Mutation: This is a simple mutation scheme. Given an individual $x^{(t)} = \left(x_1^{(t)}, ..., x_n^{(t)}\right)$ from a population P_t, a new value for a gene $x^{(t)}$ is randomly obtained $\left(x_k^{(L)}, x_k^{(U)}\right)$ with probability $1/\left(x_k^{(U)} - x_k^{(L)} + 1\right)$ from a discrete uniform distribution in range $\left(x_k^{(L)}, x_k^{(U)}\right)$, where $x_k^{(L)}$ e $x_k^{(U)}$ are receptively the minimum and maximum allowed values k $(k = 1, ..., n)$.
- Incremental/Decremental Mutation: This operator replaces a randomly selected gene by the successor or predecessor value, with equal probability, within the allowed values range.

3.3 Evolutionary Algorithms for Multi-Objective Problems

The no "free lunch" theorem for optimization [118] states that there is no single best algorithm for all optimization problems. Hence, algorithm selection and settings might involve trial and error for each distinct multi-objective optimization problem as those here explored. Testing all available MOEAs for the addressed problems is well beyond the scope of this work. After considering the available possibilities, also taking into account software availability, we selected two commonly used MOEAs, whose performance has been recognized by the community, the Non-dominated Sorting Genetic Algorithm (NSGA-II), the Strength Pareto Evolutionary Algorithm (SPEA2) and a Single-Objective Evolutionary Algorithm (SOEA) using weighted-sum aggregation.

3.3.1 Single-Objective Evolutionary Algorithms (SOEA)

A Single-Objective Evolutionary Algorithm works by updating the population in discrete iterations, called generations. It begins with an initial, generally randomly generated, population. This acts as a set of parents from which a number of offspring are produced and will integrate the next generation. The population size stays fixed at its initial size, which is a parameter of the algorithm. There exists some theoretical work investigating a good choice of population size in different situations, but there are few general principles [57]. A correct size is usually set by trial and error and depend on the optimization problem.

The critical points of a SOEA, which were previously discussed, are the selection method used to choose individuals from the current population, and the mutation and crossover methods used to generate new offspring. The idea is that selection will favor better solutions with a better fitness, that mutation will introduce slight variations in the current chosen solutions, and that crossover will combine together parts of different good solutions to, hopefully, form a better combination. The pseudo-code of a generic SOEA is presented in Algorithm 2.

Algorithm 2 Generic SOEA.

```
1: procedure SOEA($N, F(x)$)
2:     Initialize Population $P'_0$
3:     Evaluate the population against the objectives
4:     while not stopping condition do
5:         Parent Binary Tournament Selection
6:         Recombination and Mutation
7:         Select the survivors from the current population to be kept in the next
8:         Insert offspring in the next population
9:     end while
10: end procedure
```

Classical optimization methods suggest converting the multi-objective optimization problem to a single-objective optimization problem resorting to aggregation functions. The role of aggregation functions is to discriminate non-dominated points according to some preferences by emphasizing one particular Pareto-optimal solution at a time. More precisely, an aggregation function is a function which associates a unique real value to every point of the search space. The most widely used aggregate function is the weighted sum where a multi-objective optimization problem is transformed into a scalar optimization problem that can be addressed resorting to a single objective EA (SOEA). The corresponding aggregate optimization problem can be stated as:

$$\text{Minimize } \mathbf{F}(\mathbf{x}) = \sum_{i=1}^{M} w_i F_i(\mathbf{x}) \tag{3.2}$$

$$\sum_{i=1}^{M} w_i = 1 \tag{3.3}$$

where $w_i \geq 0$ are non-negative weighting coefficients and F_i are the different objective functions in the MOP.

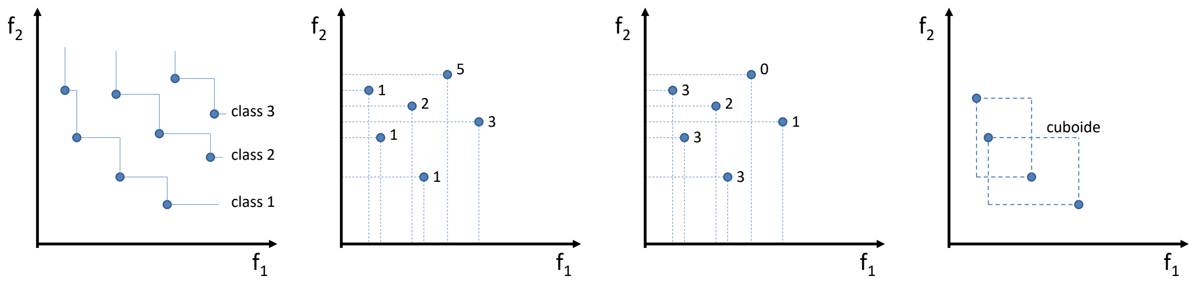
Although the use of an aggregation weighted function is computationally more efficient, it has some disadvantages. For every choice of the weights vector w, the problem yields to a unique optimal Pareto point and the weights vector w, which defines a search direction vector, needs to be firstly defined. When the aggregated objective functions are normalized, the w_i coefficients translate the relative importance given to each objective. Another weakness of this aggregate function is the failure to capture points on a concave Pareto front. Every point captured by Equation 3.2 is in a convex region of the non-dominated frontier. Additionally, because aggregation functions lead to a unique solution, in contrast to a set of Pareto optimal set, no following solution comparison, and subsequent choice can be made. Therefore, if such a method is to be used for finding multiple solutions, it has to be applied many times, with distinct weighs, hopefully finding a different solution at each simulation run.

3.3.2 Non-dominated Sorting Genetic Algorithm (NSGAII)

Multi-objective evolutionary algorithms that use non-dominated sorting and sharing were, for some time, criticized for their computational complexity, their non-elitism approach and the need to specify a sharing parameter. The Non-dominated Sorting Genetic Algorithm (NSGAII) [104] aims to address those issues. The NSGAII algorithm attempts to find multiple Pareto-optimal solutions in a multi-objective optimization problem while attending to the three main ideas: *1)* It uses an elitist principle; *2)* It uses an explicit diversity preserving mechanism; *3)* It emphasizes non-dominated solutions. In most aspects, this algorithm does not have many similarities with the original NSGA, but the authors kept the name NSGA-II to highlight its genesis and place of origin [27].

At every generation, the offspring population is firstly created by using genetic operators applied over the parent population. In this case, the reproduction operators will be the same defined above for the SOEA. The two solution sets are then combined to form a new population of size $2N$, from which an N dimension population is formed through selection based on dominance. A pseudo-code for NSGAII is presented in Algorithm 3.

The new population is filled with points from different non-dominated fronts, one at a time. The filling starts with the first non-dominated front (class 1) and continues with points of the second non-dominated front (class 2), and so on, as shown in Figure 3.8a. When the last allowed front is reached, and if not all members can be included in the new population, the points with highest crowding distance are chosen. The crowding distance d_i of point i is the average side-length of the cuboid formed by using as vertices the nearest neighbors in the front, Figure 3.8d. The crowding distance is therefore a measure of the objective space around a solution which is not occupied by any other solution in the population. By choosing the points with highest crowding distance it is possible to introduce a greater solution diversity in the new population.



(a) *Dominance depth* (b) *Dominance rank* (c) *Dominance count* (d) *Crowding distance*

Figure 3.8: *Dominance strategies and crowding distance.*

3.3.3 Strength Pareto Evolutionary Algorithm (SPEA2)

While NSGA-II uses a dominance depth sorting of solutions to fill the new population, the Strength Pareto Evolutionary Algorithm (SPEA2) resorts to dominance count and dominance rank sorting strategies on the partially ordered solution space. In

Algorithm 3 NSGA-II Algorithm.

```
1: procedure NSGA-II($N, f_k(x_k)$)
2:     Initialize Population $P'$
3:     Evaluate the population against the objectives
4:     Assign Rank based on Pareto Dominance (sort)
5:     Generate Child Population
6:           Parents Binary Tournament Selection
7:           Recombination and Mutation
8:     while not stopping condition do
9:         for each Parent and Child in Population do
10:            Assign Rank based on Pareto (sort)
11:            Generate sets of non-dominated vectors along $PF_{known}$
12:            Loop (inside) by adding solutions to next generation starting from the first front until $N$ individuals found determine crowding distance between points on each front
13:        end for
14:        Select points (elitist) on the lower front (with lower rank) and are outside a crowding distance
15:        Create next generation
16:              Parent Binary Tournament Selection
17:              Recombination and Mutation
18:    end while
19: end procedure
```

a dominance rank strategy, the rank associated with a solution is related to the number of solutions in the population that dominate the considered solution (Figure 3.8b), whereas the dominance count of a solution is related to the number of solutions dominated by it (Figure 3.8c). Both strategies are used to establish an order between the solutions. The single value fitness rank assigned to each solution evaluates the quality of a solution in relation to the whole population. The SPEA2 algorithm, Algorithm 4, when compared with SPEA, introduces improvements to the dominance fitness assignment scheme and incorporates a nearest neighbor density estimation and a new archive truncation method.

3.3.4 MOEA Comparison Metrics

The performance assessment by means of quantitative metrics is an important issue when comparing multi-objective optimization algorithms. The three optimization aims for a MOP consist of: i) minimal distance to the Pareto-optimal front, ii) good distribution, and iii) maximum spread. While the first is related to convergence, the remaining reflect the diversity of the obtained solutions. When comparing MOEAs,

Algorithm 4 SPEA2 Algorithm.

```
1: procedure SPEA2($N, E, f_k(x_k)$)
2:     Initialize Population $P'$ of size $N$
3:     Evaluate objective values
4:     Create an empty Archive $A$ with capacity $E$
5:     while not stopping condition do
6:         Evaluate the population $P'$ against the objectives
7:         for each solution in the Population and Archive do
8:             Calculate Raw Fitness
9:             Calculate Solution Density
10:            Compute Solution Fitness based on Raw Fitness an Density values
11:        end for
12:        Copy all non-dominated solutions from $P'$ and $A$ to the $A$
13:        if the capacity of $A$ has not been exceeded then
14:            use dominated individuals in $P'$ to fill $A$
15:        else
16:            use the truncation operator to remove elements from $A$.
17:        end if
18:        Create next generation
19:                Parent Binary Tournament Selection
20:                Recombination and Mutation
21:    end while
22: end procedure
```

those objectives should be somehow reflected in the applied metrics. From the cast set of available metrics, and considering that an SOEA is also compared, we selected the following comparison metrics:

- *C-measure*: It is based on the concept of solution dominance. Given two Pareto Fronts, PF1 and PF2, the measure C(PF1; PF2) returns the fraction of solutions in PF2 that are dominated by at least one solution in PF1. A value of 1 indicates that points in PF1 dominate all points in PF2, so values near 1 favor the method that generated PF1; values near 0 show that solutions in PF1 dominate few solutions in PF2.

- *Hypervolume*: It is the n-dimensional space that is contained by a set of points. It encapsulates in a single unary value a measure of the spread of the solutions along the Pareto front, as well as the closeness of the solutions to the Pareto-optimal front.

- *Trade-off analysis* (TOA): By defining different trade-offs between the objectives, F_1 and F_2, it allows to compare non-dominated solutions obtained from several multi-objective optimizers but also from traditional single-objective algorithms.

For a pareto front PF1, and given a value of β, the solution that minimizes the TOA is the value that minimizes $\beta \times F_1 + (1 - \beta) \times F_2$. Parameter β can take distinct values in the range $[0; 1]$ which correspond to different trade-offs.

The Pareto-optimal front is, in most problems, very difficult or impossible to identify. Therefore, the non-dominated solutions of all simulations in the same context, regardless of the algorithm, can be considered as an approximation for the Pareto-optimal front.

3.4 Link State Networks Traffic Variation Problem

The problem of finding a set of OSPF link weights that optimizes network performance usually only considers static conditions, where the network topology is set and traffic requirements are known and expressed as a single traffic matrix. The weight setting problem in such conditions can be formulated as given a directed network $G = (N, E)$, where N is the set of nodes and E is the set of edges, with capacity c_a for each $a \in E$, and a demand matrix D that, for each pair $(s, t) \in N \times N$, specifies the demand d_{st} in traffic flow between s and t, we want to determine a positive integer weight $w_a \in [1, 65535]$ for each edge $a \in E$ such that the objective function Φ^* (Section 2.3.2) is minimized.

The aforesaid formulation has been proven to deliver near optimal solutions [39], however, network conditions are not static. Traffic necessities vary over time and fault events, such as link or node failures, alter the network topology. Consequently, a weight configuration that is suited for a static environment may not be adequate in case of a change in operating conditions.

3.4.1 Problem Formulation

Traffic demands have temporal properties that significantly impact internet traffic engineering. The diversity of services available on contemporary networks, as well as human behaviors and habits, cause variations in traffic volumes and flow patterns not accommodated by traditional routing solutions. To acknowledge these variations, for example between two periods, such as night and day periods, we aim to find a link weights configuration that enables the network to sustain good functional performance in both periods. Thus, given two Traffic Matrices (TMs), D_1 and D_2, that represent the demands of two distinct periods, we want to find a link weights configuration w that simultaneously minimizes the objective functions Φ_1^* and Φ_2^*. Each Φ_i^* is the normalized cost functions Φ^* (Equation 2.20) that considers the traffic demands matrix D_i. The SOEA weighted-sum aggregation scheme for this set of experiments is expressed in Equation 3.4:

$$f(w) = \alpha \times \Phi_1^* + (1-\alpha) \times \Phi_2^*,\ \alpha \in [0;1] \tag{3.4}$$

A configuration suited for both matrices implies a compromise on the congestion level of each individual scenarios. Under the scheme Equation 3.4, an administrator can fine tune adjustments, such as favoring one of the matrices and penalizing the other, by setting the α parameter accordingly.

3.4.2 Experiments Setup

All the experiments were run on a publicly available optimization framework, NetOpt[2], that was extended in the context of this work. The framework aims to provide tools to optimize network operational performance by resorting to evolutionary computation methods. The optimization meta-heuristic algorithms are provided by a Java-based library, JEColi [31] with relevant extensions and alterations to fulfill the framework requirements. The general architecture of the framework is presented in Figure 3.9. In addition to the meta-heuristics library that provides the Genetic, NSGA-II and SPEA2 implementations, NetOpt also includes an OSPF routing simulator. Given a topology, a weight configuration (solution) and one or more traffic matrices, the OSPF simulator computes the routes between all nodes, using the Dijkstra shortest path algorithm, and accommodates the traffic demands accordingly onto the topology arcs. The solution can then be evaluated by applying the congestion function cost from Equation 2.20. The process is repeated for each solution.

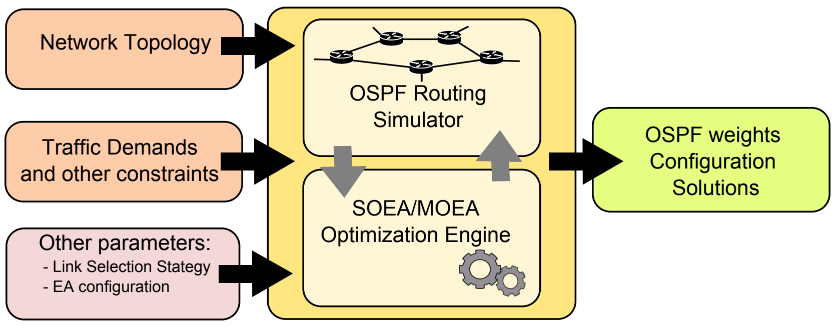

Network Topology
Traffic Demands and other constraints
Other parameters:
- Link Selection Stategy
- EA configuration
OSPF Routing Simulator
SOEA/MOEA Optimization Engine
OSPF weights Configuration Solutions

Figure 3.9: *General architecture of the optimization framework.*

[2]A more extensive description of the NetOpt framework is presented in Chapter 6

Simulation Case Studies

The simulations were run on two synthetic topologies, denoted as $Rand30_2$ and $Rand30_4$ (30 denotes the number of nodes, while the indexes 2 and 4 stand for the minimum in/out-degree of each node), as well as for a real backbone topology, the Abilene network topology [120]. The synthetic topologies were generated with the Brite topology generator [68], using a Barabasi-Albert model, with a heavy-tail distribution and an incremental grow type. The link capacities uniformly vary in the interval $[1; 10]$ Gbits. The characteristics of each topology are summarized in Table 3.1.

Table 3.1: *Synthetic and realistic network topologies.*

Name	Topology	Nodes	Links
Abilene	backbone	12	30
$Rand30_2$	random	30	110
$Rand30_4$	random	30	220

For each network, a set of three distinct traffic demand matrices D_i, $i \in \{0.3, 0.4, 0.5\}$, were used where i represents the expected mean of congestion in each link. Thus, larger values imply more difficult problems, as more traffic needs to be accommodated on the available resources. The set of demands D_i for the Abilene network were obtained by scaling Netflow data [21] publicly available and measured on March 1st, 2004 and September 1st, 2004. Traffic demands for the synthetic network topologies where randomly generated using the scheme defined in Section 2.4. In problems that have more than one traffic matrix as input, the linear correlation between them is approximately 0.5.

EA Configuration

The experiments were run with the same global configurations so that the results could directly be compared. Individuals are encoded as an integer vector of link weights, with length equal to the number of topology links, which take values in a user-configurable range $[w_{min}; w_{max}]$. In real implementations, OSPF link weights are valued from 1 to 65535, but here only values in the range [1; 20] were considered. A smaller range enables to reduce the search space and, simultaneously, increases the probability of finding equal cost multi-paths. The individuals that constitute the initial populations are randomly generated, with link weights drawn from a uniform distribution within the mentioned range.

To generate new individuals, the EAs use several reproduction operators, making possible to obtain new offspring through parents' recombination and mutation: Random mutation, Incremental/decremental mutation, Uniform crossover, and Two Point crossover (Section 3.2.4), all with the same probability of being applied.

The selection of mating progenitors as well as of individual that survive to integrate the new population is achieved using a Tournament selection mechanism, Section 3.2.3.

For the SOEA experiments, three trade-offs where considered, with α values in $\{0.25, 0.5, 0.75\}$. The population size was set to 100 individuals, and when applied, with an archive of the same size. Each simulation configuration was run 30 times with a stopping criterion of 1000 generations.

3.4.3 Results

The experimental results, in terms of TOA, for each of the three algorithms (SOEA, NSGA-II, and SPEA2), are summarized in Tables 3.2 and 3.3, which respectively present the best and the mean fitness values of all runs, organized by traffic demands levels and different values of the trade-off parameter (β). Congestion values need to be below 10 2/3 to be acceptable. For congestion values above this threshold, the network starts to endure heavy packet losses, due to the overutilization of some links, whereas a congestion value of 1 (optimal) translates into all links utilization being under 1/3 of their capacity.

The experiments with the $Rand30_4$ network topology only considered the traffic demand matrices with $D0.3$ level, as for higher levels of demand all obtained solutions have a congestion level that surpasses the threshold value of 10 2/3, above which the network ceases to operate acceptably. As the number of nodes and their degree increase, so does the optimization problem difficulty, and a higher number of iterations is required for the algorithms to converge.

The results for the Abilene topology show that all three algorithms were able to converge to the same best solution in at least one of the 30 simulations. This can be considered an easy problem due to the relatively small size of the topology. The mean fitness values, for all levels of demands and trade-offs for the Abilene topology, in Table 3.3, are also very similar among the three algorithms.

The performance metric C-measure, in Table 3.4, reinforces the conclusion that all algorithms perform equally well. The metric values are all of the same magnitudes and, consequently, no algorithm's PF set dominates the others. Although for the Abilene network topology, SOEA, NSGA-II, and SPEA2 provide equally good solutions, NSGA-II and SPEA2 are able to obtain, in a single run, a larger set of non-dominated solutions than the SOEA. The fact that the Abilene network only has 15 edges, which translates into solution vectors with 30 genes (two weights are assigned for each arc, one for each direction) leads to a reduced search space contributing to the fast convergence of all EAs to the same optimal solution.

Table 3.2: *Best fitness for two demand matrices optimization.*

Algorithm	First Demands	Second Demands	Abilene			$Rand30_2$			$Rand30_4$		
			$\beta = 0.25$	$\beta = 0.50$	$\beta = 0.75$	$\beta = 0.25$	$\beta = 0.50$	$\beta = 0.75$	$\beta = 0.25$	$\beta = 0.50$	$\beta = 0.75$
SOEA			1.14	1.16	1.18	1.37	1.39	1.40	2.50	2.49	2.47
NSGA-II	0.3	0.3	1.14	1.16	1.18	1.34	1.35	1.36	1.91	1.97	1.96
SPEA2			1.14	1.16	1.18	1.46	1.45	1.44	4.34	5.62	6.18
SOEA			1.45	1.37	1.28	1.75	1.64	1.53	-	-	-
NSGA-II	0.3	0.4	1.45	1.37	1.28	1.66	1.56	1.45	-	-	-
SPEA2			1.45	1.37	1.28	1.88	1.72	1.56	-	-	-
SOEA			1.52	1.52	1.52	1.95	1.99	2.02	-	-	-
NSGA-II	0.4	0.4	1.52	1.52	1.52	1.84	1.88	1.92	-	-	-
SPEA2			1.52	1.52	1.52	2.14	2.18	2.21	-	-	-

Table 3.3: *Mean fitness for two demand matrices optimization - Trade-off analysis.*

Algorithm	First Demands	Second Demands	Abilene			$Rand30_2$			$Rand30_4$		
			$\beta = 0.25$	$\beta = 0.50$	$\beta = 0.75$	$\beta = 0.25$	$\beta = 0.50$	$\beta = 0.75$	$\beta = 0.25$	$\beta = 0.50$	$\beta = 0.75$
SOEA			1.22	1.22	1.22	3.10	2.98	2.79	38.02	40.59	43.17
NSGA-II	0.3	0.3	1.21	1.21	1.21	1.93	1.83	1.73	4.31	4.44	4.58
SPEA2			1.21	1.21	1.21	2.19	2.06	1.94	43.98	46.92	49.86
SOEA			1.49	1.40	1.31	7.95	6.12	4.29	-	-	-
NSGA-II	0.3	0.4	1.48	1.39	1.30	3.54	2.86	2.18	-	-	-
SPEA2			1.48	1.39	1.30	4.55	3.62	2.69	-	-	-
SOEA			1.57	1.55	1.54	10.45	10.21	9.98	-	-	-
NSGA-II	0.4	0.4	1.56	1.55	1.54	2.92	2.80	2.68	-	-	-
SPEA2			1.56	1.55	1.54	5.40	5.27	5.14	-	-	-

For larger network topologies, with more nodes and edges, the performances of the three algorithms start to differ. The results for the synthetic topology $Rand30_2$, with 30 nodes and 55 edges (solutions with 110 genes), show that the NSGA-II algorithm attains better fitness values for every trade-off value (β) and demands level pair. This can be observed, for instance, with $D0.4$ traffic demand matrices and a trade-off weight β of 0.25, where the best and mean fitness values are, respectively, 1.951 and 10.446 (SOEA), 1.841 and 2.919 (NSGA-II), 2.139 and 5.401 (SPEA2). Although the NSGA-II and SOEA best values are relatively similar, and both better than SPEA2 results, the mean values for NSGA-II results are substantially better, and therefore, indicate a more robust algorithm able to obtain good solutions more consistently in the different runs.

The averaged C-measure metric values for the $Rand30_2$ network topology scenarios, in Table 3.4, show that the solutions in NSGA-II PFs dominate SPEA2 PFs, on average, in more than 56% of the cases, with a reverse C-measure of nearly 0%. When compared with SOEA PFs solutions, NSGA-II PFs solutions dominate approximately 16% of the SOEA FP solutions and, for the reverse case, SOEA PFs solutions dominate NSGA-II PFs solutions in roughly 6%. The NSGA-II algorithm, for the $Rand30_2$ topology scenarios, offered globally better solutions than any of the two other algorithms, while the SPEA2 algorithm had the worst performance of all.

Table 3.4: *Overall C-Measure for two traffic demand matrices optimization - Trade-off analysis.*

	Abilene			$Rand30_2$			$Rand30_4$		
	SOEA	NSGA-II	SPEA2	SOEA	NSGA-II	SPEA2	SOEA	NSGA-II	SPEA2
SOEA	-	0.143	0.179	-	0.062	0.553	-	0.029	0.602
NSGA-II	0.150	-	0.182	0.161	-	0.564	0.150	-	0.600
SPEA2	0.110	0.113	-	0.001	0.004	-	0.000	0.001	-

As the size of the topology increases, the performance of the NSGA-II algorithm detaches from the others. The experiments with the $Rand30_4$ network topology show that, while the best values of the three algorithms remain acceptable, NSGA-II features the best solutions. More importantly, considering mean results, it is the only algorithm whose fitness values remain within the acceptable operating limits of the network.

The C metric results for this new set of simulations are very similar to those obtained for the $Rand30_2$ topology, and again, NSGA-II PFs are globally better than those provided by the SOEA and SPEA2.

Table 3.5: *Mean normalized hypervolume for two traffic demand matrices optimization.*

Algorithm	**Topology**		
	Abilene	$Rand30_2$	$Rand30_4$
SOEA	1.000	0.127	0.004
NSGA-II	1.000	0.069	0.000
SPEA2	1.000	1.000	1.000

The hypervolume indicators, presented in Table 3.5, also support that NSGA-II is the best choice in the context of link-state weights setting for two traffic demand matrices. The PFs of NSGA-II are closer to the Pareto-optimal approximation than SOEA and SPEA2.

An illustrative representation of the best solutions obtained by each algorithm for the $Rand30_4$ network topology, with two $D0.3$ traffic matrices is presented in Figure 3.10.

3.5 The Problem of Single Link Failure in Link State Networks

3.5.1 Problem Formulation

Link failures on network topologies can occur for different reasons. At the physical layer, a fiber cut or a failure of optical equipment may cause a loss of physical con-

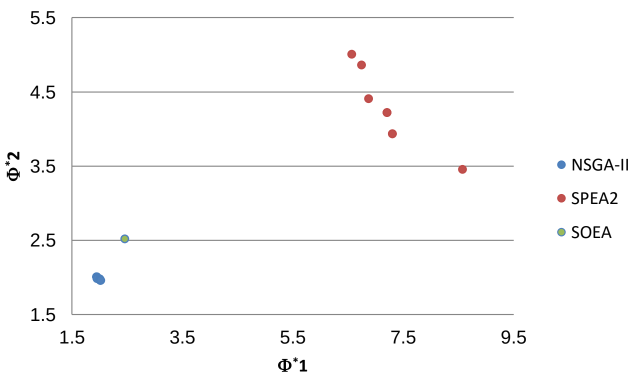


Figure 3.10: *Two demands congestion optimization. (Topology: Rand*30_4*, D_1: 0.3 D_2:0.3)*

nectivity. Other failures may be related to hardware, such as line card failures. Router processor overloads, software errors, protocol implementation, and misconfiguration errors may also lead to loss of connectivity between routers [107].

Failures may also vary in nature. They can be due to scheduled network maintenance or unplanned. Although backbone networks are usually well planned and adequately provisioned, link failures may still occur and undermine their operational performance.

Several mechanisms can be used to protect an IP network against link failures, such as overlay protection or Multi-Protocol Label Switching (MPLS) fast re-route [78], but protecting all links remains a very difficult task, or even impossible, especially for large network topologies. Thus, protection against failure continues to be link based.

As it is not possible to create scenarios where the optimization is done to address the failure of all links simultaneously, we assumed in this work that the optimization will be done for a scenario with no failures and for an alternative scenario where a selected link fails, thus defining two objective functions.

For a given network topology with n links and a traffic demands matrix D, the aim is to find a set of weights w that simultaneously minimizes the function Φ^*_n, representing the congestion cost of the network in the normal state, and Φ^*_{n-1}, representing the congestion cost of the network when a selected link has failed.

The single-objective version, considering a weighted-sum aggregation model, is provided by Equation 3.5:

$$f(w) = \alpha \times \Phi^*_n + (1-\alpha) \times \Phi^*_{n-1},\ \alpha \in [0;1] \tag{3.5}$$

An administrator is able to define a trade-off between the objectives by tuning the value of the α weight. When $\alpha = 1$, the optimization is only performed for the normal state topology, without any link failure, whereas when using $\alpha = 0.5$ the same level of importance is given to the two topology states. However, as the link failure optimization can compromise the network congestion level in a normal state, a network administrator may wish to focus on the performance of the normal state network, e.g. using a α value between 0.5 and 1, at the expense of the congestion level in a failed state, that may not occur.

Several criteria can be used to select the failing link, some are dynamic and depend on the solution that is being evaluated, while others are user pre-defined choices. In this study, we only considered two of the possible single link selection criteria that may configure severe cases of single link failure:

- *Highest Load*: The selected link, for each evaluated solution, is the one that has the highest load after the allocation of the traffic demands. Therefore, distinct solutions may have a different failing link.
- *Centrality*: The selected link in each topology is such that it occurs in the largest number of shortest paths, when assigning to each link a weight inversely proportional to its capacity (a commonly used weight setting strategy used, for instance, in OSPF Cisco implementations).

3.5.2 Optimization Results for the Failure of the Link with Highest Load

The failure of the network link that carries the highest traffic load configures one of the worst-case scenarios for the failure of a single link in a network. Its failure would translate into the rerouting of the highest amount of traffic which may lead to the congestion of some links.

Distinct levels of traffic demands, $D0.3$, $D0.4$ and $D0.5$ were used to compare the algorithms in problems with increasing difficulty. For comparison purposes, Table 3.6 that shows the obtained minimum weighted-sum aggregation fitness values, also includes the optimized congestion values for the networks without link failure optimization, and the respective congestion level after the failure of the link with the higher load (i.e., without any optimization for the future scenario). The single objective optimization was performed by the SOEA algorithm and run with the same previously presented configurations. On the other hand, the mean value, Table 3.7, provides a global assessment of each algorithm in all runs, and the best TOA value enables to compare each of the

Table 3.6: *Best fitness values for single link failure weights setting optimization - Highest Load Link.*

Topology	Demand	Without Link Failure Optimization		With Link Failure Optimization						Algorithm
				$\beta = 0.25$		$\beta = 0.5$		$\beta = 0.75$		
		Before	After	Before	After	Before	After	Before	After	
Abilene	0.3	1.20	1.76	1.29	1.23	1.29	1.23	1.23	1.33	NSGA-II
				1.34	1.21	1.33	1.22	1.24	1.35	SOEA
				1.29	1.23	1.29	1.23	1.23	1.34	SPEA2
	0.4	1.53	32.22	1.63	1.58	1.63	1.58	1.55	1.70	NSGA-II
				1.69	1.58	1.69	1.58	1.55	1.73	SOEA
				1.64	1.58	1.64	1.58	1.55	1.70	SPEA2
	0.5	1.91	309.48	2.14	1.91	2.14	1.91	2.05	2.17	NSGA-II
				2.26	1.93	2.26	1.93	2.26	1.93	SOEA
				2.14	1.91	2.14	1.91	2.04	2.14	SPEA2
$Rand30_2$	0.3	1.49	14.20	1.55	1.42	1.44	1.48	1.44	1.48	NSGA-II
				1.56	1.58	1.56	1.58	1.54	1.61	SOEA
				1.57	1.50	1.56	1.51	1.49	1.64	SPEA2
	0.4	1.79	41.44	1.83	1.76	1.75	1.80	1.75	1.80	NSGA-II
				2.07	2.09	1.85	2.22	1.85	2.22	SOEA
				1.95	1.93	1.91	1.96	1.91	1.96	SPEA2
	0.5	5.49	180.94	4.99	3.70	4.99	3.70	4.11	5.31	NSGA-II
				12.61	17.58	12.61	17.58	12.61	17.58	SOEA
				8.23	8.41	8.15	8.48	7.86	9.03	SPEA2
$Rand30_4$	0.3	3.67	73.69	2.38	2.20	2.30	2.25	2.10	2.59	NSGA-II
				11.14	7.91	11.14	7.91	6.04	13.64	SOEA
				59.48	29.64	28.95	47.39	28.95	47.39	SPEA2
	0.4	33.93	223.04	18.66	10.13	18.66	10.13	10.07	28.42	NSGA-II
				77.09	88.80	77.09	88.80	58.81	140.07	SOEA
				355.03	139.57	205.65	190.92	159.03	325.12	SPEA2

Table 3.7: *Mean fitness comparison for single link failure - Highest Load Link.*

Algorithm	**Demands**	**Abilene**			$Rand30_2$			$Rand30_4$		
		$\beta = 0.25$	$\beta = 0.50$	$\beta = 0.75$	$\beta = 0.25$	$\beta = 0.50$	$\beta = 0.75$	$\beta = 0.25$	$\beta = 0.50$	$\beta = 0.75$
SOEA	0.3	1.53	1.46	1.39	36.29	31.12	25.96	146.21	130.40	114.60
NSGA-II		1.37	1.34	1.30	5.76	6.64	7.51	23.07	30.95	38.83
SPEA2		1.37	1.33	1.30	8.10	11.05	14.00	128.35	135.72	143.10
SOEA	0.4	10.80	7.89	4.97	45.26	38.92	32.59	251.79	238.54	225.30
NSGA-II		3.24	2.69	2.14	11.26	12.35	13.44	74.61	79.91	85.21
SPEA2		3.40	2.79	2.19	12.29	13.00	13.71	417.23	420.31	423.38
SOEA	0.5	175.38	119.23	63.09	149.58	126.89	104,20	-	-	
NSGA-II		116.54	78.38	40.22	46.97	41.95	36.93	-	-	-
SPEA2		117.56	79.08	40.59	53.18	51.70	50.22	-	-	-

objectives fitness values, that is, the congestion values before and after the link failure.

The simulation results show that, for the smallest topology, Abilene, all three algorithms behave alike producing equally good solutions. However, as the topology size increases, or with the escalation of traffic requirements, NSGA-II is able to obtain solutions which translate into lower congestion values, both in the scenarios with or without a link failure, presenting the best solution for both objectives, as well as the best mean TOA congestion values. In some cases, NSGA-II was able to obtain better results for the first objective, when optimizing both states, than the ones provided by the SOEA when optimizing solely for the first objective, which is a surprising result. This occurred for the more difficult problems, such as the $Rand30_4$ network topology optimization experiments, Table 3.6.

In the $Rand30_4$ network topology scenario, with $D0.3$ traffic demands and $\beta = 0.5$, the fitness values before and after the link failure are, respectively in Table 3.6, 2.30 and 2.25 for NSGA-II; 11.14 and 7.91 for SOEA; 28.95 and 47.39 for SPEA2. The results allow observing that the more difficult the problem is, the greater the difference between the quality of the solutions produced by each of the three EAs. NSGA-II is able to outperform SOEA and SPEA in all scenarios. The SOEA algorithm low-quality results may be explained by its requirement of a higher number of generations to properly converge. The gap in performance between the algorithms is also mirrored in the averaged TOA, as shown in Table 3.7.

The C-measure values in Table 3.8 show that, despite being able to provide solutions with equivalent best fitness for the Abilene topology, the SOEA produces more solutions that are neither dominated by NSGA-II or SPEA2 solutions. In contrast, for the more demanding topologies, $Rand30_2$ and $Rand30_4$, NSGA-II PFs solutions dominate approximately 14% of the SOEA PF solutions, while the reverse is about 7%. When compared against SPEA2 PFs solutions, both NSGA-II and SOEA present better values.

Table 3.8: *C-measure of the highest Load link failure optimization.*

	Abilene			$Rand30_2$			$Rand30_4$		
	SOEA	NSGA-II	SPEA2	SOEA	NSGA-II	SPEA2	SOEA	NSGA-II	SPEA2
SOEA	-	0.173	0.242	-	0.071	0.702	-	0.056	0.887
NSGA-II	0.007	-	0.098	0.143	-	0.709	0.143	-	0.887
SPEA2	0.010	0.108	-	0.003	0.005	-	0.000	0.000	-

Concerning the hypervolume metric, in Table 3.9, NSGA-II performs slightly worse for the Abilene topology, but better for the other two networks. Figure 3.11 presents the best solutions for the single link failure congestion optimization when applied to the $Rand30_4$ topology for traffic necessities with 0.3 expected mean of congestion ($D0.3$). For the same experimental conditions, with the same number of iterations, the non-dominated solutions provided by NSGAII are significantly better than those provided by SPEA2 and the SOEA algorithms.

Table 3.9: *Mean normalized hypervolume for single link failure optimization*

Algorithm	**Topology**		
	Abilene	$Rand30_2$	$Rand30_4$
SOEA	0.667	0.180	0.048
NSGA-II	1.000	0.178	0.020
SPEA2	0.674	1.000	1.000

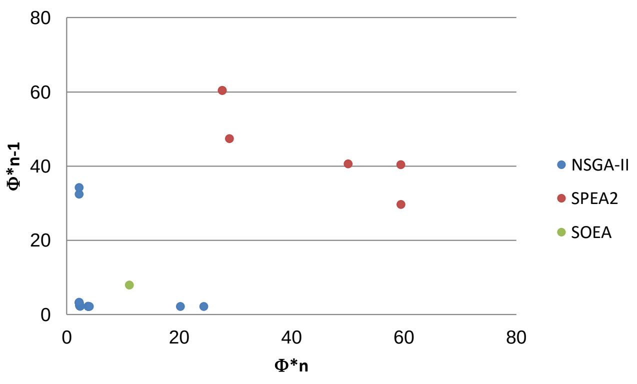


Figure 3.11: *Single link failure congestion optimization. (Topology: Rand*30_4*, D:0.3)*

3.5.3 Optimization Results for the Failure of the Link with Greatest Centrality

A network administrator can consider that a particular link is more crucial than others, because of its capacity, failure probability, or for any other reason. It is, therefore, important to enable an administrator to select the link that needs to be protected against failure. For this set of simulations, the selected link in each topology is such that it occurs in the largest number of shortest paths when assigning to each link a weight inversely proportional to its capacity. The minimal congestion values before and after the failure of the selected link, for distinct trade-offs ($\beta = 0.25, 0.5, 0.75$), are presented in Table 3.10, while the average congestion TOA values are included in Table 3.11.

The results of this new test suite consolidate previous observations, that is, for simpler problems, with smaller topologies and lower traffic demand levels, the SOEA, and MOEAs algorithms provide equally good solutions, but, as the number of nodes and links increases, or with the growth of traffic demands, NSGA-II is able to deliver better solutions, in the large majority of the tested scenarios, both before and after the link failure.

The C metric results, in Table 3.12, are also similar to those observed for the highest load link failure optimization. SOEA continues to have, in average, more solutions that are not dominated by any of the non-dominated sets of solutions resulting from NSGA-II and SPEA2 based optimization in the context of simpler problems. As the difficulty of the MOP increases, NSGA-II begins to provide better sets of non-dominated solutions.

Table 3.10: *Best fitness values for single link failure weights setting optimization - Centrality-based Link*

Topology	Demand	Without Link Failure Optimization		With Link Failure Optimization						Algorithm
				$\beta = 0.25$		$\beta = 0.5$		$\beta = 0.75$		
		Before	After	Before	After	Before	After	Before	After	
Abilene	0.3	1.20	1.76	1.23	1.73	1.20	1.74	1.20	1.74	NSGA-II
				1.22	1.71	1.21	1.72	1.20	1.74	SOEA
				1.24	1.72	1.20	1.74	1.20	1.74	SPEA2
	0.4	1.53	25.57	1.58	33.44	1.58	33.44	1.53	33.52	NSGA-II
				1.56	5.26	1.56	5.26	1.56	5.26	SOEA
				1.58	33.44	1.58	33.44	1.53	33.52	SPEA2
	0.5	1.91	309.48	1.97	119.30	1.95	119.32	1.93	119.34	NSGA-II
				1.98	281.63	1.98	281.63	1.98	281.63	SOEA
				2.01	119.29	1.95	119.32	1.93	119.34	SPEA2
$Rand30_2$	0.3	1.49	8.17	1.40	1.50	1.40	1.50	1.40	1.50	NSGA-II
				1.54	4.55	1.54	4.55	1.54	4.55	SOEA
				1.61	1.74	1.60	1.74	1.60	1.74	SPEA2
	0.4	1.79	58.65	1.76	1.93	1.76	1.93	1.75	1.93	NSGA-II
				2.10	3.40	2.10	3.40	2.10	3.40	SOEA
				2.18	2.50	2.18	2.50	2.18	2.50	SPEA2
	0.5	5.49	193.16	5.79	41.19	5.44	41.51	5.44	41.51	NSGA-II
				22.45	87.53	17.23	91.56	8.07	117.40	SOEA
				28.13	47.86	12.86	55.66	12.30	56.43	SPEA2
$Rand30_4$	0.3	3.67	117.13	2.20	2.29	2.19	2.29	2.18	2.33	NSGA-II
				5.81	33.01	5.81	33.01	5.81	33.01	SOEA
				204.29	219.19	204.29	219.19	204.29	219.19	SPEA2
	0.4	33.93	98.98	10.43	9.67	9.90	10.05	9.85	10.11	NSGA-II
				49.27	111.08	49.27	111.08	49.27	111.08	SOEA
				456.36	509.50	456.36	509.50	440.71	544.54	SPEA2

Table 3.11: *Mean fitness comparison for single link failure - Centrality-based Link.*

Algorithm	**Demands**	**Abilene**			$Rand30_2$			$Rand30_4$		
		$\beta = 0.25$	$\beta = 0.50$	$\beta = 0.75$	$\beta = 0.25$	$\beta = 0.50$	$\beta = 0.75$	$\beta = 0.25$	$\beta = 0.50$	$\beta = 0.75$
SOEA	0.3	1.63	1.49	1.36	55.57	37.61	19.65	165.95	126.13	86.31
NSGA-II		1.61	1.48	1.35	1.75	1.66	1.58	7.48	6.83	6.18
SPEA2		1.61	1.48	1.35	2.67	2.44	2.21	426.39	421.56	416.73
SOEA	0.4	19.45	13.49	7.53	74.27	50.65	27.03	165.95	126.13	86.31
NSGA-II		25.52	17.54	9.56	6.37	4.89	3.40	38.74	37.29	35.83
SPEA2		25.52	17.54	9.56	11.03	8.59	6.15	714.89	707.28	699.67
SOEA	0.5	227.12	152.07	77.02	178.50	129.88	81.26	-	-	-
NSGA-II		90.07	60.71	31.36	45.40	34.02	22.65	-	-	-
SPEA2		90.06	60.71	31.36	80.59	71.55	62.51	-	-	-

Table 3.12: *C-measure of Centrality-based Link failure optimization.*

	Abilene			$Rand30_2$			$Rand30_4$		
	SOEA	NSGA-II	SPEA2	SOEA	NSGA-II	SPEA2	SOEA	NSGA-II	SPEA2
SOEA	-	0.162	0.203	-	0.074	0.443	-	0.065	0.659
NSGA-II	0.008	-	0.063	0.122	-	0.468	0.163	-	0.657
SPEA2	0.024	0.125	-	0.014	0.019	-	0.000	0.001	-

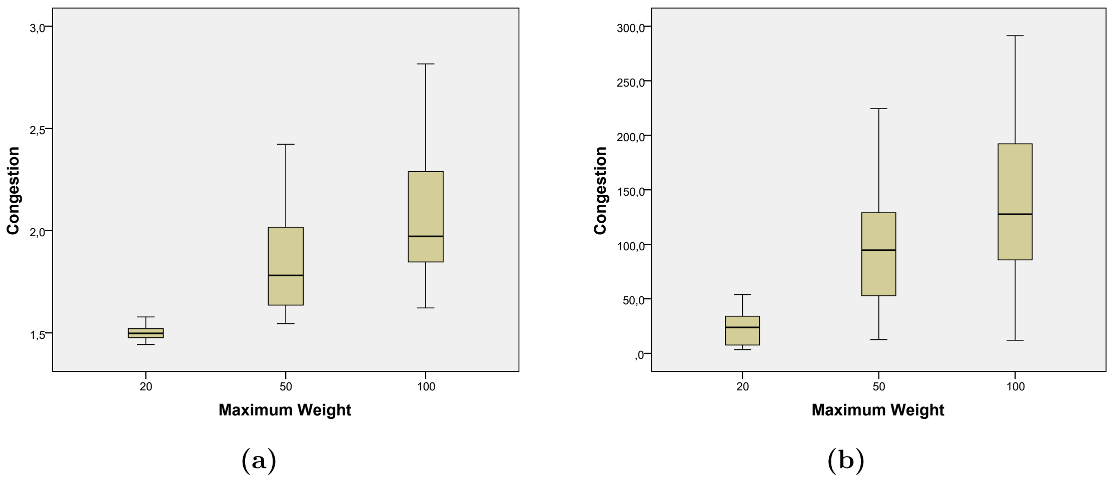


Figure 3.12: *Maximum weight influence in the congestion value. (a) Topology $Rand30_2$ (b) Topology $Rand30_4$*

3.6 Convergence of the Evolutionary Algorithms

3.6.1 Link Weights Range

As mentioned earlier, solutions are evaluated from a reduced search space, where weights take values within the $[1; 20]$ range. Although this reduced range has been used in several other studies that aim to find optimized weight settings for congestion [39], we here also present a brief comparison of the effect of such reduced search space on the quality of the obtained solutions.

The experiments consider weight ranges $[1, w_{max}]$, with $w_{max} \in (20, 50, 100)$. A SOEA, with the previously described running configurations, was used to optimize the congestion for each of the network topologies, considering $D0.3$ traffic matrices. The optimization results obtained by running 30 times each scenario are presented in Figure 3.12.

It is possible to observe that the quality of the solutions depends on the value of the maximum weight parameter. By reducing the search space, the EA converges faster, requiring fewer iterations. Simultaneously, shorter ranges increase the probability of finding equal cost multi-paths. It is expected that by increasing the number of iterations, solutions with equally good quality will be found independently of the weight range. It is important to observe that the best solution for all ranges was always obtained with the narrower weight configuration and that by increasing the size of the weight range no better single solution was obtained.

In the context of multi-objective problems, the weight range assigned to each link also has an impact on the convergence and diversity of the solutions. As the weight range increases, the number of iterations, required by the algorithm, to converge to the Pareto-optimal also increases. The range selection choice reflects the importance given to the convergence, versus the diversity of the solutions. As an illustrative example, Figure 3.13 presents the best Pareto fronts of 30 runs for MOPs that aim to optimize two distinct D0.3 traffic matrices for the $Rand30_4$ network topology.

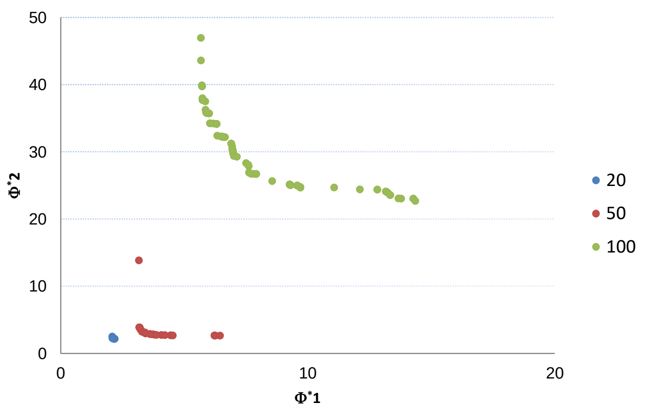


Figure 3.13: *Pareto fronts with distinct weight range configurations.*

3.6.2 EAs Running Time

The amount of time required to obtain optimized solutions is also a very important factor when choosing an MOEA. As the size of the solutions depends on the number of links, the time required by each MOEA depends mostly on the number of edges on the topology. Some illustrative running times are presented in Table 3.13 which were obtained with a Core i7 processor.

None of the algorithms uses multi-threading, and, consequently, the optimization process duration can directly be compared. The amount of time, per iteration, presented for the SOEA is relative to a single aggregated sum configuration, and not the three trade-offs considered in the experiments.

As an example, the amount of time required for each algorithm, when applied to congestion optimization for the $Rand30_2$ topology with a stopping criterion of 1000 iterations, is 31 minutes for SOEA (3 runs are needed to obtain the 0.25, 0.5 and 0.75 trade-offs), 12.3 minutes for NSGA-II, and 24.3 minutes for SPEA2.

It is, nonetheless, possible to improve the time required by each algorithm. One obvious way to achieve this goal is to resort to multi-threading in the evaluation process,

Table 3.13: *Average run time per iteration for two demands optimization (in seconds).*

Topology	Nodes	Edges	SOEA	NSGA-II	SPEA2
Abilene	12	15	0.061	0.031	0.070
$Rand30_2$	30	55	0.635	0.735	1.459
$Rand30_4$	30	110	0.964	1.234	1.889

where each parallel thread would evaluate a portion of the solutions.

3.7 Conclusions

The simplicity of link-state protocols and their reliability is proven over the last decades, continues to justify the use of such routing algorithms in the context of IP backbone networks. However, the dynamic conditions of IP networks, such as changes on traffic demands and disruptions on the underlaying topology need to be addressed so that the network continues to ensure a good operational performance even if such events take place. An administrator could react to such changes by reconfiguring the link weights assigned to each link but with a temporary negative impact on traffic flows. Preventive optimization approaches can effectively take into consideration such changes to compute weights configurations that allow the network to continue to ensure proper levels of performance even in variable environmental conditions.

In this chapter, two link-state weights setting configuration problems were approached using single and multi-objective EAs: NSGA-II, SPEA2 and single objective EA using weighted-sum aggregation (SOEA). The results showed that for simpler problems the single objective optimization approaches provide solutions with fitness values as good as the MOEA algorithms. As the difficulty of the addressed problem increases, for more complex network topologies and more demanding traffic requirements, NSGA-II provides better solutions. Moreover, NSGA-II can offer a broader set of solutions, with clear trade-offs between objectives. Additionally, the time required by NSGA-II to deliver such a set of solutions when distinct trade-offs must be considered is far less than an SOEA using weight-aggregation. Finally, we also showed that the use of a reduced search space does not negatively impact the quality of the solutions and that it enables a faster convergence of the EAs.

Chapter 4

Optimizing Traffic Load Balancing using SDN

Software-Defined Networking (SDN) is an architecture designed to make the network more flexible and agile with a centralized view and control of the network's operations. This chapter addresses the optimization of resources utilization in hybrid IP/SDN networks, that is, in networks where both traditional networking and SDN protocols operate in the same environment. We start by arguing the advantages of having both architectures, SDN and traditional routing protocols, in the same network. Traffic load balancing has an important role in the distribution of traffic and consequently in the utilization of network resources. Section 4.2 presents two traffic load balancing functions, which are at the core of the devised work, and proposes a new method to implement corrections in traffic load balancing. Two routing models for hybrid IP/SDN networks are then defined which enable near optimal network resources utilization. The chapter proceeds evaluating and comparing the proposed routing models when all nodes implement SDN functionalities, but also for incremental SDN deployments where hybrid IP/SDN nodes coexist with legacy routing devices. Traffic variations, which may undermine the network's operation, are also addressed by adding to the routing models the ability of adjusting traffic load balancing to the new network operational context. The same strategy is later used to improve the distribution of traffic after single link failures, which may also have a severe impact on the network performance.

4.1 Introduction

Software Defined Networking (SDN) has emerged as a new networking paradigm that decouples the network control plane from the network data plane (section 2.2.5). In SDN based networks, the controller makes decisions and distributes instructions to devices on how to forward traffic flows. This separation of the network control and data planes eases the network control and management tasks, thus enabling new approaches to deal with network congestion and traffic distribution optimization problems.

SDN networks have many advantages over traditional networks, for example, they provide fine-grained control for data traffic flows and support the enforcement of security policies. However, SDN is typically not fully deployed in networks due to several reasons, such as a limited budget for network infrastructures and the fear of downtime during the transition to SDN. Additionally, an Internet Service Provider usually handles millions of forwarding entries, and SDN-enabled switches can typically support only tens of thousands. One possible solution to address these concerns is to deploy a limited number of SDN-enabled devices alongside traditional, legacy, network devices. A network containing a mix of SDN and legacy network devices or functionalities, such as the Google's private SDN backbone network B4 [56], is commonly referred to as a hybrid IP/SDN network.

Hybrid IP/SDN refers to a networking architecture where both centralized and decentralized paradigms coexist and communicate together to different degrees to configure, control and manage network behavior, optimizing its performance [95]. Hybrid IP/SDN networks can commonly be deployed in three different ways: 1) setup up SDN switches in a legacy network, that is, the network is built with two types of devices, legacy routers and SDN switches, 2) hybrid IP/SDN switches having both SDN switching and legacy switching functionality, and 3) deploying hybrid IP/SDN switches on legacy networks as a combination of the previous. The last ones have particular advantages that result from combining the robustness of Interior Gateway Protocols (IGP) with the flexibility of SDN. We adopt a relaxed designation of "Hybrid IP/SDN switches" to identify devices running both traditional IP routing protocols and SDN related processes. Such devices perform both layer 2 and layer 3 traffic forwarding, and run a routing process such as OSPF. Hybrid IP/SDN switches add flexibility to traditional IGP by enabling the configuration of arbitrary forwarding paths that overcome distributed routing protocols restrictions. On the other hand, IGPs give SDN networks scalability and robustness. Enabling destination based and SDN match/action forwarding strategies allows to dramatically reduce the number of switch entries that need to be managed by a controller. Not less important is that in case of a SDN failure, such as a controller failure, the network can continue to perform relying only on IGP routing.

The global visibility offered by SDN controllers has been exploited in various recent proposals to distribute traffic near-optimally across precomputed paths. SWAN [53] distributes traffic flows across k-shortest paths, using reserved amounts of links' capacity for configuration updates. B4 [56] distributes flows across multiple paths to improve utilization while ensuring fairness. B4 intents to make use of all links' capacity, however, by doing so packet loss is inevitable in the occurrence of link failures. The system proposed in [108] incorporates failures into a linear programming formulation and computes a diverse set of paths offline. It then uses a simple local strategy to dynamically adapt sending rates at each source. Also addressing concerns about network failure recovery and robustness, the authors in [112] proposed a hybrid SDN architecture that employs a controller for long-term optimization and relies on distributed protocols for short-term reaction to failures.

A discussion on the cooperation of distributed (e.g., OSPF) and centralized (SDN based) routing control planes in hybrid SDN switches is presented in [116]. The authors presented a theoretical framework to address novel forwarding anomalies that may arise from the cooperation of distributed and centralized control planes and derived sufficient conditions to assure anomaly-free routing. Similarly, in [121] a hybrid switching is employed for scalability.

This chapter addresses the optimization of resources utilization in hybrid IP/SDN networks. We explore distinct deployment scenarios, namely scenarios where all switches are hybrid, that is, implement both traditional IGP routing (OSPF or IS-IS) and SDN functionalities, and scenarios where hybrid IP/SDN networks are incrementally deployed. We propose a routing model that explores the optimization of a traffic splitting function adopted from a recently proposed link-state protocol. We later extend the routing model to enable traffic load balancing corrections that allow improving networks performance when disruptive events need to be addressed, such variations in traffic demands and single link failures.

4.2 Exponential Flow Splitting

4.2.1 Traffic Load Balancing

The flexibility of SDN opened new possibilities for Internet TE and, in particular, new solutions to efficiently distribute network traffic, preventing the congestion of specific networks links. Traditionally, to overcome congestion on carrier-grade networks, reactive measures, such as adapting traffic flows at the edge of the network [18, 89], needed to be fulfilled. There are also preemptive approaches, such as those explored in the previous chapter, where estimated or known traffic requirements are distributed optimally on the available resources. However, forwarding path restrictions imposed by IGP routing and the limited load balancing capabilities of Equal-Cost Multi-Path (ECMP) undermine both preemptive and reactive congestion avoidance measures. With ECMP, when more than one shortest path exists to the same destination, traffic is split evenly between next hops on the shortest paths. Although the use of ECMP enables a better traffic distribution on traditional routing, it can not offer an optimal use of resources [39, 119].

Optimal traffic distribution can only be achieved by unequal load balancing, where different amounts of traffic are forwarded to next hops. MPLS, briefly introduced in Section 2.2.4, was designed to counter these limitations but brought a high complexity to network management and configuration tasks. To enable unequal load balancing, Cisco proposed the Interior Gateway Routing Protocol (IGRP) and the Enhanced Interior Gateway Routing Protocol (EIGRP). Apart from being proprietaries, IGRP is a Distance Vector protocol, hence with a slow convergence. IGRP is now an obsolete routing protocol and was completely replaced by EIGRP. However, EIGRP has also some scaling issues. As the scale of the network increases the risk of instability or long convergence times becomes greater [113]. EIGRP also reveals some problems in the redistribution of external routes when used with multiple Autonomous Systems. Additionally, the fractions of traffic assigned to each outgoing link are proportional to a variance multiplier, which prevents realizing optimal resource utilization.

Valiant Load Balancing (VLB) [115] was proposed as a way to route packets between sparsely connected parallel computers, without two packets passing down the same link at any time. The general idea is to forward packets between a source and destination through random midpoints of fully-connected logical meshes thus obtaining an optimal usage of resources. VLB has recently been applied in a number of other settings including datacenter networks, wide-area networks, and software switches [9, 125]. However, sending traffic through intermediate nodes increases path length, which can increase latency dramatically. Furthermore, VLB can exhibit degraded performance compared to the optimum when traffic is dropped due to congestion [26].

The SDN architecture enables new approaches to the link load balancing problem. Hedera, proposed in [2], addresses the congestion problem on data centers caused by multiple flows being mapped to the same path. Hedera identifies elephant flows, extremely large and continuous flows, and resorting to greedy first-fit or simulated annealing algorithms, the load balancing decisions making chooses an alternative path to a such flow that improves network resources utilization. However, this scheme is only suitable for specific data centers and lacks flexibility and scalability. CONGA [5] also proposes to optimize congestion on data centers by estimating per link loads and allowing source nodes to balance the network load according to the congestion state. Although CONGA provides a quick response to congestion it does not scale well. HULA [60] overcomes the CONGA scalability issues by resorting to a Distance Vector algorithm and allowing next-hops to make load balancing decisions. HULA has a good scalability and can rapidly adapt to new conditions but is difficult to implement.

DIFANE [123] addresses the problem of maintaining per flow rules in all switches and the heavy dependence on centralized controllers to make forwarding decisions. DIFANE keeps all traffic in the data plane by selectively directing packets through intermediate switches that store the necessary rules. The controller is relegated to the simpler task of partitioning these rules over the switches, which are only modified when the network changes. Despite its virtues, DIFANE requires a complex switches architecture not suitable for large-scale networks. The traffic load balancing problem was also addressed for hybrid IP/SDN networks, such as the one presented in [109], where the SDN controller starts by mapping multiple flows with the same destination to disjoint paths. Flows on the same path are then aggregated, and a Lazy Route Update (LRU) improves the load balancing by spreading network traffic across the entire network. An extensive survey of load balancing research in SDNs can be found in [25].

Load balancing technologies are one of the most important ways to effectively allocate resources and improve networks performance. Although many proposals have emerged, they suffer from scalability, complexity or implementation problems. The Distributed Exponentially-weighted Flow Splitting (DEFT) [119] is a link state routing protocol whose main advantage is to allow forwarding traffic along links on non-shortest paths. While OSPF and IS-IS implement ECMP to evenly split traffic along multiple paths with equal cost, DEFT aims to diminish the overall congestion levels in the network by exploring the usage of non-equal cost paths. The load balancing strategy proposed by DEFT assigns flows to a next-hop with a probability that decreases exponentially with the additional length of the path, when compared with the shortest path length. Other uneven traffic splitting proposals for OSPF exist, such as the ones proposed in [103, 105]. However, they present a relevant limitation. First, if only link weights are available, routers are unable to compute the traffic splitting fractions independently. Second, they must rely on solutions that involve, for example, a distributed

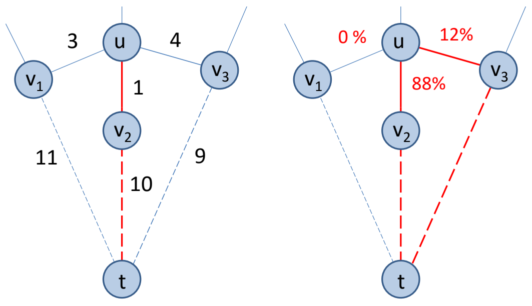


Figure 4.1: *DEFT traffic splitting example.*

and synchronized load balancing management, which increases complexity and communication overhead in the network [74]. DEFT on the other hand, only requires the link weights configuration to compute traffic splitting rations.

The Penalizing Exponential Flow-spliTting (PEFT) [120] was proposed as an evolution of DEFT and adds ”information” concerning the entire forwarding path to traffic splitting. The resulting difference between PEFT and DEFT is that, in terms of flow splitting, DEFT is a link-based protocol, whereas PEFT is a path-based protocol. In PEFT, outgoing flows at any node are split among all feasible paths to a destination node, whereas in DEFT, the outgoing flows are split among all possible shortest path links. In the next sections, we describe both load balancing mechanisms of DEFT and PEFT characterized by making use of non-shortest paths to forward portions of traffic. By relying only on link weights to make forwarding decisions, they provide a scalable and straightforward load balancing solution.

4.2.2 DEFT Traffic Load Balancing

DEFT load balancing mechanism assigns flows to a next-hop with a probability that decreases exponentially with the additional length of the path. The distance from a node u to a node t, when traffic is routed thought a next-hop v, can be expressed as $w_{u,v} + d_v^t$, where d_v^t is the shortest distance from the next-hop v to t, and $w_{u,v}$ is the weight of the link (u, v). The extra length of the path from u to t through v, when compared to the shortest path with the same origin and destination, d_u^t, can thus be expressed by Equation 4.1, and denoted as $h_{u,v}^t$. The exponential function $\Gamma\left(h_{u,v}^t\right)$, Equation 4.2, indicates the relative amount of traffic destined to t that node u will forward via outgoing link (u, v), and decreases with the extra length $h_{u,v}^t$ of a path. The proportion of traffic with destination t to be forwarded at u using the outgoing link (u, v) is then computed by Eq. 4.3, where E is the set links.

$$h^t_{u,v} = d^t_v + w_{u,v} - d^t_u \tag{4.1}$$

$$\Gamma\left(h^t_{u,v}\right) = \begin{cases} e^{-\frac{h^t_{u,v}}{p}}, & \text{if } d^t_v < d^t_u \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

$$P\left(h^t_{u,v}\right) = \frac{\Gamma\left(h^t_{u,v}\right)}{\sum_{(u,i)\in E}\Gamma\left(h^t_{u,i}\right)}, \tag{4.3}$$

In Figure 4.1, an illustrative example is provided where traffic needs to be routed from a node u to a node t. The only next-hop on a shortest path from u to t is node v_2. Therefore, and using an ECMP load balancing scheme, all the traffic would be forwarded by v_2. However, DEFT enables routers to forward traffic on non-shortest paths. All adjacent nodes whose shortest distance to t is less than $d(u,t)(=11)$ will be considered as next-hop. The node v_3 is in such conditions, $d(v_3,t) = 9$, and thus, with the DEFT strategy, an extra path with destination t can be used, and a fraction of the traffic destined to t can be forwarded through node v_3. The computed proportional factors $\Gamma\left(h^t_{u,v_2}\right)$ and $\Gamma\left(h^t_{u,v_3}\right)$ (Eq. 4.2 with $p = 1$) translate into fractional splits (Eq. 4.3) of, respectively, 88% and 12%. The condition $d^t_v < d^t_u$ in Equation 4.2 ensures that traffic is always forwarded towards the destination, thus preventing loops.

4.2.3 PEFT Traffic Load Balancing

Link-state routing protocols can be categorized as link-based and path-based in terms of flow splitting. While DEFT is a link-based flow splitting function, PEFT is a path-based flow splitting function. The computation of splitting ratios at each node u include locally significant "path information" variables, γ^t_v , that correct the splitting ratios for the next hops v and enable to realize path-based optimal traffic forwarding, Equation 4.4. Each positive real number γ^t_v is interpretable as the "equivalent number" of shortest paths from the node v to the destination t. In the particular case of γ^t_t such value is defined as 1.

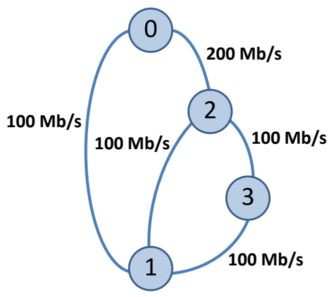
$$\Gamma\left(h^t_{u,v}\right) = \begin{cases} e^{-\frac{h^t_{u,v}}{p}}.\gamma^t_v, & \text{if } d^t_v < d^t_u \\ 0, & \text{otherwise} \end{cases} \tag{4.4}$$

The authors in [120] formally proved that PEFT is able to achieve optimal routing configurations by solving a network entropy maximization problem. Although an exact traffic splitting function for PEFT is provided, to reduce the computational complexity we adopt an alternative computational method known as Downward PEFT. This PEFT implementation is faster than the exact one and provides an approximate solution to realize an optimal traffic distribution. Though Downward PEFT does not provably achieve optimality, it comes extremely close, and the γ_v^t values needed to compute the fractional splits can be obtained by solving the linear system presented in Equation 4.5. In practice, when the lower bound of all link weights is large enough, Downward PEFT solutions are of the same quality as those provided by an exact PEFT [120].

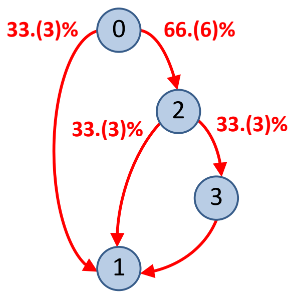
$$\begin{cases} \sum\limits_{v \in N, v \neq u_1} e^{-h^t_{u_1,v}/p} . \gamma_v^t - \gamma_{u_1}^t = 0 \\ \vdots \\ \sum\limits_{v \in N, v \neq u_{n-1}} e^{-h^t_{u_{n-1},v}/p} . \gamma_v^t - \gamma_{u_{n-1}}^t = 0 \\ \gamma_t^t = 1 \end{cases} \tag{4.5}$$

To better understand the differences between PEFT and DEFT let us consider a practical example. Figure 4.2 presents a network whose only traffic requirement is to route 300 Mb/s of traffic from node 0 to node 1. Considering the network links' capacity, Figure 4.2a, the only solution capable of responding to the traffic necessities consists in forwarding 1/3 of the intended traffic on link (0,1) and 2/3 across link (0,2), Figure 4.2b. Such a solution is not feasible using ECMP load balancing, Figure 4.2c, and thus unequal load balancing needs to be put in place.

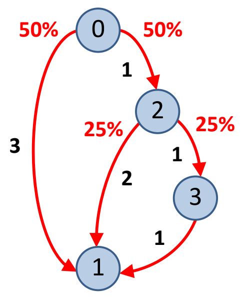
A solution that uses DEFT load balancing considers two shortest paths from node 0 to node 1 (path 0-2-1 and 0-2-3-1) and a non shortest path through link (0,1). To accomplish the intended traffic splitting at node 0, $\Gamma\left(h^1_{0,1}\right)$, Equation 4.2, needs to be 1/2 $((1/2)/(1/2+1) = 1/3)$, and therefore $h^1_{0,1}/p$ should be equal to $\ln 2 \approx 0.693$. If the parameter p is set to 1, the extra length $h^1_{0,1}$ needs to be the nearest integer value to 0.693, that is $h^1_{0,1} = 1$. A possible link weights configuration that reflects the enunciated conditions, as well as the respective traffic splitting ratios, is presented in Figure 4.2d. Such a solution, although better than the one provided by ECMP, is not optimal as $h^1_{0,1}/p = 1$ is a poor approximation to 0.693. A better approximation can be obtained by considering greater values of p in Equation 4.2 and adjusting the value of the extra length $h^1_{0,1}$. As examples, a configuration with $p = 10$ and a extra length $h^1_{0,1} = 7$ provide an $h^1_{0,1}/p$ of 0.7, while with $p = 100$ and $h^1_{0,1} = 69$, $h^1_{0,1}/p$ is equal to 0.69. While the first combination of values already enables to achieve the intended load balancing of traffic at node 0, see Figure 4.2e, greater values of the parameter p increase the solution accuracy. However, a greater value of the parameter p implies a greater



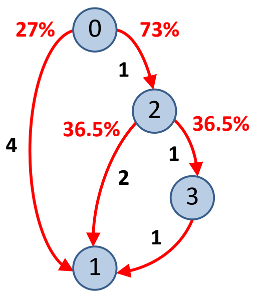
(a) *Topology*



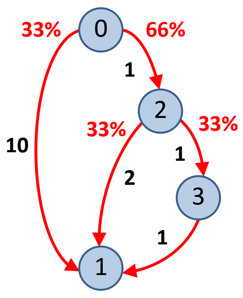
(b) *Optimal Load Balancing*



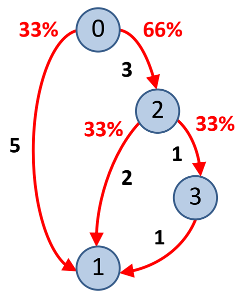
(c) *ECMP*



(d) *DEFT*
$p = 1$ *and* $h^1_{0,1} = 1$



(e) *DEFT*
$p = 10$ *and* $h^1_{0,1} = 7$



(f) *PEFT*
$p = 1$

Figure 4.2: *PEFT versus DEFT: Illustrative example.*

extra length value, $h^1_{0,1}$, and as consequence requires a wider range of link weights with implications in the size of the optimization search space.

With regard to PEFT, it is a path-based flow splitting function which considers all possible acyclic paths between a source and a destination to distribute and load balance traffic. In the illustrative example, there are three equal-length paths from node 0 to node 1, and node 0 evenly splits traffic across them as shown in Figure 4.2f. Considering the link weights configuration in Figure 4.2f and a parameter $p = 1$, the matrix representation of the linear equations system that computes the γ^1_i variables (Equation 4.5) can be expressed as shown in Equation 4.6. All links are shortest path links and therefore all $\Gamma\left(h^t_{u,v}\right)$ are 0 or 1. Consequently, the system's solutions, γ^1_i, are nothing more than the number of available equal cost paths from node i to node 1,

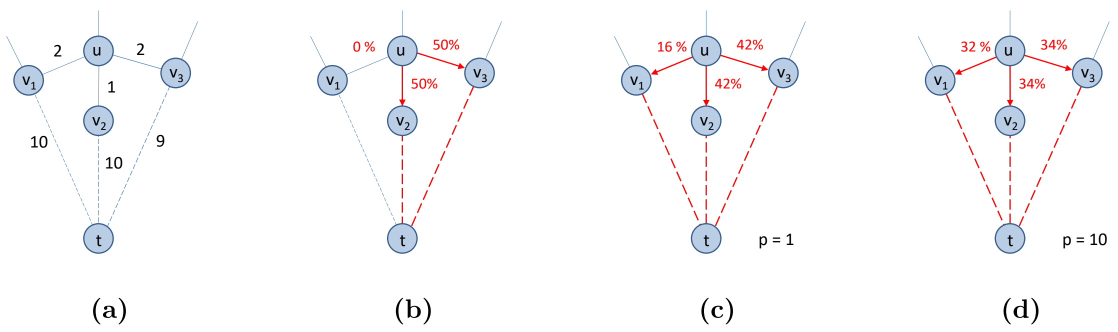


Figure 4.3: *Node-p value effect on traffic load balancing.*

Equation 4.7.

$$\begin{bmatrix} -1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & -1 & 1 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} \gamma_0^1 \\ \gamma_1^1 \\ \gamma_2^1 \\ \gamma_3^1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{4.6}$$

$$\begin{bmatrix} \gamma_0^1 \\ \gamma_1^1 \\ \gamma_2^1 \\ \gamma_3^1 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ 2 \\ 1 \end{bmatrix} \tag{4.7}$$

4.2.4 Node-p Values

The extra length $h_{u,v}^t$ is an integer that takes values in range $[0, w_{\max} - 1]^1$ and defines the fraction of traffic with destination t to be forwarded along (u,v). The parameter p in Equations 4.2 and 4.4 was initially thought as a weighted constant to scale the penalizing function Γ in a range $[w_{\min}, w_{\max}]$ of possible link weights. However, by assigning to each network node distinct values of p, which we denote as node-p[2], they become an additional parameter that can be tuned to alter traffic load balancing and improve traffic distribution. To better understand the effect introduced by distinct node-p values in DEFT, we next provide an illustrative example, Figure 4.3, where the node-p variable at a node u is set to distinct values.

Considering the installed link weights, shown in Figure 4.3a, the next-hops on a shortest path from u to t are nodes v_2 and v_3. Therefore, using the ECMP load balancing

[1] when $d_v^t < d_u^t$

[2] to contrast with statistical p-values

scheme, traffic will be equally split and forwarded to both next-hops (Figure 4.3b). However, DEFT enables routers to forward traffic on non-shortest paths. All adjacent nodes on non-shortest paths that are at a distance less than $d(u,t)(=11)$ of t will be considered next-hops. The node v_1 is in this condition, $d(v_1,t)=10$. For $p=1$, 16% of traffic arriving at u with destination t will be forwarded to v_1 (Figure 4.3c). For $p=10$, this percentage raises to 32%, and v_1 receives almost one third of the traffic to t (Figure 4.3d). The remaining traffic is equally split between the links on the shortest paths.

The manipulation of node-p variables permits to change the amount of traffic forwarded on non-shortest paths without any change to the installed link weights and consequently on forwarding paths. Instead of assigning a unique node-p value to all nodes on the topology, a different value is assigned to each node u, corresponding to a total of $|N|$ values that can be used to improve hop-by-hop traffic load balancing, where N is the set of topology nodes. This contribution enables, for example, to address changes on traffic requirements by optimizing the node-p values and without any alteration to the installed link weights. Although such a solution does not provide an optimal load balancing, it has the advantage of requiring the optimization of a reduced set of values, $|N|$, which contrasts with an optimization that contemplates all outgoing links at each node for all destinations, that is, in the order of $|N|^3$ variables.

4.3 Routing Model

The DEFT and PEFT load balancing functions, although being both capable of delivering unequal hop-by-hop load balancing are in practice very different. While PEFT may assign different amounts of traffic to each next-hop on the shortest path, in this same case, DEFT only can perform ECMP. As a consequence, in a SND enabled PEFT model all traffic needs to be managed with SDN switching. Contrariwise, DEFT performs an even load balancing between shortest paths and only uneven fractions of traffic are forwarded using non-shortest path links. The two distinct load balancing solutions spawn two distinct routing models.

4.3.1 Routing Model with DEFT

Hybrid IP/SDN deployments offer advantages to networks' operations and management. While SDN offers more expedite and flexible configuration solutions and finer grain traffic forwarding alternatives, traditional IGP routing offers scalability and reliability to networks' operations. In the presence of a link failure, for example, IGPs such as OSPF and IS-IS provide an automatic rerouting of traffic which contrasts with a more complex management required by pure OpenFlow SDN deployments. To address a link failure in SDN networks, a SDN controller needs to compute recovery paths and install new forwarding rules for the affected traffic. Apart from such tasks being potentially overly complex in large networks, they are also time consuming which may undermine networks operational conditions. Still, SDN offers a great flexibility and can implement a cross-layer packets classification by considering packet headers at different protocol levels. Therefore it is possible to specify a set of conditions regarding flows that have to be handled by classical routing protocols and the ones to be handled using SDN.

In view of the above mentioned, the traffic load balancing carried out by DEFT suggests a natural separation between shortest path and non-shortest path traffic forwarding. While all traffic that travels over edge-to-edge shortest paths is to be managed by the IGP, traffic that makes use of non-shortest paths is to be managed via SDN. When all nodes are hybrid IP/SDN switches, ingress edge routers classify incoming traffic as suitable to be routed by IP routing protocols or as traffic to be routed via SDN. A Local Management Entity (LME) identifies which flows are candidate to be routed via SDN using a match criterion (e.g., source/destination, type of service, protocol, in/out TCP port). If a flow is identified to be be SDN routable, its first packet is sent to a controller who will select the forwarding path and install the necessary rules for that flow on the Ternary Content-Addressable Memory (TCAM) of all SDN enabled switches on the selected path. The SDN controller assures the splitting of traffic on non-shortest paths by querying statistics, such as the flow count, gathered by all hybrid switches. A SDN controller may also decide that a flow should be forwarded by the IGP and, in such a case, no rules are installed for that particular flow and the information contained in the classical routing tables will be used to forward such flow.

The IGP, OSPF or IS-IS, routes traffic flows that are not managed by SDN and ECMP assures the load balancing of traffic between equal cost paths. Only flows and traffic splittings for non-shortest path links are managed and tracked by a SDN controller. In case of a SDN failure, such as a controller failure or if a switch ceases to be able to communicate with a controller, the IGP assures connectivity and the routing of all traffic for which there is no TCAM entry. This model provides scalability, reliability, and robustness to the network routing and forwarding functions with low overhead. Although such routing scheme could be implemented with an extension of OSPF or IS-IS routing protocols, the correction of traffic load balancing between non-shortest path links and shortest path links needs to be centrally managed and will be further detailed in the next sections.

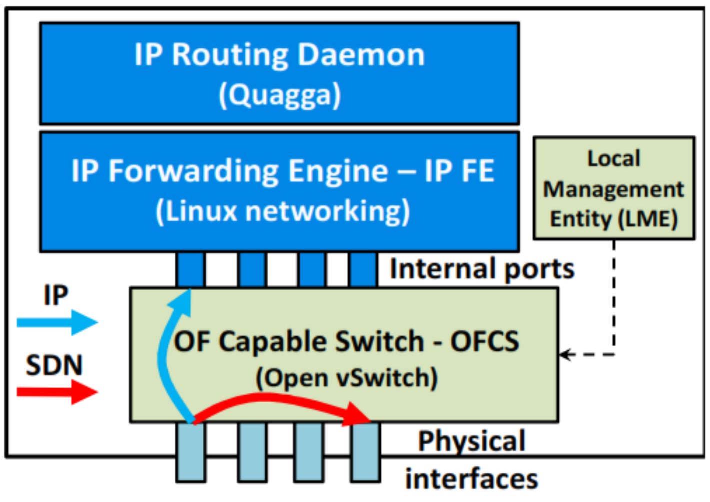


Figure 4.4: *The Open Source Hybrid IP/SDN (OSHI) node (Adapted from[94])*

In [94] the authors proposed a hybrid IP/SDN switch architecture similar to the one assumed in the previous scenario, the Open Source Hybrid IP/SDN (OSHI), Figure 4.4. OSHI consists of an OpenFlow Capable Switch (OFCS), an IP forwarding engine, and an IP routing daemon. The OFCS is connected to the set of physical network interfaces of the integrated IP/SDN network, while the IP forwarding engine is connected to the virtual internal OFCS ports.

Incremental deployments of hybrid IP/SDN switches follow a slightly different model. Flows are classified to be routed by the IGP at each individual hybrid IP/SDN switch. This task can be performed solely by the LME without the need to send any packet to the controller. The role of the controller is to evaluate the network performance and implement load balancing corrections on non-shortest paths whenever necessary.

4.3.2 Routing Model with PEFT

The PEFT routing model, contrary to the DEFT model, requires all traffic to be managed by SDN , in all nodes where an uneven splitting of traffic is performed. Due to the uneven splitting of traffic between shortest paths, traditional routing protocols, which usually implement ECMP to load balance traffic between shortest paths, can not be used to achieve the proposed goal of optimal network resources utilization. As a consequence, the hybrid IP/SDN routing model for PEFT contemplates networks which are built with two types of devices, legacy and SDN switches, or networks where all devices are SDN switches.

It is important to emphasize that both routing models offer loop-free routing paths. Regardless of the current and next-hop node type, a legacy router, a SDN switch or a hybrid IP/SDN node, the next-hop will always be closer to the destination. This assertion is a consequence of the condition $d_v^t < d_u^t$ in Equations 4.2 and 4.4.

4.4 Single Objective Congestion Optimization

DEFT and PEFT enable to use non-shortest path links to route portions of traffic and thereby improve the utilization of network resources. In this section, we evaluate the proposed models and resort to Evolutionary Algorithms (EAs) to compare optimized solutions with traditional routing schemes. The authors in [119, 120] resorted to integer linear programming to minimize congestion on networks running DEFT or PEFT routing protocols. In [46], a Biased Random-Key Genetic Algorithms (BRKGA) is used with the same goal. Although we here also use an EA based approach, there are some relevant differences. The authors in [46] set a global value $p = 1.8$ and combined this value with a maximum $h_{u,v}^t$ of 9 to implicitly define a fractional threshold beyond which a link would not be used to forward traffic, that is, the fraction of traffic is approximately null. We choose to take another direction and define this threshold explicitly. A routine added to the traffic splitting computation imposes the configured minimum traffic fractioning. This method allows to provide freedom to the node-p values and set a controlled minimum split fraction. Another difference is that the single objective BRKGA encodes solutions as a vector of real values in the range [0,1[which are decoded into "link weights". We, on the other hand, use an integer representation and solutions are directly mapped into link weights. The EAs survival and parents selection processes are also significantly different. In the BRKGA approach, each individual is generated combining one element selected at random from the elite partition in the current population and one from the non-elite partition, thus the term "bias". This selection mechanism contrasts with the roulette wheel selection described in Section 3.2.3.

Similarly to the previous chapter, network topologies are modeled as directed graphs $G(N, E)$, where N represents a set of nodes, and E a set of links, with capacity constraints c_a for each $a \in E$. Each node may be configured as legacy only, as hybrid IP/SDN switches (DEFT) or as SDN switch (PEFT). We devise two set of experiments, for fully deployed SDN networks, where all nodes possess SDN functionalities, and for SDN incremental deployments where a subset of nodes are legacy routers which only run the IGP. The optimization problem can be formulated in both cases as follows:

Table 4.1: *Congestion comparison of OSPF/ECMP, DEFT Hybrid IP/SDN and PEFT SDN with traditional weighting schemes.*

Topology			Demands	Traditional Schemes		Optimized Weights		
Name	Nodes	Links		Unit	InvCap	ECMP	DEFT	PEFT
$Rand6_2$	6	18	D0.40	229.70	731.40	2.00	1.89	1.89
$Rand30_2$	30	110	D0.30	15.60	130.69	1.42	1.38	1.37
			D0.50	317.73	644.47	3.43	2.95	2.97
$Rand30_4$	30	220	D0.30	260.54	498.36	2.70	1.71	1.67
			D0.40	426.76	717.95	20.68	6.14	6.14
$Rand50_2$	50	194	D0.30	437.70	339.96	1.68	1.61	1.61
$Rand50_4$	50	380	D0.30	156.04	261.38	3.67	2.38	2.38

given a network topology and a traffic demand matrix, the aim is to find a set of weights w that minimizes the function Φ^*, defined in Section 2.3.2, which evaluates the network congestion cost.

4.4.1 Fully Deployed SDN Networks

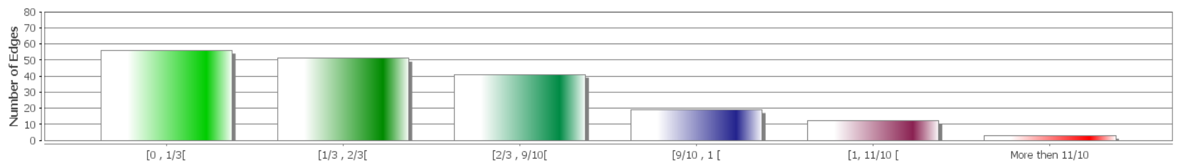
A set of single objective optimization problems were used to compare congestion metrics on networks where 1) all nodes are OSPF/ECMP routers; 2) all nodes are hybrid IP/SDN switches that use the DEFT traffic load balancing function; and 3) all nodes are SDN switches and the controller implements the PEFT load balancing function. For comparison purposes, we also include the congestion levels obtained by two commonly used traditional OSPF link weights configuration schemes, namely: *InvCap*, where each link weight is assigned to a value inversely proportional to its capacity; and *Unit*, where every link weight is set to one. The network topologies and traffic matrices were randomly generated as described in Section 3.4.2. In the experiments, all hybrid IP/SDN nodes (DEFT) and pure SDN nodes (PEFT) have a default node-p value of 1. For both models, we defined a non-shortest path link usage threshold of 5%, that is, when an outgoing link at a node u is used to forward traffic with destination t, it forwards at least 5% of the aggregated received traffic with destination t. The experiments were run in randomly generated networks varying in size and minimum in/out node degree. The synthetic topologies were generated with the Brite topology generator [68], using a Barabasi-Albert model, with a heavy-tail distribution and an incremental grow type. The link capacities uniformly vary in the interval $[1; 10]$ Gbits. All the optimization procedures use the same SOEA configuration described in Section 3.4.2 and with a stopping criterion of 1000 iterations. Table 4.1 presents the averaged results of 10 runs of each scenario.

The illustrative results show that any optimization, independently of the routing mechanism and load balancing scheme, outperforms the congestion levels attained by the traditional weighting schemes *Unit* and *InvCap*. Although this result is not surprising in itself, it displays the enormous gap between the solutions and clearly states the advantages of resorting to TE optimization. As mentioned, congestion values above 10 2/3, observed in all *InvCap* and *Unit* experiments, indicate that the network topology is unable to support the considered demands resulting in severe congestion and packet loss.

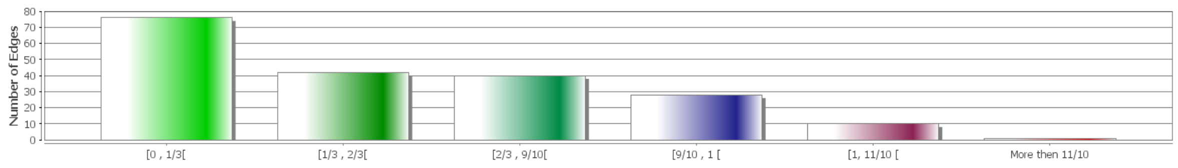
A direct comparison of the three optimization schemes allows recognizing that, in all case scenarios, the deployment of SDN reduces the network congestion to lower values than those attained by OSPF/ECMP routing optimizations. Furthermore, DEFT and PEFT load balancing, with their respective implementation model, always provide solutions which are of the same quality. There is, however, a clear advantage when using the DEFT model when compared to the PEFT model. The DEFT model is an hybrid IP/SDN model and, as already argued, it inherits the IGP advantages (scalability, simplified management, reliability in case of failures, etc...). The DEFT load balancing model forwards portions of traffic along non-shortest path links which are managed via SDN forwarding rules. The remaining traffic is managed by the IGP and equally split between outgoing shortest path links. The PEFT load balancing model, however, requires all traffic to be managed with SDN and that a forwarding rule must be installed for each flow in all traversed switches. With PEFT, outgoing shortest path links at a node may not forward an equal amount of traffic. The introduction of the γ variable in the splitting ratios computation, Equation 4.4, may cause traffic not to be evenly divided between links on shorter paths. Although the PEFT model facilitates network management and enables programmatically efficient network configuration, it requires the management of more forwarding rules and therefore, it is more prone to configurations errors. Despite their differences, they produced similar results in all case scenarios.

It is important to note that, due to the heavy penalization applied to each link by the congestion measure Φ^*, differences between congestion values do not linearly translate into link utilization. In the case scenario with a 30 nodes topology and 220 links, for a D0.4 demand matrix, the mean optimized congestion values are 20.678 and 6.138 for ECMP and DEFT respectively. In Figure 4.5 an illustrative network load chart for both optimization load balancing schemes shows that, in reality, the difference between the overutilization of link's capacity is less than it could be expected.

There are also cases, such as the optimization for the topology with 30 nodes and



(a) OSPF/ECMP



(b) Hybrid IP/SDN (DEFT)

Figure 4.5: *OSPF/ECMP vs hybrid IP/SDN networks link loads optimization. (Network topology with 30 nodes and 220 edges for traffic demand matrices D.04)*

110 links, for a traffic matrix D0.30, where the congestions of OSPF/ECMP and hybrid IP/SDN networks (DEFT) or pure SDN networks (PEFT) are very akin. This observation is a consequence of the low average in/out degree of each topology node. The hybrid IP/SDN implementation can only outperform OSPF/ECMP if there are a significant number of nodes with enough available links for traffic engineering.

4.4.2 Incremental Deployment of Hybrid IP/SDN Networks

We, so far, only considered hybrid IP/SDN deployments where all devices implement both SDN switching and legacy routing functionalities. A deployment of SDN replacing all the existing legacy devices with hybrid IP/SDN devices can be financially very costly. Providentially, hybrid IP/SDN networks can be set up without extending SDN functionalities on all network nodes and still allowing to reap some of the SDN paradigm benefits [110, 117]. With this in mind, we devised a set of experiments where we evaluate the proposed congestion optimization method with an incremental number of hybrid switches.

This new optimization problem is formulated as follows: given a defined number k of hybrid IP/SDN switches (DEFT) [or SDN switches (PEFT)] to be deployed, a network topology and traffic necessities, we aim to find the location where SDN functionalities should be installed as well as a link weights configuration that minimizes the network congestion evaluated with the metric Φ^*.

A solution for this problem is encoded as a vector of size $m + n$, where m is the number of links and n the number of nodes, Figure 4.6. The m first values of the solution vector are link weights with values in range $[w_{min}, w_{max}]$. The last n vector

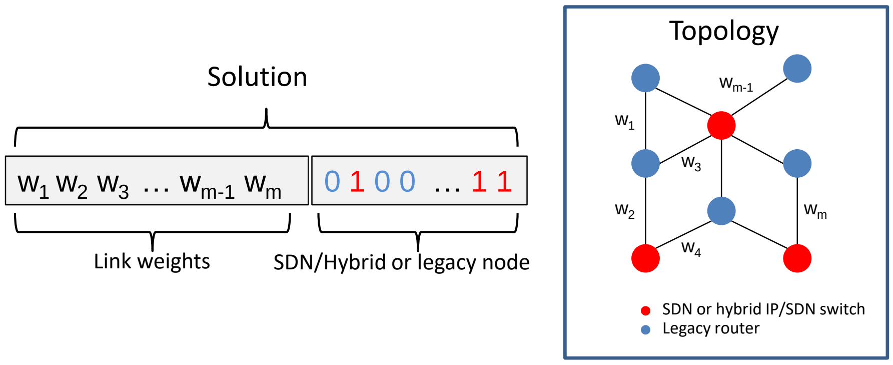


Figure 4.6: *Solution representation for hybrid IP/SDN incremental deployments*

values identify if a node i, $i = 1..n$, is either a SDN or hybrid IP/SDN node, with value 1, or a legacy node, with value 0. The node type component of the solution vector is a binary list of n elements with k values equal to 1, and the remaining values are equal to 0. To evolve the solutions, the EAs use two types of mating and mutation operators, the integer operators, described in section 3.2.4, applied to the first m genes of the solution vector, and permutation mutation operators that are applied to the last n values. The permutation mutation operators are adjacent and not adjacent swaps. Additionally, a hybrid crossover operator inter-exchanges the link weights configuration and node type configuration from two parents producing two new offspring solutions. The probability of applying permutation mutations decreases as $|n - k|$ increases recognizing that the number of possible permutations of node types also decreases.

As usual, we devised a set of experiments to evaluate the optimization method considering distinct network topologies. The number k of hybrid nodes to be deployed varies taking values of 10, 20, 30 and 40% of the total number of topology nodes. We also include results where all nodes are legacy routers, that is 0% of hybrid nodes, which corresponds to the scenario of running only the IGP with an optimized link weights configuration, as well as results where all nodes are hybrid IP/SDN switches.

Traffic demands have an expected averaged resources utilization of 30%, D0.3, and were customarily generated at random. The results presented in Table 4.2 are the mean of 30 runs, each with 1500 iterations.

The results show that not all nodes need to implement SDN functionalities to attain congestion values equivalent to those obtained when all nodes are hybrid. A percentage of 40% of hybrid IP/SDN nodes, or even less, is sufficient to achieve such congestion levels. With only 20% it is already possible to observe a significant improvement in congestion. The algorithm tends to select as hybrid nodes with the higher in/out degree and contribute more significantly to the TE tasks. Nodes with an in/out degree 2 have a

Table 4.2: *Congestion by percentage of hybrid IP/SDN nodes for the (DEFT).*

Topology	0%	10%	20%	30%	40%	100%
$Rand30_2$	1.42	1.40	1.39	1.38	1.38	1.38
$Rand30_4$	2.70	2.48	1.90	1.86	1.72	1.71
$Rand50_2$	1.68	1.66	1.64	1.63	1.62	1.61
$Rand50_4$	3.67	2.45	2.41	2.4	2.39	2.38

minor impact on improving traffic distribution, as their role is mostly to act as transient nodes. They only perform traffic load balancing when acting as an edge node. Therefore topologies with a higher average in/out degree, such as $Rand30_4$ and $Rand50_4$, have a greater congestion gap between using 0% and 100% of hybrid nodes and reveal more improvement on using more hybrid IP/SDN nodes.

4.5 Traffic Variation

Traffic patterns and volume change over time, and a routing configuration that is suitable for particular traffic necessities, modeled as a traffic matrix, can become inadequate for new demands. In the previous chapter, we addressed the problem by resorting to multi-objective optimization and showed that the Non-dominated Sorting Genetic Algorithm (NSGAII) provides good configuration solutions for foreseen traffic demands. However, it is not always possible to predict, with the necessary accuracy, future traffic requirements and, therefore, such an approach is not always sufficient. If an installed routing configuration ceases to be suitable, it would be necessary to update routing decisions by performing a new optimization which would provoke a temporary network instability. Moreover, as solutions are integer representations of size $|E|$, where E is the number of edges on the topology, the number of iterations required by the EA to converge increases with the size of the network topology. Larger network topologies have larger solutions search spaces and the computation time required by each iteration increases significantly. Consequently, frequent reoptimizations become almost prohibitive within the required timescale.

The optimization model for hybrid IP/SDN networks, proposed in Section 4.3, offers an alternative to handle some changes in traffic demands. Node-p variables can be manipulated to correct traffic load balancing on non-shortest path links so that the network congestion decreases. Each network node has an associated node-p value. Thus a node-p configuration can be encoded as a size $|N|$ vector of real values in the range

[0.01,10], where N is the set of topology nodes. The significantly smaller search space allows obtaining good node-p configuration solutions in an acceptable amount of time.

To evaluate the proposed optimization method, we devised a collection of experiments where traffic variations undermine the network performance. The first set of scenarios considers that the hybrid IP/SDN network has a routing configuration optimized considering a single traffic demand matrix. In the second set of experiments, the network is initially configured considering two distinct traffic matrices and a solution that simultaneously minimizes congestion for both traffic necessities was obtained using the NSGAII algorithm. In both cases, the initial network configuration is set with a default node-p value of 1, assigned to all network switches.

The variation of traffic necessities embodies a new optimization problem that can be formulated as follows: Given a network topology and new traffic necessities, modeled as a TM, for which the already installed hybrid IP/SDN configuration is not satisfactory, the aim is to find a node-p values configuration that minimizes the network congestion for the new traffic conditions. To reduce the EA search space, each node-p variable takes values in the real subset $R = \{p = a/100 : a = 1..1000\}$ and EA solutions are thus encoded as a set of integer in range [1,1000]. We use a single objective EA with the same running configuration as the one for link weights optimization.

4.5.1 Single Traffic Demands

Table 4.3: *Node-p values optimization - Single traffic matrix.*

Measure	$Rand30_2$						
	D_0	D_1			D_2		
	Opt.	Before	After	Opt.	Before	After	Opt.
Congestion	1.39	8.61	3.28	1.49	16.45	5.93	1.76
Avg Link Utilization (ALU)	35%	37%	37%	37%	43%	44%	43%
Max. Link Utilization (MLU)	75%	121%	115%	76%	144%	117%	90%

Measure	$Rand50_2$						
	D_0	D_1			D_2		
	Opt.	Before	After	Opt.	Before	After	Opt.
Congestion	1.59	8.22	1.75	1.83	31.29	13.46	11.18
Avg Link Utilization (ALU)	37%	38%	38%	37%	50%	51%	49%
Max. Link Utilization (MLU)	94%	130%	100%	101%	155%	132%	121%

The aforementioned traffic distribution problem was addressed, in the first set of experiments, in networks whose initial configuration was optimized for a single traffic matrix. Each network, $Rand30_2$, with 30 nodes and 110 links, and $Rand50_2$, with

50 nodes and 194 links, was configured with a hybrid IP/SDN routing configuration (DEFT) which minimizes the congestion metric Φ^* for a random demand matrix D_0 generated with the scheme describe in Section 2.4. New mutable traffic conditions, represented as two new traffic demand matrices, D_1 and D_2, are applied to the network. Both TMs D_1 and D_2 were randomly generated using the same scheme and have a linear correlation of approximately 0.5 with D_0. The installed routing configuration, being unsuited for the new traffic demands, precipitates a poor distribution of traffic into the network resources, causing severe packet loss. The congestion values (Φ^*), Average Link Utilization (ALU) and Maximum Link Utilization (MLU) are used to evaluated the network conditions in such state and are identified in Table 4.3 as "*Before*", that is, before any action is taken. To improve the distribution of traffic, node-p values are optimized and the results presented as "*After*" in Table 4.3. For comparison, we also include the same metrics with optimized configurations, that is, link weights optimization with default node-p values, for the initial and new traffic demands (*Opt.*).

The results show that by solely optimizing the node-p values, the congestion levels in the network decay in all scenarios. In some cases, the network is able to accommodate the changes in traffic and continue performing efficiently simply by reconfiguring the traffic load balancing ratios for non-shortest path links. This approach imposes a disruption on the forwarding paths of a reduced percentage of flows. Furthermore, it keeps the link weights unchanged and can be deployed in a relative short amount of time. In fact, new node-p values configurations for the presented examples were obtained in approximately 30 seconds for the $Rand30_2$ topology, and 100 seconds for the $Rand50_2$ topology. Such a solution is intended to transient conditions, while temporary traffic changes are imposed.

4.5.2 Two Traffic Demands

In Section 3.4, we discussed briefly the advantages of optimizing a link weights configuration for two TMs simultaneously, D_1 and D_2, which allows to maintain the network in good (or satisfactory) operational conditions in two different moments in time with distinct traffic requirements. Additionally, if a configuration is suited for both traffic matrices, it is expected that most traffic matrices D whose values are bounded by D_1 and D_2 will also be routable with the same configuration [40], that is, with a favorable utilization of the network's infrastructure. Futhermore, flow measurements can be difficult to obtain and estimated origin/destination flow aggregates are not always available [12]. In such a context, the authors in [8] discussed how to route flows with limited knowledge or no knowledge of the traffic matrices. In theory, they proved that a semi-oblivious routing can be achieved using two TMs that bound a TMs space. In such a case, the scheme in Equation 3.4, which embodies a tradeoff between the network operational conditions in both moment, can be used to obtain a link weights configuration robust to variations within a set of TMs bounded by D_1 and D_2. Although such result is significant in itself, it can be further improved in the context of hybrid IP/SDN deployments.

A configuration solution for hybrid IP/SDN, that considers a tradeoff between two distinct TMs, is not tied to any particular TM and, therefore, can always be improved by adjusting load balancing ratios, in this case, by optimizing the node-p values. In such a context, we devised a set of experiments where the goal is to obtain a hybrid IP/SDN configuration that allows the network to properly route a set of distinct TMs. For each network topology, $Rand30_2$ and $Rand50_2$, we generated a set M of 100 random TMs with expected mean links capacity utilization in the range [0.3,0.4]. From the set of M of TMs we obtained two traffic demand matrices, D_1 and D_2, following three distinct methodologies:

- **Demand matrices with bounding values (2-BV)**:
  To obtain D_1 and D_2, we first define two auxiliary TM, D_{max} and D_{min}, which for each pair of nodes (i, j) contain the maximum and minimum traffic requirements from i to j. The two traffic matrices are than obtained by randomly swapping all values from both matrices D_{max} and D_{min} with a probability $p = 1/2$. The two TMs are consequently opposite vertices of the higher-dimensional polyhedra which bounds the TMs space. This method allows to produce two traffic matrices in the same range of expected mean links utilization while confining all origin/destination traffic requirements.

- **Demand matrices as k-means centroids (2-KM)**:
  K-means [59] is a clustering algorithm based on minimizing a formal objective function. Given a set of n data points in real n-dimensional space, $\mathbb{R}^n$, and an integer k, the problem is to determine a set of k points in $\mathbb{R}^n$, called centroids, so as to minimize the mean squared distance from each data point to its nearest centroid. In this case, the set of data points is the set M of traffic matrices which is divided in two k-means clusters ($k = 2$) considering as distance between two traffic matrices the euclidean distance applied to their vectorizations. The traffic matrices D_1 and D_2 are the centroids of the obtained clusters. The idea is to group all TMs which share the most similarities into two groups, clusters, and from each group select a representative TM, the centroid.

Table 4.4: *Comparison of traffic variation approaches (mean congestion Φ^* values)*

Topology	Single TM initial optimization				Two TMs initial optimization					
	1-AM		1-RD		2-BV		2-KM		2-RD	
	Without	With	Without	With	Without	With	Without	With	Without	With
$Rand30_2$	5.79	2.81	6.26	3.21	3.60	2.55	6.70	2.75	6.59	3.31
$Rand50_2$	9.39	5.80	15.97	11.66	7.13	5.08	7.02	4.80	6.48	4.67

- **Random demand matrices (2-RD)**:
  The two TMs D_1 and D_2 are randomly select from the set M of all TMs.

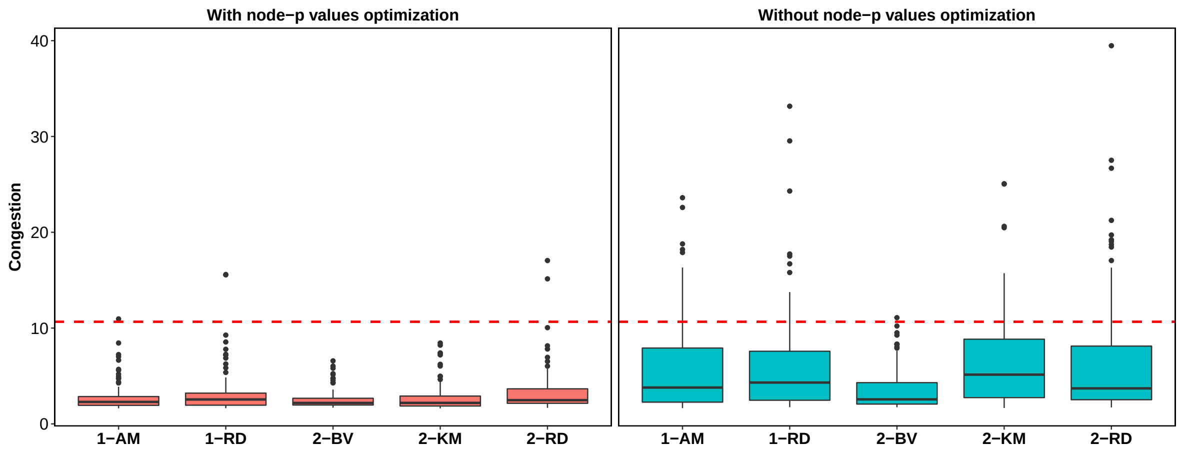
We also want to compare hybrid IP/SDN configuration solutions that consider two TMs in the optimization process with those which consider only one TM. Considering the same set M of traffic demands, we defined two methodologies to select a single TM D_0 to be used as argument in the single objective congestion optimization:

- **Random demand matrix (1-RD)**:
  The traffic matrix D_0 is randomly selected from the set M of all TMs.

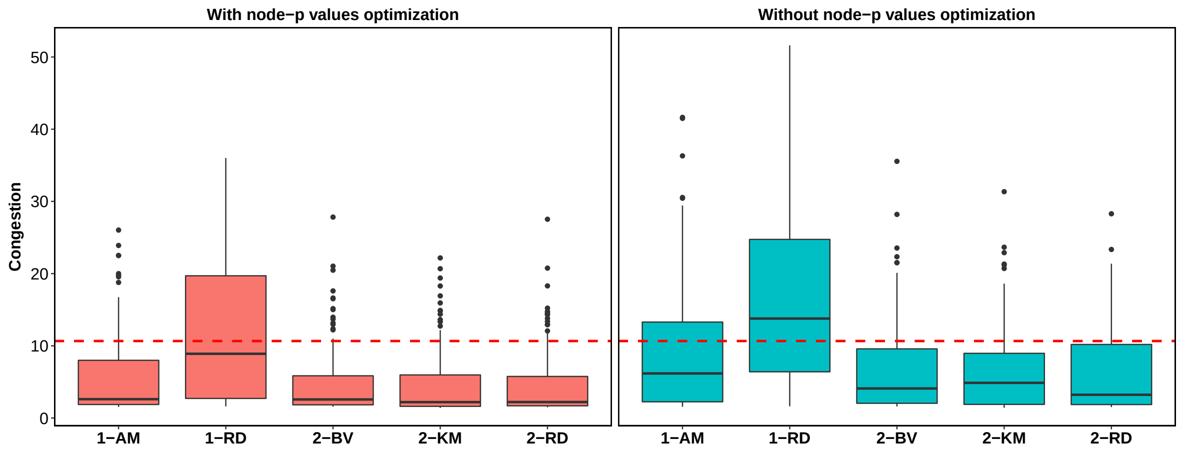
- **Arithmetic mean matrix (1-AM)**:
  The traffic matrix D_0 is the mean of all TMs in M.

The initial hybrid IP/SDN configurations were obtained by running each EA with a stopping criterion of 1500 generations (SOEA for optimizations with one TM and NSGAII for optimizations with two TM). The configuration with the smallest tradeoff value $\alpha \times \Phi_1^* + (1-\alpha) \times \Phi_2^*$ is selected from the Pareto optimal set, where Φ_1^* and Φ_2^* are the congestion measure Φ^* when considering, respectively, the TMs D_1 and D_2, and α is a real value in range [0;1] which depends on the TMs selection procedure. While α is 0.5 for the 2-BV and 2-RD methods, for the k-means TM selection procedure α is proportional to the number of elements in each of the two clusters. For each configuration solution, the congestion measure Φ^* is used to evaluate networks resources utilization for all TMs in M. Additionally, node-p values are optimized for all TMs and the networks resources reevaluated. Results for random matrix selection (1-RD and 2-RD), as well as for the kmeans matrix selection method (2-KM) are, for each network, the best of 10 runs.

The distributions of congestion values Φ^* for each scenario are presented in Figure 4.7 and the mean values in Table 4.4. An observation which can be immediately drawn is that, independently of the initial optimization procedure, number of objectives and matrix selection method, the optimization of node-p values always translates into a significant improvement on network resources utilization (with vs. without node-p optimization in Figure 4.7). Another important observation is that hybrid IP/SDN



(a) *Rand*30_2 *network topology*



(b) *Rand*50_2 *network topology*

Figure 4.7: *Traffic variation with single and two traffic demands initial configurations*

configurations obtained from the optimization that considers a random selection of only one TM, 1-RD, is always among those that provide worst results. In the particular case of the *Rand*50_2 topology, before a node-p values optimization, more than 50% of the TMs originate congestion values above the operating threshold. The boundary of the acceptable operating region of the network is identified in Figure 4.7 by the dashed red line, which corresponds to a congestion value Φ^* of 10 2/3.

With respect to initial configurations optimized for two traffic matrices, the TMs selection method that bounds the set M (2-BV) and the kmeans method (2-KM) behaved consistently in both topologies and provided the best results before and after traffic load balancing corrections. However, the kmeans method (2-KM) is heavily dependent on the initial selection of centroids and thus might not always produce satisfactory results. The random selection of the TMs has the same limitation, and although the node-p values optimization is able, within a certain extent, to minimize the effect of such behavior, it might not be always the case. Consequently, non stochastic methods, such as the selection of TMs that bound the TMs space, are preferable.

The optimization of node-p values allows no narrow the number of TMs that are not sustained by the installed optimizaed network configuration. However, a correction of load balancing is not always suficcient to restore the network to satisfactory operational condiditons and thus additonal messures need to be taken. An example of such measures is to use SDN functionalities to forward portions of traffic along paths computed with the Edmonds–Karp algorithm [63], or resorting to any other greedy algorithm that computes the maximum flow in a network.

4.6 Single Link Failure

Hybrid IP/SDN networks present an additional advantage. On the occurrence of a link failure, the IGP automatically reroutes the affected traffic. However, since this rerouting is blind to congestion, and affected traffic is only rerouted along recomputed shortest paths, massive packet drop may occur due to congestion originated by the fault. SDN offers different alternatives to address this problem. For example, alternate paths can be preinstalled for each individual flow [64]. However, in most cases, such approach results in storing thousands of forwarding rules. Furthermore, the dependence on the controller for dynamic per-flow detouring may delay the recovery. In [20] the authors developed an approach for hybrid IP/SDN networks, based on ECMP to deal with link failures. To reduce post-recovery congestion, the router at the point of recovery reroutes the affected traffic to designated SDN switches across IP tunnels on equal cost paths. Such an approach requires that designated SDN switches be optimally placed and link weights configured accordingly.

The hybrid IP/SDN model explored in the previous sections, and node-p values optimization, in particular, provide an alternative approach. In case of a link failure, the end-to-end connectivity is automatically re-established by the IGP and traffic is forwarded along the new computed shortest paths. This redistribution of traffic may lead to congestion and consequently we propose to correct load balancing ratios for the new network conditions by optimizing node-p values. This approach has the advantage of altering a small portion of traffic flows routing paths, instead of altering potentially all routing paths by reoptimizing link weights. Results for this approach are presented in Table 4.5 and Figure 4.8. We considered distinct hybrid IP/SDN networks, where all nodes possess hybrid functionalities, which were initially optimally configured using the DEFT load balancing mechanism. As usual, traffic routed along shortest paths is managed by the IGP and non-shortest paths forwardings are managed with SDN. We considered traffic matrices that use approximately 30% of networks resources. An experiment contemplates the failure of all network links, one at a time, and for each failure, the node-p values are optimized 10 times with a stopping criterion of 100 iterations. In turn, each experiment was ran for 10 times using distinct optimized link weights configuration. The mean network congestion values for all experiments are presented in Table 4.5

Table 4.5: *Mean congestion with and without node-p optimization after a link failure for D0.3 traffic matrices.*

Topology	$Rand30_2$	$Rand30_4$	$Rand50_2$	$Rand50_4$
Without node-p optimization	7.94	4.62	7.36	6.52
With node-p optimization	6.29	2.16	5.80	3.72

Results show that by solely optimizing the traffic load balancing on non-shortest path links it is possible to reduce congestion significantly. The improvement on congestion increases as the average node in/out degree also increases as more traffic is forwarded along non-shortest path links. The benefit of load balancing correction can be further observed in Figure 4.8, where congestion values observed after the link failure without any additional measure are labeled as "Without", while the congestion values after the node-p optimization are displayed as "With". The red lines in Figure 4.8 identify the network congestion operational threshold (10 2/3).

The number of link failures which result in over congested networks is far lesser after optimizing node-p values. For example, for the $Rand50_4$ topology without load balancing correction 15.3% of failure results in congestion above the network functional threshold against 4.7% with load balancing correction, and for the $Rand30_4$ those percentages are respectively 10% and 2.7%. Although this method does not always assure a satisfactory network congestion level, its benefits are undoubtedly significant.

4.7 Algorithms Running Time

The time needed by an algorithm to produce a solution is an important factor. While the optimization of link weights, and thus of hybrid IP/SDN configurations, is as task performed offline, the node-p values optimization is intended as a reactive measure to address changes on traffic requirements and also to adjust the installed configuration to

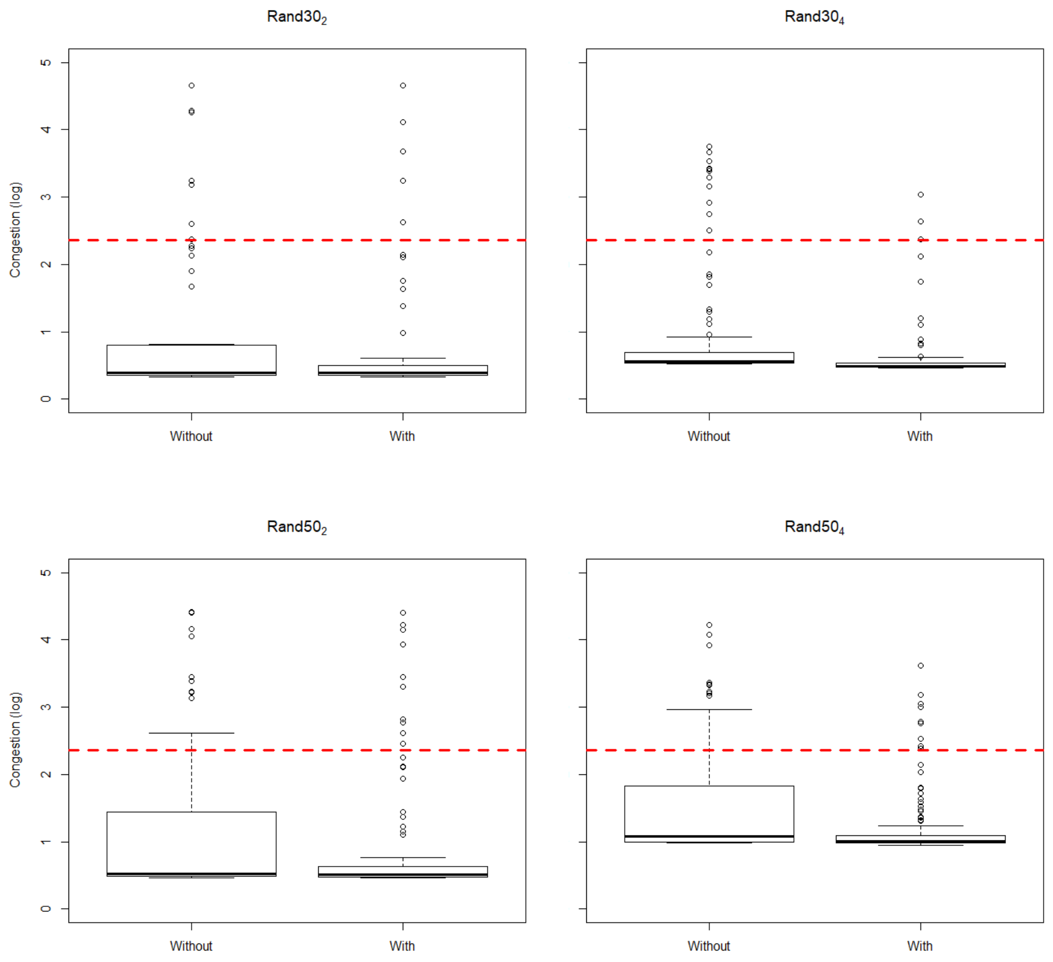


Figure 4.8: *Congestion distribution with and without node-p optimization after a link failure for D0.3 traffic matrices.*

alterations on traffic forwarding caused by a link failure. The last, to be considered as a possible reactive measure, needs to be performed online within an acceptable amount of time. Each iteration time depends on the size of the network, that is, on the number of topology nodes and links, but also on the used EA an its settings (e.g., population size). The running time for each algorithm per iteration in seconds, with the EAs settings describe in the experiments, is presented in Table 4.6.

Although each iteration of link weights and node-p values optimizations are very similar in the SOEA, the last requires much less iterations to converge. All results that involved a node-p values optimization required 100 iterations. However, after 50 iterations the algorithm already provides acceptable solutions. A network administrator can therefore in a reduced amount of time test if a node-p values optimization can solve an occurring congestion problem.

Table 4.6: *Algorithms Running Time (in seconds per iteration).*

Topology	DEFT Weigths		PEFT Weights		Node-p Values	
	SOEA	NSGAII	SOEA	NSGAII	DEFT	PEFT
$Rand30_2$	0.31	1.51	0.32	1.68	0.32	0.33
$Rand50_2$	1.45	6.94	1.57	7.63	1.47	1.62

4.8 Conclusions

Hybrid IP/SDN networks combine the benefits of traditional link state routing with the flexibility of Software Defined Networking. SDN has added the ability to move network management to a centralized controller allowing to exercise per-flow traffic engineering and better usage of networks' infrastructures. However, SDN has scalability issues and added complexity in managing network functions. Hybridizing SDN with a link state protocol, such as OSPF, solves the scalability problems and attenuates complexity as not all flows need to be individually managed. In this context, we proposed a congestion optimization scheme for hybrid IP/SDN networks which makes use of non-shortest path links to forward portions of traffic and thereby provides a better usage of network links capacity. The routing decision method has the advantage of always forwarding traffic towards the destination, i.e., the shortest path distance to the destination decreases after each hop. The model also simplifies switch entries management. Traffic that travels along shortest paths is managed by the IGP while SDN manages the unequal traffic load balancing and non-shortest path forwarding.

The proposed model, coupled with an EA engine, supplies an optimization method for traffic distribution on hybrid IP/SDN networks capable of attaining near-optimal configurations. Although most validation experiments were run considering that all nodes have SDN and legacy switch functionalities, we also showed that the model is valid for hybrid deployments where only a subset of nodes have both functionalities. Additionally, we introduced a new set of variables, node-p values, that allow to alter traffic load balancing without changing forwarding paths. By optimizing traffic load balancing ratios on non-shortest links, that is, by optimizing the node-p values, the congestion introduced by events that alter networks operational conditions can be effectively addressed. We experimentally showed that, in most cases, and especially when nodes have a higher in/out degree, traffic load balancing corrections are sufficient to address congestion problems that result from changes in traffic demands or link failures.

Chapter 5

Segment Routing: Traffic Engineering with Three Segments

Segment Routing (SR) combines the simplicity of Link-State routing protocols with the flexibility of Multiprotocol Label Switching (MPLS). By decomposing forwarding paths into segments, identified by labels, SR improves Traffic Engineering (TE) and enables new solutions for the optimization of network resources utilization. This chapter proposes an Evolutionary Computation approach that enables Path Computation Element (PCE) or Software-defined Network (SDN) controllers to optimize label switching paths for a better utilization of networks' resources while using at the most three labels to configure each label switching path. The flexibility introduced by Segment Routing enables new responsive and dynamic methods to respond to changes on the network operational conditions, such as changes on the traffic necessities, as well as fault events, such as link failures. The SR model proposed in this chapter would not be complete without addressing such issues. Ergo, the proposed model is extended to enable traffic load balancing corrections between end-to-end SR paths and better accommodate the new traffic necessities. Additionally, typical responses to single link failures only aim to reestablish loop-free connectivity between affected routers and do not account for impacts in network congestion. This chapter also addresses the single link failure congestion problem, and compare post-convergence network congestion values obtained with distinct Evolutionary Computation based recovery paths computation methods with the solution adopted by the industry.

5.1 Introduction

Traffic engineering techniques evolve with the emergence of new methods and trends. However, its primary goal remains unchanged, to efficiently allocate network resources to meet user constraints while increasing operators' benefits. Although there are a vast number of proposals which aim to achieve this end, they only can be viable if two key attributes are preserved: simplicity and scalability.

Since Link State (LS) routing protocols were firstly proposed, network engineers have given them particular attention. They intrinsically possess those two essential attributes, and this is one of the reasons why, after almost 40 years, they continue to be widely deployed. Their simplicity lies in the fact that they only require a set of weights assigned to each network link to compute the forwarding paths, the shortest paths.

Most implementations of LS routing protocols use Equal Cost Multi-Path (ECMP) to achieve a better usage of the available network resources. When more than one shortest path exists to the same destination, ECMP enables to split traffic along the next-hops on the paths evenly. Nevertheless, although LS/ECMP routing with optimized weights configurations enables a proper distribution of traffic load, it can not deliver an optimal usage of a network's resources as shown in previous chapters. The even splitting of traffic across multiple shortest-path routes performed by ECMP is an obstacle to an optimal resources utilization. In some cases, even with optimized configurations, a set of network links might not be used at all.

To address this specific issue, unequal load balancing of traffic among outgoing links needed to be considered. Many proposals were presented able to implement unequal load balancing [103, 105], but failed in preserving the simplicity and scalability of what would be expected from desired routing protocols. The Distributed Exponentially-weighted Flow SpliTting (DEFT), on the other hand, as well as the Penalizing Exponential Flow-spliTting (PEFT), can forward traffic along links on non-shortest-paths and induce unequal traffic splitting, without loosing in simplicity and scalability. By solely relying on a link weights configuration, DEFT/PEFT assigns flows to a next-hop with a probability that decreases exponentially with the extra length of the path relative to the shortest path. Chapter 4 experimentally showed that DEFT and PEFT were capable of offering a better utilization of resources than the one provided by optimized OSPF or IS-IS with ECMP in hybrid IP/SDN networks. Additionally, the contribution made in Section 4.2.4 introduced an additional mechanism able to handle most traffic variations as well respond to single link failures. However, DEFT and PEFT have some drawbacks. Firstly, a packet may travel from source to destination using solely non-shortest path links. Secondly, in an SDN environment implementation, as the one presented in the previous chapter, switches need to maintain a per-flow status.

Segment Routing (SR) [33] was recently proposed as an SDN technology and briefly introduced in Section 2.2.6. As an implementation of the source routing paradigm, SR adds important simplifications to the data plane and the control plane operations. SR decouples edge-to-edge routing paths into smaller paths called segments. Analogously to Multi-Protocol Label Switching (MPLS) [78] or a path shim header in IPv6, SR uses a path-label mechanism to specify the route that packets must take through a network. Thus, in this technology, a route is uniquely defined as a list of segment IDs (SIDs). This strategy provides an enhanced packet forwarding while minimizing the need for maintaining awareness of mass volumes of network states. In fact, instead of pushing a flow entry to all the switches in a path, SR pushes a label stack, SIDs, into the packet header when it arrives at the ingress nodes.

In this context, this chapter proposes TE solutions for SR able to deliver a near optimal use of network resources. The main characteristic that ties this contribution is that routing paths need at most three labels to be configured. We will also explore the utilization of a split computation parameter to respond to variations in traffic requirements and avoid congestion in parts of the network due to changes in traffic demands. Additionally, the flexibility introduced by SR also enables new responsive and dynamic methods to address network fault events, such as link failures. Typical SR responses to single link failures only aim to reestablish loop-free connectivity between affected routers and do not account for impacts in network congestion. As Internet routing protocols recompute shortest-paths after a link failure, traffic that traveled over the failed link is rerouted influencing the overall network congestion. To address this problem, we will compare post-convergence network congestion values obtained with distinct Evolutionary Computation based recovery paths computation methods for Segment Routing.

5.2 Segment Routing Traffic Engineering

5.2.1 Segment Routing

Segment routing is a simple, scalable and highly flexible platform. SR is a label switching technique that allows edge routers to steer a packet through the network by using a list of segments. Each segment identifies a topological or a service instruction. While a service instruction pinpoints a service, provided by a node, where a packet should be delivered, a topological instruction identifies a path across which a packet should be forwarded. Any edge-to-edge path can be fully identified using one or a combination of topological segments. Although SR is very similar to MPLS, it neither requires a Resource Reservation Protocol (RSVP) or a Label Distribution Protocol (LDP), which makes it much more flexible. An extension assures the distribution of labels to the Interior Gateway Protocol (IGP), IS-IS or OSPF, or centrally managed by a Path Computation Element (PCE) or a SDN controller. The path information required for a packet to traverse the network is a list of segments IDs (SID), which is allocated in the packet's header at the provider's edge (PE).

In the context of the proposal made in this work only two of the available types of segments IDs are required:

- **Node SID**: A Node Segment identifies a network Node, a destination that should be reached using IGP shortest paths.
- **Adjacency SID**: An Adjacency Segment represents a local segment (interface) to a specific SR node.

A key advantage of SR is that, contrary to a SDN implementation, intermediate routers do not have to maintain any per-flow state. The intermediate nodes only need to know the globally distributed segment labels. Additionally, SR supports ECMP by design. Paths identified by node segments are IGP shortest-paths, and they intrinsically include all the ECMP paths to the node. These two features provide tremendous gains in network scalability.

Exploring the mentioned advantages, we propose an optimization mechanism for Segment Routing able to deliver near-optimal traffic distribution resorting to label path configurations which contain at most three segment IDs, with at the most one adjacency SID.

5.2.2 Three-Segments SR Paths

The problem of determining optimal configurations for Segment Routing using the least number of segments in the SR header has been approached in some seminal works. In [15] the authors propose offline, online and traffic oblivious optimization algorithms to minimize the the Maximum Link Utilization (MLU) for a given IGP link weights configuration. The proposal idea is to find k intermediates nodes on non shortest paths to the destination that enable to optimize the network traffic load balancing. This approach, which only uses nodal segments to configure SR paths result in longer hop-by-hop paths which can contribute to a higher delay. On the other hand, the online optimization approach is based on a flow acceptance mechanism where traffic is only forwarded if a k-segment path that supports the flow traffic requirements is available. Such approach has two main inconvenients. It requires predetermined knowledge of each arriving flow routing necessities, which is not always possible, and the routing path availability needs to be verified for each new flow, which may not be accomplishable for large networks. The authors show that, for unit link weights and random IGP configurations, the optimization of load balancing with 2-segments provides a significant improvement on the MLU.

In [42] the authors propose a local search algorithm able to quickly rearrange a few forwarding paths such that the MLU is minimized. Such an approach needs to be frequently run disrupting for each solution already configured paths and without any warranty that those few changes will be sufficient to prevent congestion. The authors in [73] also propose an integer programming algorithm as well as a heuristic approach to assess the traffic engineering of packet networks with SR. By defining a maximum label stack depth, the heuristic begins by distributing traffic on SR path with one SID (shortest paths) and consecutively distributes traffic on SR paths with increasing segment list depth. This greedy approach, although providing an improvement on congestion, is unable to deliver (near) optimal configurations as it only provides a locally optimal solution. Nonetheless, in all cases, results indicate that SR does not require a high label stack depth to perform well.

The proposed algorithm uses a different approach. The main idea is to attain a near optimal resources utilization using forwarding paths which deviate at the most one hop from the shortest path. Such an objective is accomplished using only label paths with the following alternatives of configuration formats:

- **1-Segment**: The label path is configured with a unique Node SID, the SID of the destination node t, $[\mathtt{t}]$ (Figure 5.1b);

- **2-Segment**: The label path is configured with a Node SID and an Adjacency SID. In this case there are two possible configurations:

    1. $[(\mathtt{s},\mathtt{u});\mathtt{t}]$, the adjacency segment starts at the source node s, and the Node SID identifies the destination node t (Figure 5.1c);
    2. $[\mathtt{v};(\mathtt{v},\mathtt{t})]$, the adjacency segment ends at the destination node t, and the Node SID identifies the start node of the adjacency segment, v (Figure 5.1d);

- **3-Segment**: The label path is configured with a Node SID, an Adjacency SID, and a Node SID,$[\mathtt{u};(\mathtt{u},\mathtt{v});\mathtt{t}]$, where (u, v) is the adjacency segment (Figure 5.1e).

Each SR path is configured with at most one adjacency segment and is called Single Adjacency Label Path Segment Routing (SALP-SR).

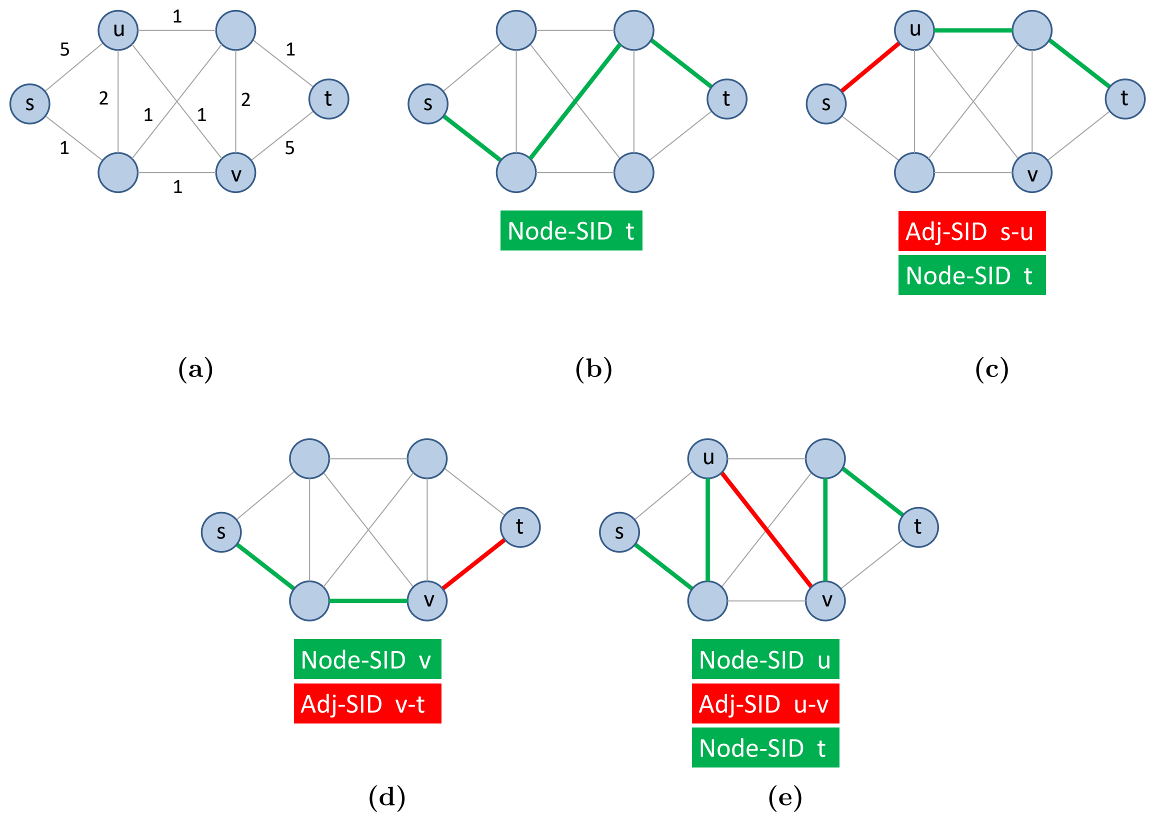
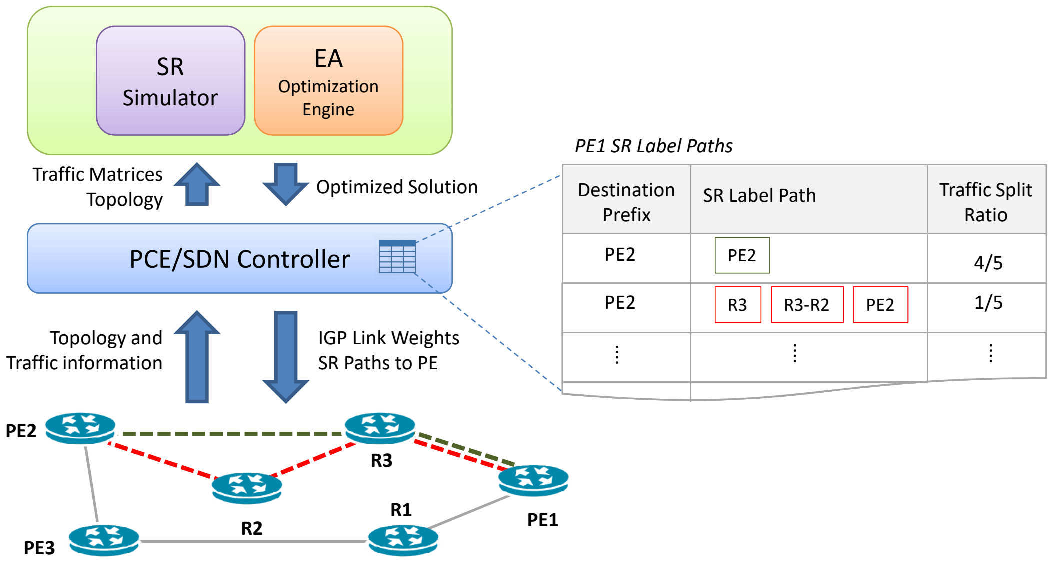


Figure 5.1: *SR path configuration formats.*

5.2.3 Traffic Engineering with Three-Segments

Network infrastructures are constantly being challenged by strong growth in traffic volume which needs to be fairly distributed into the available resources. Segment Routing is built over an IGP. As a consequence, any SR configuration solution must also include the IGP's configuration, i.e., link weights. Furthermore, to make the most of SR features, configurations must also include possible edge-to-edge parallel paths and the load balancing of traffic between them. Thus, to achieve optimal resources uti-



Destination Prefix	SR Label Path	Traffic Split Ratio
PE2	PE2	4/5
PE2	R3 R3-R2 PE2	1/5
⋮	⋮	⋮

Figure 5.2: *Conceptual architecture of the optimization model.*

lization, any optimization procedure for Segment Routing must provide solutions that encompass:

1. a link weight configuration for the link-state IGP that underlies the SR network,
2. edge-to-edge SR label paths configuration and
3. the split of traffic between parallel label paths to the same destination.

General Architecture

A conceptual representation of the SALP-SR optimization model elements is presented in Figure 5.2. The optimization model assumes that traffic necessities are known and were obtained, for example, using one or more of the methods described in section 2.4.

As depicted in Figure 5.2, the optimization is performed by a northbound application which integrates an SR simulator and an Evolutionary Algorithm (EA) optimization engine. The northbound application interacts with a PCE/SDN controller that collects topology and traffic-related information. During the optimization process, the distribution of traffic is achieved by resorting to DEFT or PEFT traffic splitting functions (described in Section 4.2). The SR routing simulator is also used to evaluate the quality of the solutions provided by the EA.

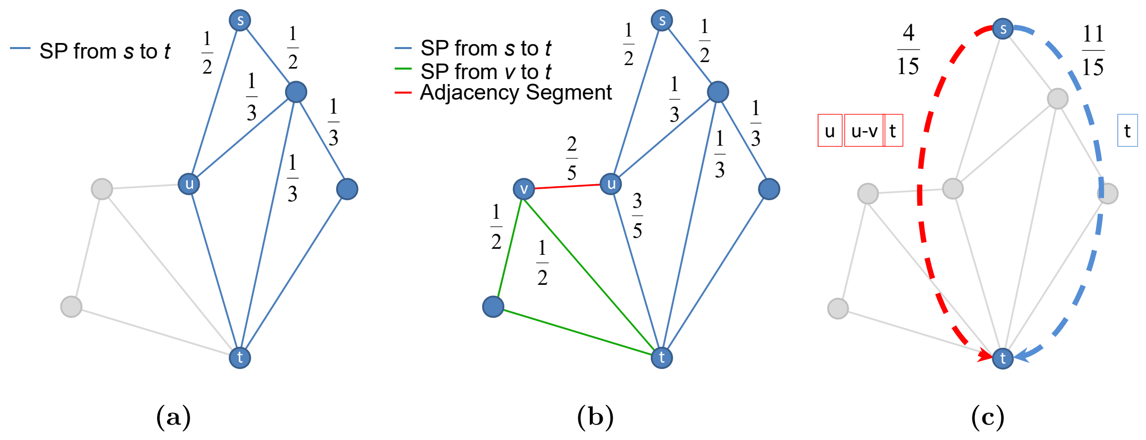


Figure 5.3: *Three-Segments Routing example.*

As observed in Figure 5.2, the optimized solution attained by the northbound optimization framework will be then installed in the network by the PCE/SDN controller. This task involves installing the links weights configuration for the IGP operation, the SR label paths and traffic splitting fractions for provider edges operations. In Figure 5.2 traffic originating in PE1 with destination PE2 is forwarded along two parallel SR paths, a shortest path and a path which makes use of an adjacency segment. Traffic is divided between both paths on a 4/1 ratio.

Illustrative Example

For an easier comprehension of the devised TE proposal for SR, we present an illustrative example.

Figure 5.3a represents part of a network topology, configured with OSPF or IS-IS, that corresponds to the shortest path tree from node s to t, and the traffic splits that resulted from a given weigh setting configuration. The amount of traffic that arrives to node u is 2/3 $(1/2{+}1/2{\times}1/3)$ of the traffic with destination t, which would be forwarded along the link (u,t). If (u,t) is unable to accommodate such volume of traffic, packet losses and increased delays will certainly occur. To prevent such scenario, a portion of the traffic arriving at u can be forwarded using an adjacency segment that does not belong to any shortest path from u to t. In such a case, and considering the already computed optimized hop-by-hop load balancing ratios, Figure 5.3b, 2/5 of the traffic from s to t arriving at u, that is $4/15 (= 2/5 * 2/3)$ of the total traffic, will be forwarded using the adjacency segment (u,v).

By optimizing the distribution of traffic, using DEFT (or PEFT) load balancing function and the EA optimization engine (of Figure 5.2), an optimal solution for traffic distribution is obtained and translated into an IGP link weights configuration, SR paths, and traffic spitting ratios, Figure 5.3c.

SALP-SR Mathematical Model

The routing optimization model uses undirected graphs $G\,(N, A)$ to represent network topologies, where N is a set of nodes and A a set of arcs. For a given traffic demand matrix and a routing configuration, the congestion level of a network is evaluated using the normalized sum of link cost Φ^*, introduced in Section 2.3.2. The first objective of SALP-SR is to find a configuration W, a set of integer values, which minimizes the normalized sum of all links' congestion, Φ_a, i.e, $min\Phi = \sum_{a \in A} \Phi_a\,(u(a))$. Other additional optimization objectives can also be considered such as the network maximum link utilization (MLU) or other restrictions.

A solution W is nothing more than a set of IGP's integer link weights $w_{u,v}$, $u, v \in N$, but from which the optimization algorithm also derives the other additional settings, such as the SR paths and load balancing ratios between parallel paths.

To compute SR paths, and later load balancing ratios, the optimization algorithm forwards packets from a flow with source s and destination t by applying a penalization on longer paths. Traffic at a node u, in a path from s to t, is forwarded to a next-hop v with a probability that decreases exponentially with the extra length of the path to t, $h^t_{u,v}$. This computation is depicted in Equations 4.1 and 4.2, proposed in [119], where d^t_u is the shortest path distance from u to the destination t. As explained, the portion of traffic at u with destination t to be routed to next-hop v is computed by the proportion function Equation 4.3. This traffic splitting method enables the optimization algorithm not only to take advantage of IGP's Equal Cost Multi-Path (ECMP) but also to use non-shortest-path links to forward traffic while ensuring that packets are always routed towards the destination.

SR paths are obtained based on the previous formulation and according to the three available configuration formats:

- **3-Segments**: For each non-shortest path link (u, v) in a path from s to t, identified in the described procedure, the algorithm produces a hop-by-hop path such that paths from s to u and from v to t are shortest-paths. Non-shortest path links (u, v) become adjacency segments, while remaining path portions are converted to node segments ($[\mathtt{u}; (\mathtt{u}, \mathtt{v}); \mathtt{t}]$).

- **2-Segments**: When u or v coincide with source or destination nodes, s and t respectively, the path becomes a 2-segment SR path ($[(\mathtt{s},\mathtt{u});\mathtt{t}]$ or $[\mathtt{u};(\mathtt{u},\mathtt{t})]$).

- **1-Segment**: A shortest-path between s and t is converted into a 1-segment SR path, $[\mathtt{t}]$.

Load Balancing Between Parallel Paths

The fraction of traffic to be assigned to each label path between s and t can easily be computed using Equation 5.1, for the label path containing an adjacency segment (u,v), and 5.2, for a label path that only contains the destination node t SID. In the equations, $\mathcal{P}^{s,u}$ represents the set of all shortest paths from s to u, and $\mathcal{A}^{s,t}$ is the set of all adjacency segments, to the nodes on the shortest paths from s to t, which are not on shortest paths.

$$F\left(s,t,u,v\right)=\begin{cases}P\left(h^{t}_{u,v}\right), & u=s\\ P\left(h^{t}_{u,v}\right)\times\sum_{p\in\mathcal{P}^{s,u}}\left(\prod_{(i,j)\in p}P\left(h^{t}_{i,j}\right)\right), & u\neq s\end{cases} \tag{5.1}$$

$$F\left(s,t\right)=1-\sum_{(u,v)\in\mathcal{A}^{s,t}}F\left(s,t,u,v\right) \tag{5.2}$$

By applying both equations to the provided example in Figure 5.3, the traffic splitting ratios are 4/15 for the label path with the adjacency SID, and the remaining traffic, 11/15, to the shortest-path, that is, the path with only the node SID of t, Figure 5.3c.

It is also important to highlight that there are some essential differences between the devised SR oriented TE proposal and the original DEFT/PEFT routing protocols. DEFT and PEFT do not ensure that flows traverse at the most one non-shortest path link, whereas ultimately, a flow could travel solely using non-shortest path links. In contrast, this proposal guarantees that flows traverse at the most one non-shortest path link. The solutions provided by this proposal are expected to have a quality very close to those offered by DEFT or PEFT, in a context of well distributed traffic necessities. Furthermore, and as mentioned earlier, SR introduces many advantages, such as less network state information to be maintained in intermediate routers, and the additional ability to perform per-flow TE without any change to the installed configuration.

Table 5.1: *Network topologies used in the experiments.*

Topology	Type	Nodes	Links
$Rand30_2$	Synthetic	30	110
$Rand30_4$	Synthetic	30	220
$Rand50_2$	Synthetic	50	194
$Rand50_4$	Synthetic	50	380

5.3 SR Congestion Optimization

5.3.1 Experiments Setup

The validation of the proposed SR TE model for network congestion optimization was performed similarly to the validation of Link State and SDN routing optimization models: given an SR enabled network topology and traffic necessities modeled as traffic matrices, the aim is to find an SR configuration (IGP Link weights, SR paths, and parallel paths load balancing ratios) that minimize the network congestion, evaluated with the congestion function Φ^*.

As a reminder, it is important to note that when the normalized sum of link costs equals 1, all loads are below 1/3 of the link capacity, and when all arcs are precisely full the value of this congestion metric is 10 2/3.

We considered a set of distinct randomly generated network topologies, summarized in Table 5.1, varying in size and average node in/out degree. These topologies are the same used in the previous chapters an results can be directly compared. For each topology, we produced a set of random traffic demand matrices D with distinct levels of expected average link utilization, using the model described in Section 2.4.

The link weights values are taken from the range $[1; 20]$ of integer values. A minimum threshold of 5% is defined for parallel traffic load balancing, that is, any parallel path, between a source s and a destination t, forwards at least 5% of the aggregated traffic from s to t. For practical reasons, it might not be justifiable to forward a lower percentage of edge-to-edge traffic along an SR path considering that the impact on the overall network congestion is negligible. All optimizations were performed by a Single Objective Evolutionary Algorithm (SOEA) optimization engine configured as in previous experiments.

5.3.2 Optimization Results

Representative results taken from 30 experiments of each scenario are presented in Table 5.2. For comparison purposes, we include the metric congestion values obtained with *InvCap* link weights configuration, where each link weight is set to a value inversely proportional to its capacity. Also included are the congestion values for OSPF/ECMP with optimized weights (*OSPF Opt.*) as well as for DEFT and PEFT routing protocols when configured with optimized settings. The SALP-SR optimization results are identified as SR-D and SR-P according to the used splitting function, respectively DEFT and PEFT.

Table 5.2: *SR optimized congestion values.*

Topology	*D* Level	InvCap	OSPF Opt.	DEFT	SR-D	PEFT	SR-P
$Rand30_2$	0.3	15.60	1.40	1.36	1.36	1.36	1.36
	0.5	494.92	3.77	2.94	2.94	2.74	2.74
$Rand30_4$	0.3	323.76	1.69	1.59	1.59	1.59	1.59
	0.4	717.95	5.17	2.94	2.94	2.94	2.94
$Rand50_2$	0.3	437.70	1.68	1.61	1.61	1.61	1.61
	0.4	474.44	2.13	1.89	1.89	1.88	1.88
$Rand50_4$	0.3	156.04	3.91	2.49	2.49	2.38	2.38
	0.4	486.09	34.70	22.27	22.27	21.12	21.12

The results in Table 5.2 show that, although the traffic distribution is not the same, i.e., flows may not follow the same path, the link utilizations are equal. Consequently DEFT and SR-D solutions, as well as PEFT and SR-P, have equivalent quality. The small differences between congestion values are due to the imposition of the minimum splitting threshold and to multiplicative propagation errors. It can also be observed that in all cases, SR-D and SR-P provide solutions with better congestion than OSPF/ECMP with optimized configurations. SR-D and SR-P can make use of the available resources. We also observe that SR-D and SR-P offer configuration solutions with approximate levels of congestion and the additional computational time required by PEFT might not be justified.

Table 5.3 provides comparative statistics of all routing techniques for a same link weights configuration on topology $Rand30_2$. Congestion values have equivalent quality. Even considering that traffic from the same flow does not follow the same path in DEFT and SR-D, the link utilizations are approximately the same. While with DEFT packets from a flow may eventually be forwarded along non-shortest path links from source to destination, on SR-D the deviation from the shortest path is of one hop at the most.

Table 5.3: *Comparison statistics for the same weights configuration.*

Measure	DEFT	PEFT	SR-D	SR-P
Congestion	1.3803	1.3815	1.3785	1.3795
Min. link utilization	3.06	3.06	3.06	3.06
Max. link utilization	72.72	73.04	72.04	71.98
Avg link utilization	35.874	35.870	36.679	35.802
Link utilization Std	16.909	16.839	16.768	16.859

5.4 Traffic Variation

Edge-to-edge traffic necessities vary over time, and although an optimized configuration provides some level of resilience to those variations, the installed configurations may be no longer suited for networks to accommodate the new traffic requirements. In some cases, by implementing per-flow TE, it is possible for a PCE or a SDN controller to find a shortest routing label path with sufficient available capacity to sustain and steer traffic to its destination or seek for the path with maximum available capacity using, for example, the Ford—Fulkerson method [38]. Although this approach can be an adequate solution for elephant flows, it is not so appealing for short duration flows, as it requires some time to identify a flow as elephant or mice. On the other hand, implementing per-flow routing for a high number of flows is not an easy task. New flows continuously appear and die which would need to be monitored and managed by an SDN controller, along with constant auditing of the network links availability. Another possibility is to install a new routing configuration which would imply a new link weight configuration with consequent impact on the network IGP and that should be avoided. A better approach consists in adjusting the traffic splitting between parallel paths in the PCE. Although this solution does not support all possible variations, it can be a good temporary fix.

SALP-SR DEFT and PEFT load balancing functions use a parameter on the spitting ratios computation, node-p values, that can be used to improve the division of traffic between source and destination pairs label paths. In Section 4.2.4, we showed how different node-p values in Equation 4.2 impact the splitting of traffic. By assigning distinct node-p values to each node, it is possible to alter the amount of traffic to be forwarded along adjacency segments, which also impact the volume of traffic traveling along shortest paths. For an installed configuration of link weights and SR label paths, we propose to adjust traffic splitting between label paths by only optimizing the node-p values for a given new traffic matrix meanwhile computed by an SDN controller module

Table 5.4: *SALP-SR node-p values optimization for traffic variations.*

Measure	$Rand30_2$							$Rand30_4$						
	D_0	D_1			D_2			D_0	D_1			D_2		
	Opt.	Before	After	Opt.	Before	After	Opt.	Opt.	Before	After	Opt	Before	After	Opt.
Cong.	1.36	23.26	1.49	1.37	15.70	3.52	1.82	1.89	14.70	2.31	1.88	22.61	3.28	2.20
ALU (%)	33	37	34	33	43	44	42	35	34	35	33	40	41	40
MLU (%)	71	153	98	88	122	110	90	90	118	102	89	125	105	101

Measure	$Rand50_2$							$Rand50_4$						
	D_0	D_1			D_2			D_0	D_1			D_2		
	Opt.	Before	After	Opt.	Before	After	Opt.	Opt.	Before	After	Opt	Before	After	Opt.
Cong.	1.69	2.42	1.83	1.78	16.39	7.57	2.00	1.90	4.35	2.32	1.78	9.48	3.25	1.80
ALU (%)	36	37	38	38	38	38	38	25	28	29	28	30	32	30
MLU (%)	94	107	98	97	144	119	100	90	123	99	91	126	109	90

or PCE. Results from this approach are presented in Table 5.4 and only consider the DEFT splitting function. The optimization was implemented using EAs, where node-p values are taken within the range $[0.01, 10.0]$ of real values. As in the previous case, a minimum of 5% of each edge-to-edge aggregated traffic is always forwarded on all SR paths.

Results in Table 5.4 include three metrics: congestion, the Average Link Utilization (ALU) and the Maximum Link Utilization (MLU). Each topology is initially optimized for a traffic matrix D_0 (Opt.) and traffic variations for two distinct moments are represented as traffic matrices D_1 and D_2 with a linear correlation of approximately 0.5 with D_0 and that represent an increase in traffic requirements, as can be observed in the ALU. For each of those moments, we present the measures obtained with the installed configuration (*Before*) and those obtained after the optimization of the node-p values (*After*). For comparison, we also include the measures for D_1 and D_2 with optimized configuration.

By optimizing the node-p values assigned to each node, SALP-SR can address some traffic demands variations that undermine the network performance. As observed in Table 5.4, there is a definite improvement on the congestion values after the optimization of the node-p values. In fact, in a significant number of scenarios the network operates now in an acceptable working region, while before the optimization of the node-p values the network was on a congested situation (values above 10 2/3) or very close to it.

Although this strategy always introduces improvements to the traffic splitting and congestion, in some cases, it does not provide a good enough response, and some links continue to be over-utilized even after the optimization process. Other TE solutions, such as those identified earlier, could, therefore, be considered to accommodate the traffic variation.

One of the advantages of this approach is the relatively short amount of time required to optimize the node-p values. For example, for the topologies with 30 nodes, this time is less than one minute while for 50 nodes topologies around two and a half minutes. Therefore this strategy can be evaluated simultaneously with the traffic necessities updates. An assessment of if it is sufficient to maintain the network performing at an acceptable level can be made, and if it is the case, no additional measure need to be taken to respond to short-term traffic variations.

5.5 Single Link Failure

SR is built over already existing Interior Gateway Protocols (IGP), such as OSPF and IS-IS, and takes advantage of multiple of their features. SR supports ECMP by design, but also the automatic rerouting of connections after a link failure. Upon a link failure, the IGP protocol recomputes all shortest-paths, and segments are automatically repaired without any additional intervention.

The time required to detect a link failure, propagate the fault and recompute the shortest-paths can be, however, excessively long and, therefore, recovery paths should preferably be pre-computed and installed in the data plane. Fast reroute (FRR) with loop-free alternates (LFA) [11] follows this strategy and provides sub-50msec loss of connectivity to IGPs. However, LFA does not offer a complete network coverage and is topology dependent. With SR, those limitations ceased to exist. Topology-Independent LFA (TI-LFA) [14] provides local protection for IGP SIDs in any topology. Backup paths can be pre-computed on a per IGP SID basis along the post-convergence path from the Points of Local Repair (PLR) to all possible destinations. In the vast majority of cases, a single segment is enough to encode the post-convergence path in a loop-free manner. However, even though TI-LFA solves the loss of connectivity problem for single link or node failures, it is insufficient to ensure that, after the IGP has converged, there is no performance degradation in the SR domain.

This section addresses the SR link failure post-convergence congestion problem. We start by analyzing post-convergence congestion levels of networks that use TI-LFA and compare them with distinct possible approaches which resort to Evolutionary Computation (EC) algorithms. Network's configuration on a normal state, that is, before a single link failure, is optimized for congestion and uses the Single Adjacency Label Path Segment Routing optimization model.

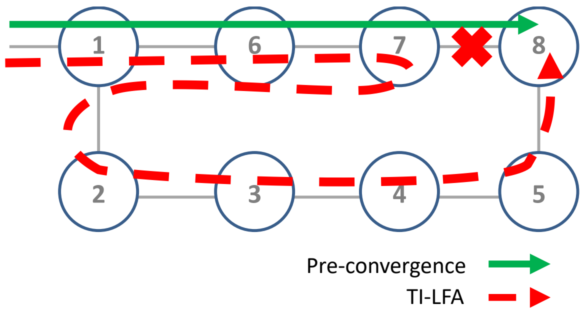


Figure 5.4: *Micro-loop example.*

5.5.1 Recovery Path Computation Alternatives

The standard SR approach to a link failure is TI-LFA. The main idea of TI-LFA is to provide loop-free recovery paths, between the PLR and provider's edge destinations, which remain unchanged before and after the IGP convergence. However, a simple approach which provides recovery paths from the Point of Local Repair (PLR) on, may not be sufficient to warrant adequate overall network congestion levels. It is also essential to take into consideration IGP shortest-path recomputation and how it affects the network congestion level. IGP shortest-paths recomputation has an impact on node segments and, consequently, on traffic distribution over the available resources after a link failure. To prevent an eventual congestion problem, network administrators need to perform over-provisioning. Furthermore, micro-loops, where traffic at the PLR is forwarded back to an already traversed node, are only solved after the IGP has converged (Figure 5.4). With TI-LFA, after a link failure, edge routers also cease to be responsible for the explicit definition of affected traffic forwarding paths. As a consequence, and for those reasons, additional measures need to be implemented.

Any additional or alternative approach needs to consider: 1) the definition of where SR segment ID stack should preferably be updated, e.g. at the PLR or at network ingress nodes; 2) which portion of the recovery path should be updated, e.g. to the destination or only to the next segment not affected by the failure. These questions define three end-to-end possibilities: PLR to the destination (TI-LFA), PLR to the next segment or edge-to-edge SR recovery paths (Figure 5.5). It might also be conceivable to implement corrections on traffic load balancing to improve traffic distribution, or even alter the SID stack of SR paths that were not affected by the failure. In this context, we devised distinct approaches to analyze single link failure impacts of on SR network's congestion and simultaneously evaluate their responses. None of the proposals consider SR path with service chaining, and only paths that steer traffic from source to destination are considered. All approaches aim to minimize the network post-convergence congestion after a single failure. Solutions for approaches that require longer computation time are preemptively computed, and stored in a PCE database as shown in Figure 5.2.

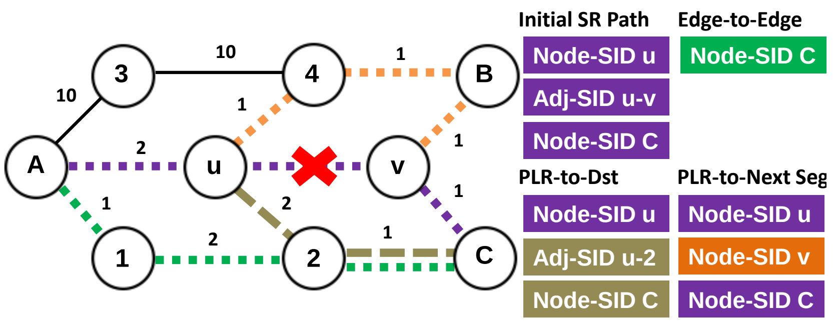


Figure 5.5: *SR recovery paths: Edge-to-Edge, PLR to destination, PLR to next segment.*

Edge-to-Edge Shortest Path Approach

The most straightforward alternative response to a single link failure is to reroute traffic using IGP shortest paths. In practice, all SR paths that included the failing link (u,v) are reconfigured with a single Node Segment [node(t)], where t is the provider's edge destination. This approach has the disadvantage of not being responsive enough. As edge routers need to become aware of the fault, it can only be implemented after the fault is announced to all routers and the IGP has converged. On the positive side, it does not require any centralized control, and only edge nodes need to recompute SR paths.

SALP-SR Approach

A SALP-SR configuration encompasses IGP link weights configuration, edge-to-edge SR path definitions and load balancing splitting ratios between parallel paths. All configurations are derived from a set of integers, the IGP link weights, and from a set of real values assigned to each node (node-p values). When the network topology changes due to a link failure, the change is announced to all network nodes and to the controller or PCE. To reflect the changes, the IGP recomputes the shortest-paths while the controller redefines the SR paths and, most importantly, the traffic load balancing between parallel paths. This is equivalent to applying the path computation process described in Section 5.2.3 to the altered network topology while preserving the already installed IGP weights.

A disadvantage of this procedure is that the fault needs to propagate through the network before it can be applied. Futhermore, it might imply to change SR paths already assigned to some flows in order to comply with the new configuration. On the positive side, this procedure does not require any preemptive computation. SR paths and load balancing ratios are computed in a few milliseconds and the new configuration is installed at the edge nodes by the controller.

The initial optimized IGP link weights, from witch SR paths and parallel paths load balancing are computed, depend on the installed node-p values. Consequently, after a link failure the node-p values and IGP weights cease to be optimally tied which impacts the quality of SR parallel paths load balancing ratios. Therefore, additionally to the SALP-SR paths recomputation, new and improved traffic splitting ratios between parallel SR paths may be installed by optimizing the node-p values thus improving network operational conditions.

The two possible approaches, without and with node-p values optimization, lead to the following two cases:

- **SALP-SR rerouting with default node-p values:** New SR paths and load balancing ratios are recomputed reflecting the topology changes. The IGP link weights and node-p values are kept unchanged.
- **SALP-SR rerouting with optimized node-p values:** New SR paths and load balancing ratios are recomputed reflecting the topology changes. The IGP link weights are kept unchanged and the node-p values are optimized. The node-p values can be preemptively optimized for each possible link failure and stored in a database to be later applied.

Multi-Objective SALP-SR Approach

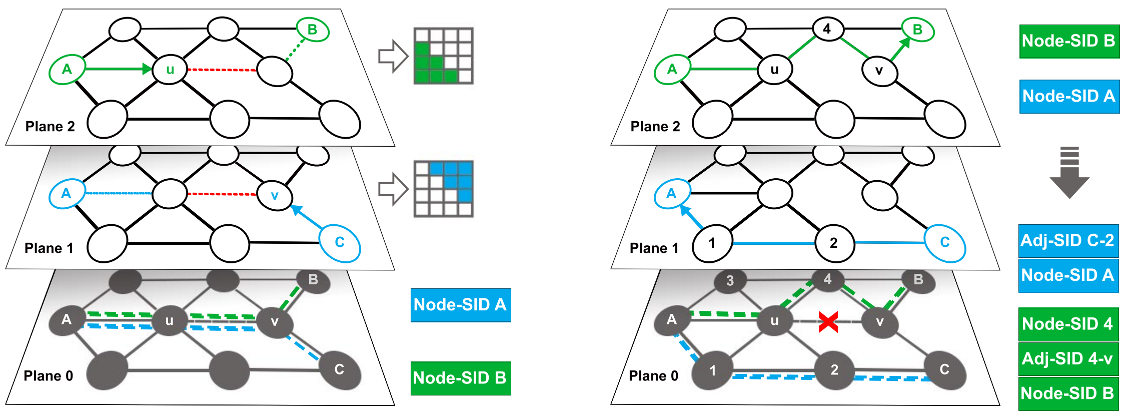
Although similar to the previous, this approach considers two objectives instead of a single objective for the initial SALP-SR optimized configuration. In the previous chapters, both on networks running SDN and Link State, multi-objective optimization for link-state routing protocols demonstrated promising results for the single link failure congestion problem. This approach follows the same strategy and aims to improve networks performance by minimizing both congestion values simultaneously, before and after the failure of any single link.

In this approach, the initial network optimization is performed considering simultaneously: First Objective - minimize the congestion of the network on a fully functional state, Φ^*; and Second Objective - minimize the maximum congestion after a single link failure, that is, each single link is set to fail and the worst measured congestion is minimized. The formulation of this second objective is $Min\left(Max\left(\Phi^*_{(n-1,a)}\right)\right)$, where $(n-1,a)$ denotes the failure of each individual link a.

As defined for the SALP-SR approach to a single link failure, the Multi-Objective SALP-SR approach contemplates two cases. In both cases, after a link failure and considering the already existing IGP link weights configuration, the SALP-SR paths are recomputed. The difference between the two cases resides in the parallel paths traffic load balancing computation, specifically in the node-p values configuration. While in the first case the node-p values remain unchanged and configured with the default value, in the second case the node-p values are optimized to reflect the topology change. To shorten the amount of time required to obtain new traffic load balancing ratios, node-p values can preemptively be computed for each possible single link failure. It is expected that, by adding the second optimization objective, the congestion levels of the network, after a link failure, improve when compared to the previous scenario.

Multiplane Approach

One of the attributes of SALP-SR is that traffic always moves towards the destination when considering distances as the sum of shortest-path link weights. Although this characteristic is a positive SALP-SR property, it nonetheless narrows the number of possible recovery path solutions. This approach forsakes this restriction, i.e., recovery paths may include segments which locally drive traffic away from the destination. This goal is achieved using additional network planes during the optimization process, where each additional plane is used to obtain SR recovery paths for the traffic affected by the failure. A conceptual representation of this approach is presented in Figure 5.6.



(a) *Traffic necessities identification* **(b)** *Recovery paths computation*

Figure 5.6: *Multiplane recovery path optimization.*

The computation of recovery paths for the failure of each link (u, v) is divided into two main steps:

1. **Identification of traffic that needs to be rerouted**. The optimization procedure identities traffic which, before failure, travels over (u, v). From this analysis, the algorithm produces two traffic demand matrices, $D1$ and $D2$, one for each of the failing link entry ports, that represent traffic necessities which need to be rerouted after the link failure, Figure 5.6a.

2. **Recovery paths computation**. The optimization of recovery paths is performed using a single objective EA whose objective is to minimize the network congestion measure Φ^*. While the link weights configuration of plane 0 remain unchanged, as well as the unaffected SR paths, each of the additional planes, planes 1 and 2, have a set of independent link weights, as seen in Figure 5.6b. During the optimization process, the link weights of plane 1 and 2 are concatenated into a single solution vector which evolves, enabling to obtain hop-by-hop unique shortest paths which are used to steer the traffic affected by the link failure. At each iteration, the hop-by-hop paths are translated into SR recovery paths that reflect the existing IGP configuration (plane 0), Figure 5.7. The SR recovery paths are then added to the unaffected SR paths in plane 0, configuring a new solution which is evaluated.

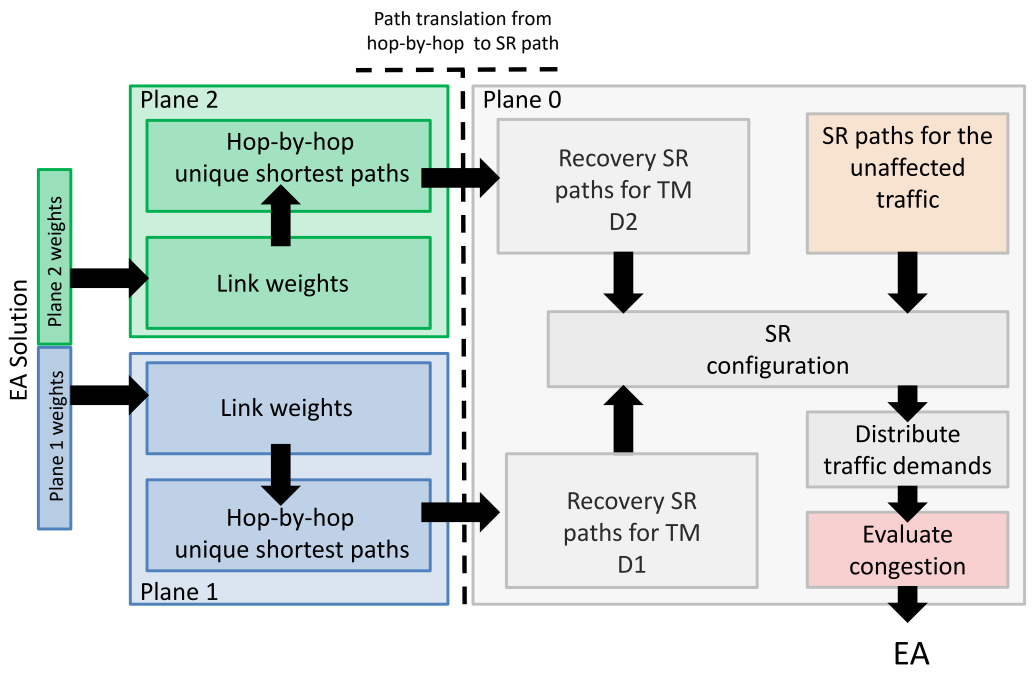

Path translation from hop-by-hop to SR path
Plane 2
Hop-by-hop unique shortest paths
Link weights
EA Solution
Plane 2 weights
Plane 1 weights
Link weights
Hop-by-hop unique shortest paths
Plane 1
Plane 0
Recovery SR paths for TM D2
SR paths for the unaffected traffic
SR configuration
Distribute traffic demands
Recovery SR paths for TM D1
Evaluate congestion
EA

Figure 5.7: *Evaluation of solutions for single link failure multiplane optimization*

The optimization process is very time consuming, therefore it must be performed preemptively for each possible single link failure. The SR recovery paths configurations resulting from the optimization process are stored into a database to be deployed if and when necessary. Planes 1 and 2 only compute unique recovery paths between each source/destination pair to ensure a unique mapping of paths' translation and affected SR paths into their respective recovery paths. If more than a SR path for a same source/destination pair is affected by the failure, the affected traffic is aggregated into the same recovery path and the traffic load balancing between parallel paths are adjusted accordingly. The remaining traffic load balancing between parallel paths are kept unaltered.

5.5.2 Experiments and Results

Experiments Setup

A set of experiments was devised to compare each approach with TI-LFA. We considered the set of synthetic network topologies used in the previous experiments which is summarized in Table 5.1, varying in size (30 and 50 nodes) and minimum in/out node degree (2 and 4). As in previous experiments, traffic demand matrices were randomly generated, using the model described in Section 2.4, and with an expected utilization of 30% of existing resources ($D0.3$).

Solutions take integers values from the range [1; 20] for link weights, while node-p values are real numbers in the range [0.01; 10]. The optimization objectives are the minimization of the normalized sum of links' congestion cost Φ^*. The initial multi-objective optimizations use NSGA-II as optimization engine and the remaining optimizations are performed using a Single Objective Evolutionary Algorithm (SOEA). The initial single and multi-objective configurations were obtained using a stopping criterion of 1000 iterations, node-p values optimization used a stopping criterion of 100 iterations, and the multiplane optimization where run with a stopping criterion of 200 iterations for each individual link failure.

Results

Results presented in Table 5.5 are average post-convergence congestion values from 10 runs of each experiment. They are divided into two main groups, before (normal state) and after a single link failure. In the last, organized by each approach previously explained, values are the mean of network congestion after the failure of each link, one at a time. Optimized congestion values for networks in their normal state, that is, fully functional, are operational benchmarks with which the remaining values contrast.

The minimum node in/out-degree of a network topology significantly influences the quality of recovery paths. Topologies with higher minimum node degree have more available edge-to-edge recovery paths after a single link failure. In this context, it is understandable that topologies with a higher minimum node in/out-degree present globally better results.

Table 5.5: *Average congestion before and after single link failure.*

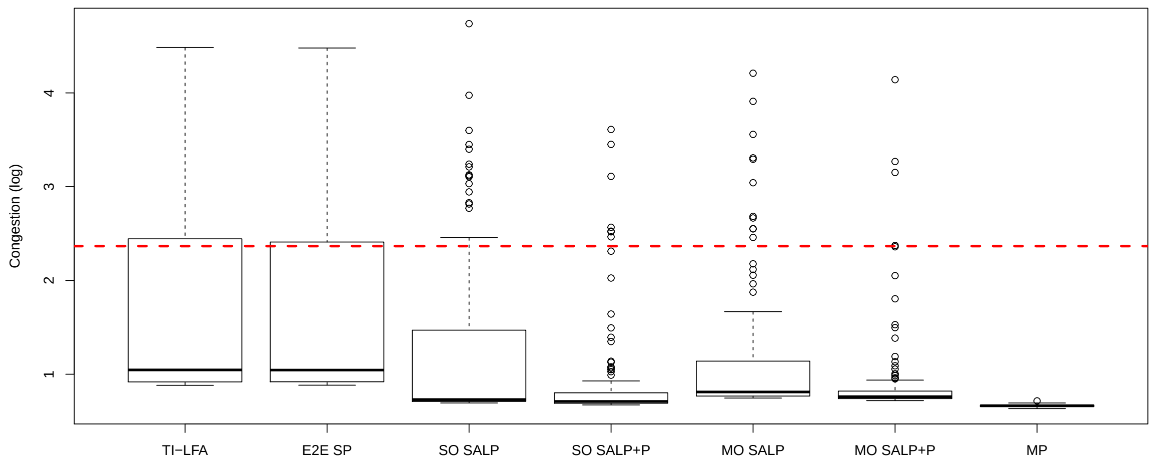
Topology			Normal State		Single Link Failure State						
							SO SALP		MO SALP		
Name	Nodes	Links	SO	MO	TI-LFA	E2E SP	SALP	SALP+P	SALP	SALP+P	MP
$Rand30_2$	30	110	1.40	1.42	10.45	10.40	8.49	6.09	8.19	6.26	5.06
$Rand30_4$	30	220	1.96	2.35	9.11	9.05	6.75	3.46	5.14	3.47	2.16
$Rand50_2$	50	194	1.79	2.07	9.53	9.41	8.17	6.10	7.01	5.98	4.86
$Rand50_4$	50	380	2.43	4.91	9.57	9.48	8.98	5.31	9.14	5.87	3.73

Edge-to-Edge Shortest Path Approach (E2E SP column in Table 5.5)

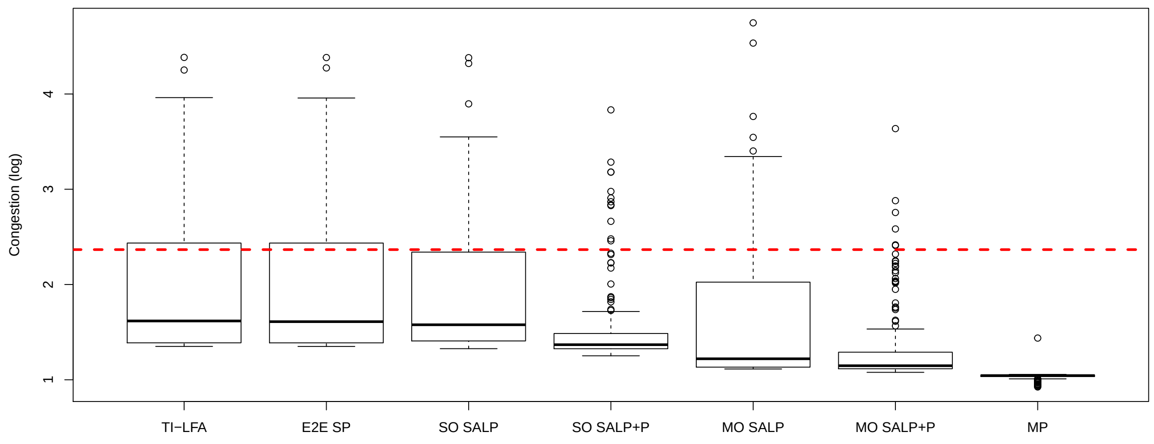
Results for the first group of link failure experiments reveal that there are no significant differences in the congestion values between recovery methods that solely rely on pos-convergence shortest paths, that is, from the PLR to destination (TI-LFA) and edge-to-edge. They both globally display values below (but near) the operational threshold of the network (10 2/3). In particular, TI-LFA simulations, although capable of shortening connectivity lost to under 50 msec, present the highest congestion values on all experiments. For the particular case of the $Rand30_4$ network topology, in both approaches, TI-LFA and Edge-to-Edge SP, the network congestion is above the network functional threshold, represented as a dotted red line in Figure 5.8, in more than 25% of link failures.

SALP-SR Approach (SO SALP columns in Table 5.5)

Approaches that use SALP-SR enhance the simple edge-to-edge recovery approach by taking advantage of non-shortest path links, and as expected, enabling network congestion to diminish. In the first approach, only SR paths are reconfigured by the SALP-SR algorithm, reflecting the newly computed IGP shortest-paths. It is important to reemphasize that none of the approaches alter IGP weights and, consequently, SR paths not affected by the link failure remain unchanged. If additionally load balancing ratios are adjusted, SALP+P column in Table 5.5, network congestion values drop to almost half, on average, of those observed with shortest path rerouting. In fact, for the $Rand30_4$ network topology with SALP+P, where a load balancing correction is performed to SALP-SR, only 3% of link failure results in a congested network state, Figure 5.8.



(a) Topology $Rand30_4$



(b) Topology $Rand50_4$

Figure 5.8: *Individual link failure congestion values distribution.*

Multi-Objective SALP-SR (MO SALP columns in Table 5.5)

The multi-objective initial SR configuration enables the network to handle single link failures better. Results for this approach show that after a SALP-SR paths recomputation networks perform better with multi-objective (MO SALP) than with single objective optimizations (SO SALP) as can be observed in Table 5.5 and Figure 5.8. However, these differences fade with the adjustment of load balancing ratios (SO SALP+P vs. MO SALP+P). A multi-objective optimization establishes a compromise between the optimization goals, a trade-off, by relaxing configuration fitness on both objectives. However, an increase in SR configurations flexibility, with a penalization on fully functional network congestion, is insufficient to improve on all results obtained with single objective optimization. Additional measures need to be installed to improve the already obtained results.

Multiplane Approach (MP column in Table 5.5)

Multiplane recovery path optimization adds to SALP-SR the ability to have more flexible SR paths. Although recovery SR paths may not always locally forward traffic towards the destination, they permit a reduction in post-failure congestion significantly. Results show that this approach facilitates post-failure congestion values significantly better than all other approaches, as can be observed in Figure 5.8, and close to those observed in fully functional networks. The higher the minimum network's node in/out degree, the lower are post-convergence congestion values. Moreover, this approach only requires edge nodes to re-configure SR paths affected by the link failure. SR recovery paths are preemptively computed and stored in a PCE/SDN controller or locally at each router. Therefore, replacement paths can be immediately installed after the fault propagation. If traffic necessities, which are used to compute recovery paths, change significantly and undermine the quality of the recovery SR paths, a PCE can quickly compute new load balancing ratios between parallel paths and improve the overall network congestion.

5.5.3 Constrained Segment List Depth Multiplane Approach

The multiplane optimization approach formerly presented has, nevertheless, a drawback: SR paths may require more than 3 SIDs to be configured. The percentage of SR paths that use more than 3 SIDs and the maximum number of SIDs for a single optimized solution for each network topology are presented in Table 5.6.

Table 5.6: *Comparison of Multiplane and Constrained Multiplane optimization (Illustrative examples)*

Topology	$Rand30_2$	$Rand30_4$	$Rand50_2$	$Rand50_4$
Percentage of recovery paths with more than 3 SIDs	4.7%	10.8%	14.6%	24.8%
Maximum Path Length	5	6	7	7
Congestion when replacing long SR paths with SR SP	6.21	2.19	6.28	3.84
Constrained multiplane optimization	4.78	1.70	4.47	2.77

It comes with no surprise that as topologies grow in the number of nodes and links, the percentage of multiplane recovery SR paths which require more than 3 SIDs to be configured escalates as well as the maximum SID header stack length. The excessive label stacking length may cause scalability issues as the maximum SID header stack length varies currently from 3 to 5 depending on the equipment manufacturer [49].

To overcome such restriction and impose a maximum length of 3 SIDs, SR paths that exceed that restriction can be, for example, substituted with shortest path SR configurations. Congestion values obtained by performing this alteration are also presented in Table 5.6. In some cases the congestion value is not significantly altered, such as for the topologies with higher average node in/out degree, $Rand30_4$ and $Rand50_4$. This observation emphasizes the higher diversity of available paths for TE in networks with such property. In other cases, however, the congestion increases to values equivalent to those obtained with the MO SAPL-SR with load balancing correction (Table 5.5 MO SALP+P).

The multiplane approach can be extended so that the segment list depth constraint is included in the evaluation of all solutions during the optimization process, as depicted in Figure 5.9, instead of only being applied to the final solution. After the hop-by-hop paths have been translated to SR paths that reflect the installed IGP waits, SR paths configurations with segment stack length greater than 3 are converted to SR paths with a single SID, the destination nodal SID, forcing traffic to be forwarded along shortest paths from source to destination.

The results for the constrained segment list depth multiplane optimization are also included in Table 5.6. By comparing the obtained congestion values with those provided by the unconstrained version of the multiplane approach, Table 5.5, we can observe that they are equivalent in quality. Such result highlights the ability of EAs to adapt to new additional constraints that would be more difficult to implement with traditional optimization mechanisms.

5.5.4 Architectural Model

Single link failures have two main impacts on the network's operations, they undermine connectivity and impact the overall network congestion negatively. Segment Routing, however, enables the deployment of a more complete and effective response to the problem of preserving network's post-failure congestion levels. We derive two main conclusions from the possible approaches to this problem explored in this work. Firstly, although TI-LFA is an excellent solution to reestablish networks connectivity

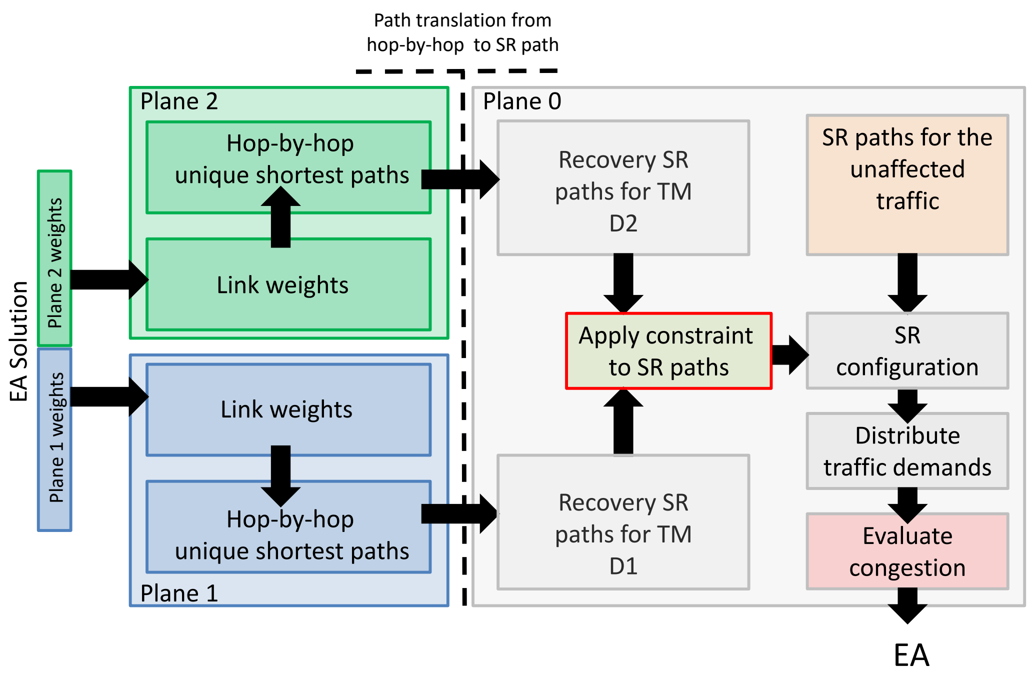

Path translation from hop-by-hop to SR path
Plane 2
Hop-by-hop unique shortest paths
Link weights
Plane 2 weights
EA Solution
Plane 1 weights
Link weights
Hop-by-hop unique shortest paths
Plane 1
Plane 0
Recovery SR paths for TM D2
SR paths for the unaffected traffic
Apply constraint to SR paths
SR configuration
Distribute traffic demands
Recovery SR paths for TM D1
Evaluate congestion
EA

Figure 5.9: *Multiplane approach: Evaluation of solutions with constrained segment list depth*

in a brief amount of time, it is insufficient to provide functional levels of congestion after a link failure. Other approaches such as SALP-SR with node-p optimization or the constrained multiplane approach deliver better post-convergence congestion levels but require more time to be deployed. Secondly, we defend that changes on SR paths should, when possible, be implemented at edge nodes instead of at points of local repair. A combination of both approaches presents itself as a good compromise to achieve both goals, shortening the reaction time and decreasing network congestion.

Some of the approaches require that recovery paths to be pre-computed and stored to be deployed when a link failure occurs. While the SALP approaches without load balancing correction do not imply any additional optimization and can be deployed in a few seconds, node-p values correction and the multiplane approaches require additional optimization. In this context we extend the SALP-SR general architecture presented in Figure 5.2 with a SR recovery path table as shown in Figure 5.10.

After a link failure, connectivity can be reestablished using TI-LFA, and as soon as the IGP converges, optimized SR paths can be installed at edge nodes, achieving this way both desired goals. Recovery paths are preemptively computed, considering foreseeable traffic necessities, and can be stored locally at edge router or centrally by a PCE. By providing a better distribution of traffic among available resources, resilience to traffic variations also increases. Nonetheless, changes in traffic necessities may also be

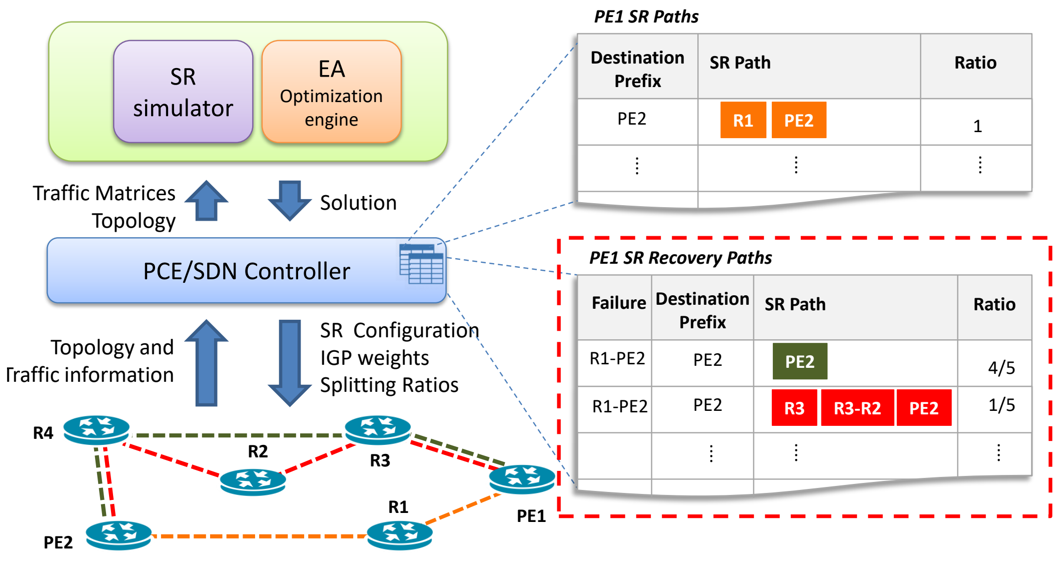


Figure 5.10: *Conceptual architecture of SALP-SR with recovery paths optimization.*

addressed by correcting traffic load balancing between parallel paths, using the node-p values optimization feature. This last can be implemented in less than a 100 seconds, with satisfactory results, in topologies with less than 50 nodes.

5.6 Extended Node-p Values Optimization

The optimization of node-p values allows to improve the traffic load balancing between SR paths with a same origin/destination pair. This strategy, used to address congestion states due to changes on network's conditions, is achieved by a transformation of the Γ function, Equation 4.2, which graphically corresponds to a expansion or contraction of its plot, Figure 5.11. If node-p values are greater than 1 (the default node-p value) the portion of traffic forwarded across non-shortest SR path increase, and if node-p values are in range $]0,1[$ the portion of traffic decreases. In both cases, the amount of traffic routed along IGP shortest paths is always greater than the amount of traffic driven along SR non-shortest paths. Such a property is generally considered as an advantage. Packets traveling along shortest paths have a single SID inserted in their header while it takes more than one SID for non-shortest path SR routing.

However, in a TE perspective and to reduce congestion after a network event, it may be beneficial for some origin/destination pairs, to forward more traffic along non-shortest paths than along shortest paths. In the context of the proposed SR optimization model, this can be achieved by redefining the Γ function to consider non-positive node-p

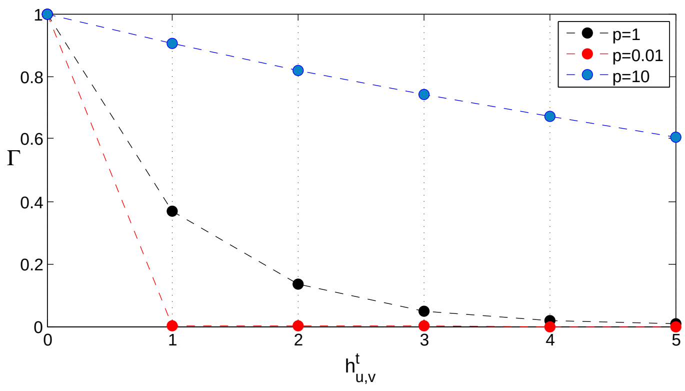


Figure 5.11: Γ *function plot*

values, as in Equations 5.3 and 5.4. For $h^t_{u,v} = d^t_v + w_{u,v} - d^t_u$ and a positive node-p values, the behavior of the Γ function remains unchanged and performs as described in Section 4.2.4. When the node-p value at a node u is set to 0, a next-hop v in a non-shortest path such that $d^t_v < d^t_u$ receives the same amount of traffic as next-hops in a shortest path to t. While the Γ function returns values in range $]0,1[$ for positive node-p values, for negative node-p values the Γ function should 1) return values greater than 1, and 2) the function should be decreasing in order to $h^t_{u,v}$ and p. From the set of all possible functions we choose the one defined in Equation 5.4 because of its simplicity and because it provides the desired set of load balancing fractions for node-p value in the range $[-10,-0.01]$.

$$\Gamma\left(h^t_{u,v}, p\right) = \begin{cases} f\left(h^t_{u,v}, p\right), & \text{if } d^t_v < d^t_u \\ 0, & \text{otherwise} \end{cases} \tag{5.3}$$

$$f\left(h^t_{u,v}, p\right) = \begin{cases} e^{-\frac{h^t_{u,v}}{p}}, & \text{if } p > 0 \\ 1, & \text{if } h^t_{u,v} = 0 \text{ or } p = 0 \\ 1 - \frac{p}{\left(h^t_{u,v}\right)^2}, & \text{if } p < 0 \end{cases} \tag{5.4}$$

In Figure 4.3 we showed an example of how positive node-p values influence the hop-by-hop load balancing. Considering the same example, Figure 5.12 illustrates how such approach is able to take farther the utilization of non-shortest path links by forwarding more traffic than shortest path links.

To evaluate the extended node-p values optimization, we considered the single link failures congestion problem, more precisely the Single Objective SALP-SR approach

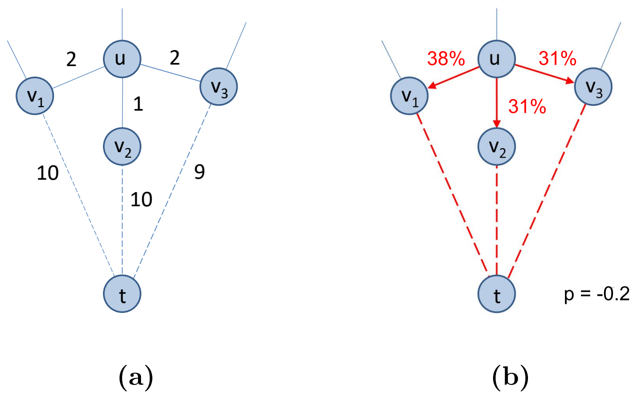


Figure 5.12: *Node-p value effect on traffic load balancing.*

Table 5.7: *Extended node-p optimization*

Topology	$Rand30_2$	$Rand30_4$	$Rand50_2$	$Rand50_4$
Node-p Optimization	5.95	2.23	5.90	3.77
Extended node-p Optimization	5.94	2.10	5.69	3.63

with node-p values optimization, and contrasted new results with the previously obtained which only use positive node-p values. In the optimization procedure, for each topology node u, the p variable take values in the range $[-10, 10]$. The results, which are the average of 10 runs for each topology considering $D0.3$ traffic demands, are presented in Table 5.7.

Allowing to forward more traffic along non-shortest paths than shortest paths does not always bring an improvement on the network congestion level. Improvements are only observed in a reduced number of link failures. In most cases the node-p optimization with positive values and its extended implementation provide solutions of an equivalent quality. However, those few cases offer node-p configurations that may deserve consideration. As an example, for the failure of a particular link in the $Rand50_4$ topology the post-convergence congestion level was 11.85, and the positive node-p values optimization only was able to reduce congestion to the value of 11.38. In contrast, by forwarding more traffic along non-shortest SR paths, the extended version was able to reduce this value to 6.62. Those gains, although significant in sporadic cases, are diluted in the averaged congestion value.

5.7 Conclusions

Segment Routing combines the simplicity of Link-State routing protocols with the flexibility of Multiprotocol Label Switching. By decomposing forwarding paths into segments, identified by labels or SIDs, SR improves Traffic Engineering and enables new solutions for the optimization of network resources utilization. In this chapter, we proposed an optimization technique, the Single Adjacency Label Path (SALP-SR) to improve resources utilization on SR networks. SALP-SR possesses several advantages. First, it provides near optimal resources utilization while using at most three segments to configure SR forwarding paths. Second, after each hop traffic is always closer to its destination. Third, SALP-SR enables to improve load balancing between parallel path by optimizing a set of variables, the Node-p variables. This feature allows, for example, address congestion issues that may result from changes in traffic demands or link failure without any alteration on SR paths or the IGP link weights.

We also compared the IETF proposal to address link failures, TI-LFA, with other possible approaches. We identified some limitations of TI-LFA, namely its inability to properly address congestion problems that may arise due to a failure. In such a context, we proposed alternative approaches that can couple with TI-LFA to provide networks solutions which are both responsive and congestion aware.

Chapter 6

The NetOpt Framework

This chapter describes the tool that was developed and used to obtain the results presented along in this thesis. After a brief introduction, this chapter provides an overview of the development history of the Network Optimization (NetOpt) framework and succinctly contextualize the optimization problems that forged its evolution. A general architecture of the framework is then presented as well as a description of its main optimization and analysis tools.

6.1 Introduction

The NetOpt framework was developed as a tool to assist network administrators in configuring traffic routing on career-grade networks. Initially developed in 2006 [101] at the Centro de Ciências e Tecnologias da Computação (CCTC) of University of Minho, the NetOpt framework aimed to provide networks with Quality of Servide (QoS) aware routing configurations. Taking as starting point the work devised by Fortz and Thorup on the minimization of network congestion for OSPF/IS-IS routing protocols [39, 40], the NetOpt framework added end-to-end delay as a new objective to the optimization problem. NetOpt uses Evolutionary Computation algorithms as optimization mechanisms that allow the handling of multi-objective problems without the need for a linearization of objective in a single cost function. Since, new functionalities have been added to the framework which contemplate new optimization objectives and scenarios for networks running OSPF/IS-IS routing protocols but also SDN and SR networks. Implemented using the Java programming language, the NetOpt framework is an open-source project publicly available at http://darwin.di.uminho.pt/netopt. By making available the NetOpt framework we want to provide a better understanding of the concepts behind all problems and proposed solutions. Not less important is that it gives the possibility to replicate all the experiments and, hopefully, allow to put the pointed solutions to practice in the real world.

6.2 The Evolution of NetOpt

The NetOpt framework initially appeared in [101] where the authors proposed to include delay restrictions to the computation of optimized OSPF weights. A propagation delay is assigned to each topology link and the delay in each path from s to t is the sum of the propagation delay of the links in the path (Del_{st}). Although the queuing delay is not considered in the end-to-end delay, the authors refer that an approximation could easily be included resorting to queuing theory. Given a matrix DR of end-to-end delay restrictions between all source/destination pair (s,t), the delay compliance ratio for (s,t) is defined as $d_{st} = Del_{st}/DR_{st}$. A delay cost function γ^*, which evaluates the compliance of the delay requirements, is defined as the normalized sum of all $p(d_{st})$ where p is a convex penalizing function similar to Φ_a, Figure 2.4. The optimization problem is consequently defined as: given end-to-end traffic demands necessities and delay restrictions, the aim is to find a configuration of weights w that optimizes both network congestion and end-to-end delay. This goal is achieved by minimizing the aggregated cost function $f(w) = \alpha\Phi^*(w) + (1-\alpha)\gamma^*(w)$, where $\alpha \in [0,1]$ is a trade-off parameter between the objectives. The results presented in [101] demonstrated that this optimization model was able to provide good OSPF weight settings able to satisfy the users demands.

The model was later extended in [102] to consider diferent classes of services. The authors contemplate C classes of service, with distinct requirements, whose congestion and delay need to be simultaneously optimized. Requirements for each class of service are modeled as individual end-to-end traffic demands, D^c, and delay restrictions matrices, DR^c. The aggregated cost function $f(w) = \sum_{c\in C}(\alpha_c\Phi^*_c(w) + \beta_c\gamma^*_c(w))$, where $\sum_{c\in C}(\alpha_c + \beta_c) = 1$, is used to optimize all congestions and delays. In [100], the authors added to the model multicast traffic routing. In [98] distinct heuristics (Evolutionary Algorithms (EAs), Differential Evolution, Local Search methods and common heuristics) were evaluated for the optimization model. By comparing results the authors concluded that EAs provide solutions with best quality.

Since 2012 the NetOpt framework underwent a new development phase. In [80, 85] we added to the initial model new optimization objectives that reflect changing conditions in network environments. The first addition addresses changes on traffic necessities. The aim is to optimize OSPF link weights for two distinct traffic matrices, that reflect different traffic necessities in time. The same problem was explored in the present work, namely in Section 3.4, Section 4.5 and Section 5.4 but in different contexts (such as in hybrid IP/SDN and SR networks) and proposing different solutions. The second addition proposed in [80, 85] addressed the single link failure problem in networks running the OSPF routing algorithm. The goal was to minimize congestion, and optionally also delays, for a specific link failure. Again only SOEAs were used as optimization engines and the provided solutions were only viable for a specific link choice. In the present work, taking the same idea, we addressed the possible failure of all links, one at the a time, in the context of hybrid IP/SDN and SR networks, and extended the basic model to include traffic load balancing optimization, making it more responsive to fault events.

The previoulsy mentioned optimization of distinct service classes using the objective $f(w) = \sum_{c \in C} (\alpha_c \Phi^*_c(w) + \beta_c \gamma^*_c(w))$ did not differentiated routing paths by service classes. All traffic with a same origin and destination will follow the same path regardless of its class of service. In [80] we proposed an optimization model for Multi-Topology OSPF routing [72] that can be used to provide distinct routing configurations for different classes of service while improving the networks resources utilization. Instead of considering as objective the minimization of the sum of the aggregated cost functions applied to the links utilization for each of the class of service traffic requirements, the cost function is applied only once considering the aggregated multi-topology traffic. In this optimization model, NetOpt resorts to a distinct mathematical model. Given a physical topology represented by the graph $G = (N; A)$, T logical topologies are defined as $G_\tau = (N_\tau; A_\tau)$ with $N_\tau \subseteq N$ and $A_\tau \subseteq A$ with $\tau = 1..T$. To model a possible traffic balancing approach, the demands D are distributed among several D_τ traffic matrices, which are mapped to the T logical topologies. In this multi-topology perspective each logical topology has associated a set of weights, W_τ, ruling the shortest paths computation over such topology and, consequently, the traffic distribution within the network. The EA represents a solution as an aggregate vector of all the w_τ weighting sets, i.e. a vector of integers in the form of $w = (w_{(1;1)}; ...; w_{(n;1)}; w_{(2;1)}; ...; w_{(n;T)})$ where n is the number of links. The total load in the physical topology is the the sum of all partial loads of the same link in each logical topology G_τ. The cost functions Φ^* and γ^* are subsequently applied to the links utilizations with the multi-topology aggregated traffic load.

The first version of the framework outfitted with a Graphical User Interface (GUI) was presented in [93], with limited functionality. In [80, 85], the GUI became fully operational with added functionalities. Additionally, the seeding of the initial EA populations was introduced enabling to shorten the number of iterations needed to obtain new configurations and simultaneously need to be correlated with a previous one, i.e., with reduced alterations on shortest paths. Additionally, a new real number measure was introduced that enables to compare differences in routing paths provided by different configurations.

In the context of the present work, we highlight the introduction of new optimization methods for SDN and SR networks, new optimization objectives, such as MLU and ALU minimization, new tools such as a load balancing comparator, an SR simulator, SDN controller connectivity, among others. Several improvement were also introduced in the EA engine as well as new interfaces. Almost all the framework's code had to be rewritten to support the new functionalities. All the experiments and results presented in the previous chapters were obtained using the NetOpt framework.

6.3 NetOpt Architecture

At the heart of NetOpt lies a set of routing simulators, Figure 6.1, whose role is to implement routing decisions and distribute traffic along the available resources. As observed in Figure 6.1, there are three types of simulators: Link State simulator, SDN simulator and SR simulator. Each simulator considers a traffic load balancing strategy and, in some cases, additional rules to distribute the given traffic demands into the network topology. NetOpt offers four load balancing strategies, namely, ECMP, DEFT, PEFT and Unique Shortest Path. The last is only used for the SR multiplane link failure optimization, Section 5.5, and although solutions do not effectively translate a link weights configuration, it uses additional criteria to select forwarding paths. If there is more than one shortest path from a current node to the destination, the next hop is chosen as the one with lower weight and lower ID number sequentially. Although this strategy is not entirely derived from link weights, it allows to obtain hop-by-hop unique shortest paths with equivalent quality to those that would be obtained by unique shortest paths.

The role of the network simulator is also to provide network load matrices used to evaluate the congestion or any of the other cost function values (delay, MLU, ALU, etc.). As shown in Figure 6.1, a common API allows EAs to use network simulators and evaluate solutions. NetOpt has a conversion layer between its core and EA engines. This allow to use any EA optimization library without the need to rewrite the entire code. An EA solution is first converted to an EA library independent instance, this instance is evaluated by the simulator and results are returned to the EA engine. When

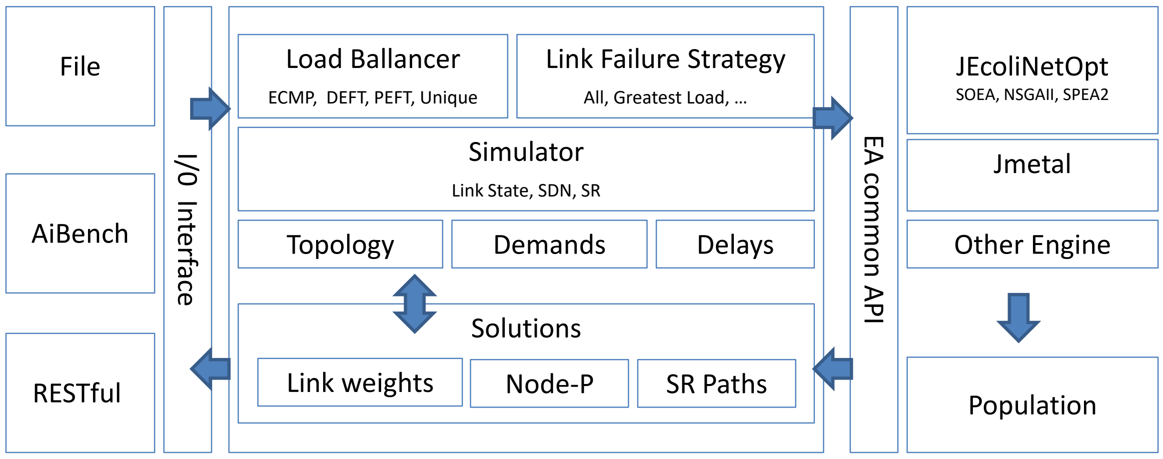
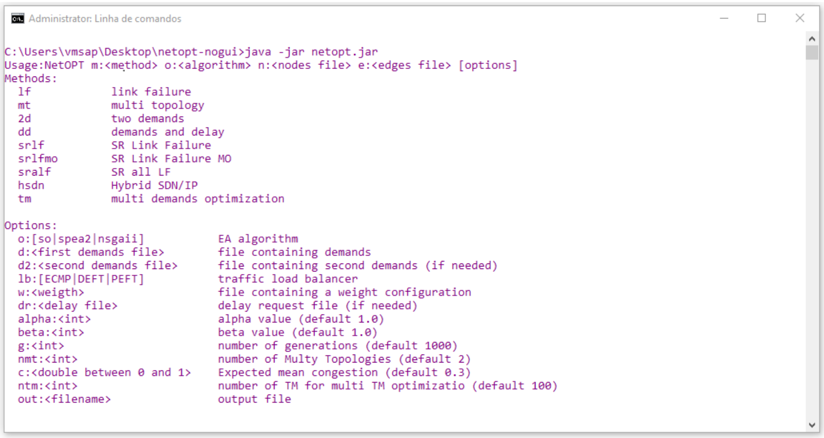


Figure 6.1: *NetOpt architecture*

the EA terminates the evolutionary process, all non dominated solutions contained in the final population are returned as a solution set and all its elements follow the same conversion process.

Solutions may encode distinct information. For example, while a solution for SR networks are decoded into IGP weights, SR paths and traffic splitting ratios, a solution for hybrid IP/SDN networks with k hybrid nodes are decoded as IGP link weights, traffic splitting ratios and a representation identifying the hybrid nodes. All representations are easily accessible through the user interface.

NetOpt relies most exclusively on an altered version of the JEcoli library [31] as optimization engine. There are significant differences between the altered version of JEcoli developed in this work, JEcoliNetOpt, and the original main development branch (version 3). First, in the original version of the library, acceptable values for a gene at position a i are taken in a range $[L_i - 1, U_i - 1[$ where L_i and U_i are respectively the lower and upper gene value bounds. In JEcoliNetOpt this range is $[L_i, U_i]$ and consequently user defined link weights ranges are the same for both EA representations and link state configurations. Second, operators in JEcoliNetOpt accept as additional parameter the position range where they should be applied allowing to aggregate in a single vector distinct representations. Third, it is possible to define a function to be applied to a solution after crossover and mutation operations produce new individuals. This enables to further evolve solutions, for example, by allowing to apply a local search on new generated solutions. JEcoliNetOpt also includes new representations and operators such as Integer Permutation and Hybrid Representation with the operators described in Section 4.4.2.



```
C:\Users\vmsap\Desktop\netopt-nogui>java -jar netopt.jar
Usage:NetOPT m:<method> o:<algorithm> n:<nodes file> e:<edges file> [options]
Methods:
  lf             link failure
  mt             multi topology
  2d             two demands
  dd             demands and delay
  srlf           SR Link Failure
  srlfmo         SR Link Failure MO
  sralf          SR all LF
  hsdn           Hybrid SDN/IP
  tm             multi demands optimization

Options:
  o:[so|spea2|nsgaii]         EA algorithm
  d:<first demands file>      file containing demands
  d2:<second demands file>    file containing second demands (if needed)
  lb:[ECMP|DEFT|PEFT]         traffic load balancer
  w:<weigth>                  file containing a weight configuration
  dr:<delay file>             delay request file (if needed)
  alpha:<int>                 alpha value (default 1.0)
  beta:<int>                  beta value (default 1.0)
  g:<int>                     number of generations (default 1000)
  nmt:<int>                   number of Multy Topologies (default 2)
  c:<double between 0 and 1>  Expected mean congestion (default 0.3)
  ntm:<int>                   number of TM for multi TM optimizatio (default 100)
  out:<filename>              output file
```

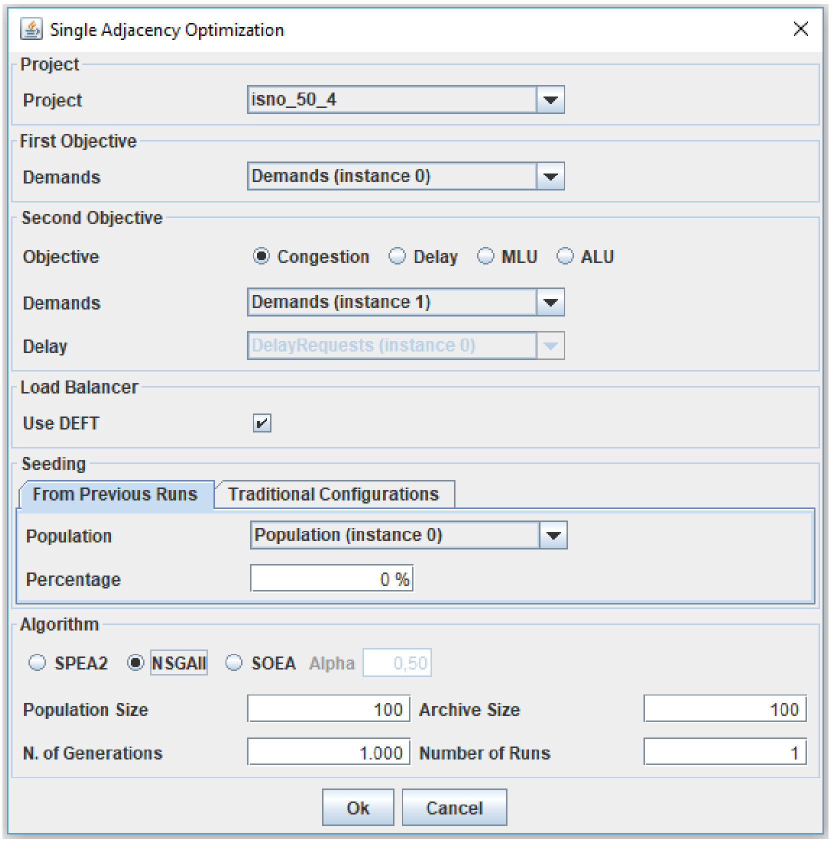
Figure 6.2: *NetOpt command line interface*

As depicted in Figure 6.1, there are three possible ways to interact with NetOpt: throught a graphical interface based on AIBench [43], by command line, Figure 6.2, and via RESTful API. The RESTful API provides limited functionalities as it is still under development. It is aimed to be used by SDN controllers or Path Computation Elements (PCE) modules to obtained new optimized configurations. The command line interface was developed to be used on a cluster context, being the main interface employed to run the experiments. The graphic interface, on the other hand, offers a collection of tools which assist networks administrator in choosing most adequate solutions and at the same time it simplifies user's experience.

6.4 NetOpt Optimization Tools

One of the benefits of the AIBench interface is that all results are kept within a clipboard which allows to carry out additional analyzes. The dialog for a SAPL-SR optimization, Figure 6.3, is one example of the available interfaces. Network topologies, traffic demand matrices and even stored configurations from previous experiments can be loaded into the clipboard and become available as parameters to run new experiments. A user can therefore use solutions from other experiments, for example, to seed a new optimization. Topologies can be loaded from different file formats: Brite, GraphML and GML. The last two enable to load custom topologies built with graph editor tools such as yEd[1] or Gephi[2] but also to experiment in real backbone topologies such as those available at The Internet Topology Zoo[3].

[1]https://www.yworks.com
[2]https://gephi.org/
[3]http://www.topology-zoo.org



Single Adjacency Optimization

Project

Project	isno_50_4

First Objective

Demands	Demands (instance 0)

Second Objective

Objective	[✓ Congestion] Delay MLU ALU
Demands	Demands (instance 1)
Delay	DelayRequests (instance 0)

Load Balancer

Use DEFT	✓

Seeding

From Previous Runs | Traditional Configurations

Population	Population (instance 0)
Percentage	0 %

Algorithm

SPEA2 [✓ NSGAII] SOEA Alpha 0,50

Population Size	100	Archive Size	100
N. of Generations	1.000	Number of Runs	1

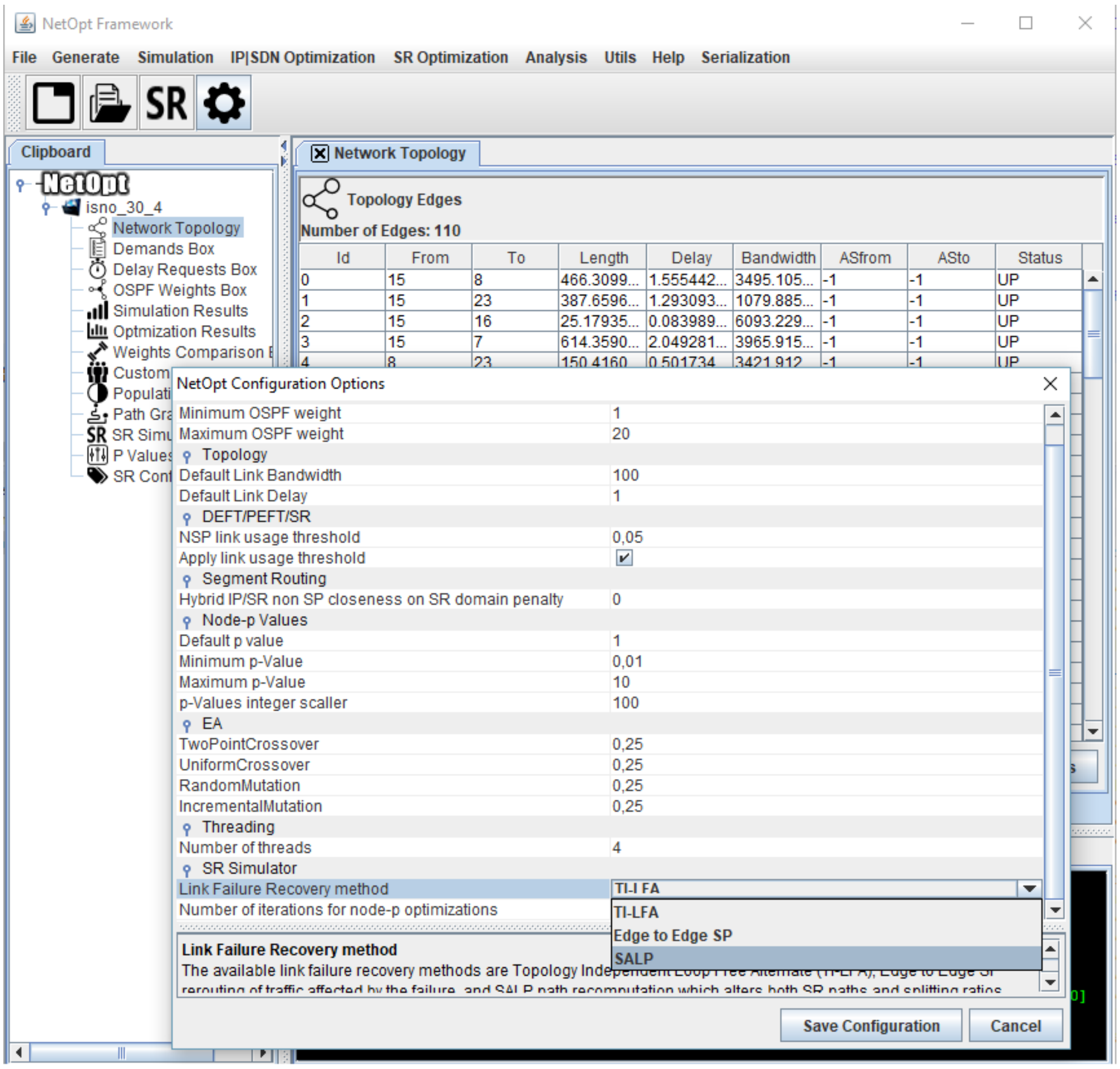
Figure 6.3: *SALP-SR optimization dialog*

The first optimization objective is always to minimize congestion for particular traffic necessities. The second objective can be selected from new traffic demands, MLU or ALU. As also observed int the example of Figure 6.3, the interface allows to choose the MOEA as well as to set its running parameters. The load balancing mechanism is also configurable, DEFT or PEFT, as well as other parameters such as the minimum fraction of aggregated traffic to be forwarded in each SR path and IGP weights range.

Depending on the optimization to be run, other options are made available. For example, single link failure optimization requires the choice of a failing link strategy. The available strategies are:

- All links: The obtained solutions are such that the congestion is minimized for the eventuality of any single failure;
- User Selected: The user selects the link for which its failure congestion should be minimized;
- Centrality: The failing link is such that it occurs in the largest number of shortest paths. This selection strategy is dynamic and depends on the configuration being evaluated;

- Highest Load : The selected link, for each solution being evaluated, is the one that has the highest load;
- Higher impact on congestion: The congestion is minimized for the failure which has a greatest impact on congestion.



Number of Edges: 110

Id	From	To	Length	Delay	Bandwidth	ASfrom	ASto	Status
0	15	8	466.3099...	1.555442...	3495.105...	-1	-1	UP
1	15	23	387.6596...	1.293093...	1079.885...	-1	-1	UP
2	15	16	25.17935...	0.083989...	6093.229...	-1	-1	UP
3	15	7	614.3590...	2.049281...	3965.915...	-1	-1	UP
4	8	23	150.4160	0.501734	3421.912	-1	-1	UP

NetOpt Configuration Options

Minimum OSPF weight	1
Maximum OSPF weight	20
Topology	
Default Link Bandwidth	100
Default Link Delay	1
DEFT/PEFT/SR	
NSP link usage threshold	0,05
Apply link usage threshold	✓
Segment Routing	
Hybrid IP/SR non SP closeness on SR domain penalty	0
Node-p Values	
Default p value	1
Minimum p-Value	0,01
Maximum p-Value	10
p-Values integer scaller	100
EA	
TwoPointCrossover	0,25
UniformCrossover	0,25
RandomMutation	0,25
IncrementalMutation	0,25
Threading	
Number of threads	4
SR Simulator	
Link Failure Recovery method	TI-LFA (TI-LFA / Edge to Edge SP / SALP)
Number of iterations for node-p optimizations	

Link Failure Recovery method
The available link failure recovery methods are Topology Independent Loop Free Alternate (TI-LFA), Edge to Edge SP rerouting of traffic affected by the failure, and SALP path recomputation which alters both SR paths and splitting ratios

Figure 6.4: *NetOpt options dialog*

The last three strategies are dynamic and depend on the final configuration and traffic demands. The optimization for the failure of all individual links, one at the time, is computationally very demanding. In this context, to diminish the time needed to obtain a good quality solution, NetOpt offers several additional options (see example in Figure 6.4):

- Multi-Threading: A user defines how many threads should be used to evaluate link failures;
- Percentage of links to be evaluated: In each iteration only a subset of link failures are evaluated which include the links with the worst 3 evaluations.
- Complete evaluation probability: From time to time, during the optimization process, solutions are evaluated for the failure of all links. This probability increases with the number of iterations.

By properly setting the available options, optimizations that otherwise could take more than two weeks to be run, only require two days, such as the optimization for the failure of all link in the $Rand50_4$ topology with 50 nodes and 380 links.

6.5 NetOpt Analysis Tools

NetOpt was not only conceived as an optimization framework but also as an analysis tool. NetOpt allows, for example, to directly compare two configurations, enabling to identify both similarities and differences on routing paths, simulate hybrid IP/SDN networks as well as SR networks, and directly alter a configuration scenario. In this section we describe the NetOpt main analysis tools.

6.5.1 Hybrid IP/SDN Networks

The framework allows to simulate hybrid IP/SDN topologies. Any topology node can be set as legacy or SDN switch, as depicted in Figure 6.5. While legacy nodes only forward traffic along shortest paths and perform ECMP load balancing, SDN nodes can additionally forward traffic on non shortest links and perform unequal load balancing. Incremental deployments of hybrid IP/SD networks can be optimized or evaluated using the model proposed in Chapter 4 where nodes with hybrid functionalities are predefined.

Presently such feature is not applicable to SR enabled topologies. SR requires all nodes to be SR enabled or, alternatively, SR can be configured for a connected subset of nodes with SR capability, an SR domain. Thus, all nodes in a SR paths need to be able to read and interpret segment headers.

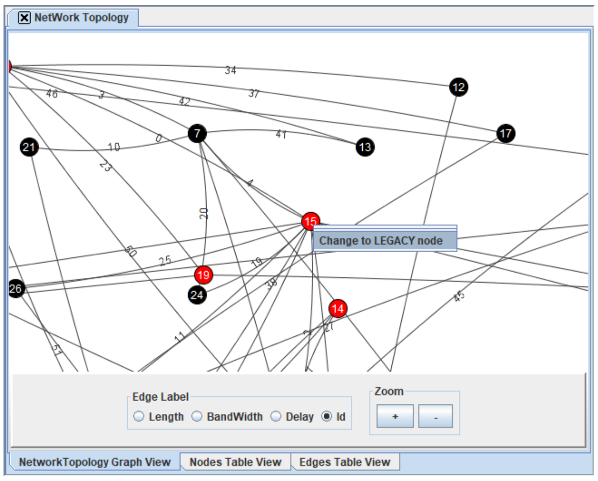


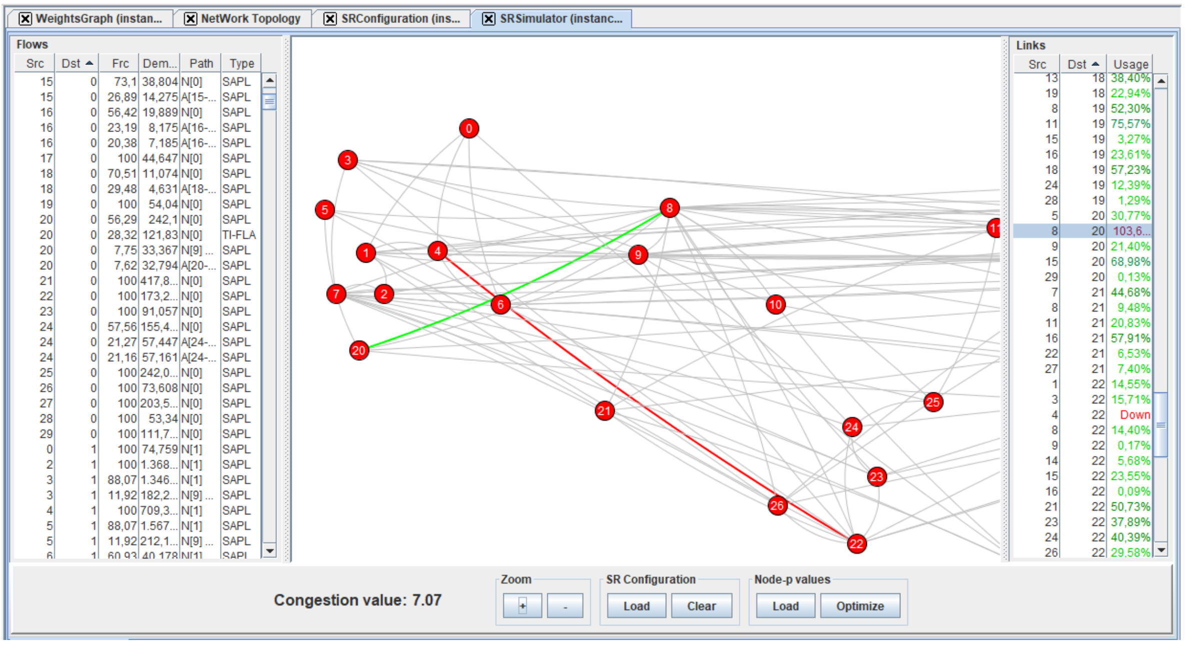
Figure 6.5: *Hybrid IP/SDN network*

6.5.2 SR Simulator

The NetOpt framework offers an SR simulator which allows to interactively evaluate SR configurations with different traffic necessities and link failure scenarios. SR paths, link loads and network congestion values are updated whenever changes are introduced in the network environment, as seen in Figure 6.6. For example, when a user sets a link to a failing state, the simulator automatically installs recovery SR paths for the affected traffic and updates links utilization ratios and the overall network congestion. By default TI-LFA is applied, however, other link failure path recovery strategies can be selected from those presented and analyzed in Section 5.5. Only the multiplane recovery strategy needs to be pre-computed using the optimization tools. Besides global traffic necessities defined from a traffic matrix, individual flows can also be loaded into the simulator. A user is only required to provide the volume of traffic to be routed and to set the forwarding path. Similarly, individual or aggregated flows can be removed from the simulator. The simulator also allows to alter SR paths from the installed configuration. The SR simulator makes available a set of features that support and assist network administrators decisions.

6.5.3 SR Simulator - Use Case Example

A typical utilization of the SR Simulator includes several steps which are nextly described:



Flows

Src	Dst	Frc	Dem...	Path	Type
15	0	73,1	38,804	N[0]	SAPL
15	0	26,89	14,275	A[15-...	SAPL
16	0	56,42	19,889	N[0]	SAPL
16	0	23,19	8,175	A[16-...	SAPL
16	0	20,38	7,185	A[16-...	SAPL
17	0	100	44,647	N[0]	SAPL
18	0	70,51	11,074	N[0]	SAPL
18	0	29,48	4,631	A[18-...	SAPL
19	0	100	54,04	N[0]	SAPL
20	0	56,29	242,1	N[0]	SAPL
20	0	28,32	121,83	N[0]	TI-FLA
20	0	7,75	33,367	N[9] ...	SAPL
20	0	7,62	32,794	A[20-...	SAPL
21	0	100	417,8...	N[0]	SAPL
22	0	100	173,2...	N[0]	SAPL
23	0	100	91,057	N[0]	SAPL
24	0	57,56	155,4...	N[0]	SAPL
24	0	21,27	57,447	A[24-...	SAPL
24	0	21,16	57,161	A[24-...	SAPL
25	0	100	242,0...	N[0]	SAPL
26	0	100	73,608	N[0]	SAPL
27	0	100	203,5...	N[0]	SAPL
28	0	100	53,34	N[0]	SAPL
29	0	100	111,7...	N[0]	SAPL
0	1	100	74,759	N[1]	SAPL
2	1	100	1.368...	N[1]	SAPL
3	1	88,07	1.346...	N[1]	SAPL
3	1	11,92	182,2...	N[9] ...	SAPL
4	1	100	709,3...	N[1]	SAPL
5	1	88,07	1.567...	N[1]	SAPL
5	1	11,92	212,1...	N[9] ...	SAPL
6	1	60,93	40,178	N[1]	SAPL

Links

Src	Dst	Usage
13	18	38,40%
19	18	22,94%
8	19	52,30%
11	19	75,57%
15	19	3,27%
16	19	23,61%
18	19	57,23%
24	19	12,39%
28	19	1,29%
5	20	30,77%
8	20	103,6...
9	20	21,40%
15	20	68,98%
29	20	0,13%
7	21	44,68%
8	21	9,48%
11	21	20,83%
16	21	57,91%
22	21	6,53%
27	21	7,40%
1	22	14,55%
3	22	15,71%
4	22	Down
8	22	14,40%
9	22	0,17%
14	22	5,68%
15	22	23,55%
16	22	0,09%
21	22	50,73%
23	22	37,89%
24	22	40,39%
26	22	29,58%

Congestion value: 7.07

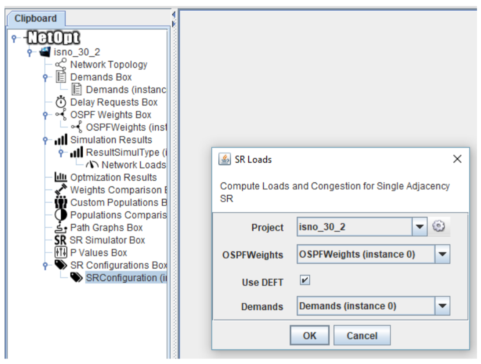
Figure 6.6: *NetOpt SR simulator*

1. Start a new project by loading a network topology and a traffic demand matrix. Alternatively, it is also possible to generate random traffic necessities by identifying an expected average link utilization.

2. The SR simulator requires a link weights configuration. Such a configuration can be generated using traditional configuration schemes, such as *InvCap*, or optimized for specific traffic demands. If an optimized configuration is preferred, we suggest to select as second objective the MLU, use DEFT load balancing, and the NSGAII optimization algorithm (Select the following list of options from the main menu `SR Optimization > Single Adjacency Optimization`). The remaining configuration options should be adequate for most networks.

3. After producing a link weights configuration, it needs to be converted into a SR configuration (`Simulation > Compute > SR Loads`), Figure 6.7a. Besides providing information regarding the network congestion and link loads for the SR configuration with the selected set of link weights, the framework computes and provides the associated SR paths configuration and parallel paths load balancing ratios.

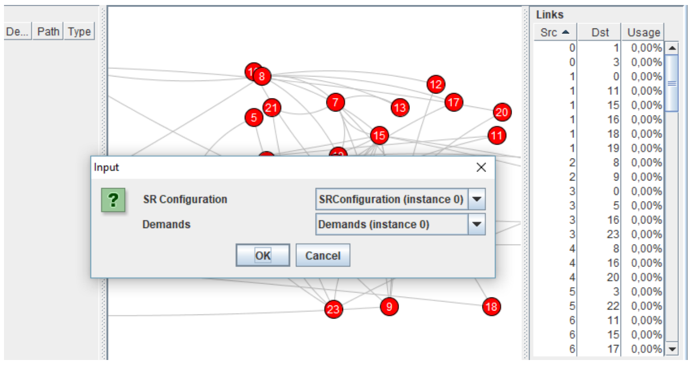
4. A new SR simulator can be created from the simulation menu (`Simulation > SR simulator`) or from the toolbar. The user is next asked to select the required parameters (Project and Link Weights Configuration). The SR simulator view is opened directly from the clipboard tree, enabling to load the SR paths configura-



Compute Loads and Congestion for Single Adjacency SR

Project: isno_30_2
OSPFWeights: OSPFWeights (instance 0)
Use DEFT: ✓
Demands: Demands (instance 0)

(a) *Creating a SR configuration from a set of link weight*



SR Configuration: SRConfiguration (instance 0)
Demands: Demands (instance 0)

Links

Src	Dst	Usage
0	1	0,00%
0	3	0,00%
1	0	0,00%
1	11	0,00%
1	15	0,00%
1	16	0,00%
1	18	0,00%
1	19	0,00%
2	8	0,00%
2	9	0,00%
3	0	0,00%
3	5	0,00%
3	16	0,00%
3	23	0,00%
4	8	0,00%
4	16	0,00%
4	20	0,00%
5	3	0,00%
5	22	0,00%
6	11	0,00%
6	15	0,00%
6	17	0,00%

(b) *Open a new SR simulator*

Figure 6.7: *SR simulator use case*

tion and traffic demands, Figure 6.7b.

5. The SR simulator allows to set links down by right clicking the link in the topology view or in the link table. The simulator automatically recomputes recovery paths for the affected traffic using TI-LFA, edge-to-edge shortest paths or a SALP-SR recomputation. The default paths recovery option is selected in the global settings dialog or, alternatively, it can be set in the `netopt.conf` configuration file.

6. It is possible to reduce the congestion produced by link failures, or by an inadequate link weights configuration, by optimizing the node-p values. The optimization process may take from a few seconds to a few minutes depending on the topology size. After the optimization is concluded, the SR paths configuration, link loads and network congestion values are updated. Previously obtained node-p values configurations can alternatively be loaded into the simulator.

7. The SR simulator also enables to evaluate the installed configuration with new traffic necessities. For such a task, it is sufficient to clear the the configured flows and load a new traffic demands matrix.

With such a tool, a network administrator can quickly and easily evaluate different SR configurations under distinct conditions and identify potential problems or limitations, such as overutilized links.

6.5.4 Traffic Load Balancing Analysis

The NetOpt framework also provides tools to analyze traffic load balancing alternatives, as can be observed in Figure 6.8. It is possible, for example, to compare how

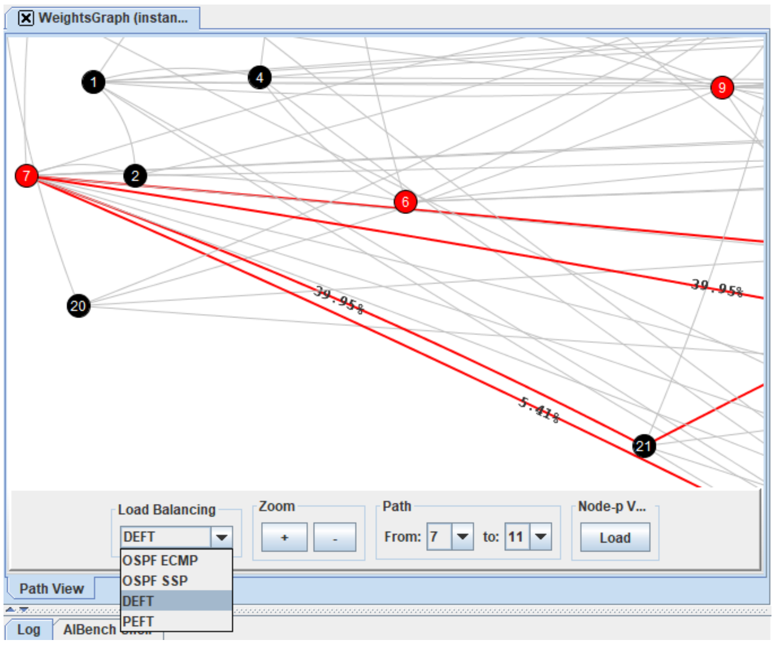


Figure 6.8: *Traffic load balancing analysis*

a defined weights configuration affects traffic load balancing in the context of distinct technologies: OSPF/ECMP and hybrid IP/SDN, both with PEFT or DEFT traffic load balancing functions. The traffic load balancing analysis tool also enables to simulate both full SDN deployments and incremental ones. Nodes can easily be switched from legacy to SDN or vice-versa. Additionally, it is also possible to compare and analize the effects on traffic load balancing of different node-p values configurations, which may result from an optimization process or be user defined.

6.6 Conclusions

The NetOpt framework encompasses all optimization methods devised in this work. As the primary tool used to perform the experiments, it underwent a large amount of validation and verification tests to ensure its correctness. Although there is still much to be done, the NetOpt framework already presents itself as a relevant tool to support administrators in the tasks of optimizing network resources utilization. It supports a myriad set of scenarios that simulate day-to-day network's events. NetOpt also offers a set of features which enables to analyze and compare different routing solutions and make sustained configuration decisions. As an open source project, the framework can be freely used and altered to meet individual network administrators or researchers needs.

Chapter 7

Conclusions

This final chapter presents the conclusions of the thesis, as well as future research directions. This chapter highlights the main contributions from the work developed in the thesis, answering to the research objectives proposed in the introduction, Chapter 1. Global remarks on the devised work are also presented, and, finally, the last section points out possible future research directions, proposing future research ideas for the developed work, as well as addressing topics out of the scope of this thesis, but that can add value when properly integrated.

7.1 Achievements on the Research Objectives

- **Research Question 1: Which Multi-Objective Evolutionary Algorithm provides the best solutions for the multi-objective OSPF weight setting problem?**
  - Multi-objective problems do not usually have a single global solution, but a set of solutions, each with a different tradeoff between the optimization objectives. Analogously, multi-objective OSPF weight setting problems have a set of solutions, link weights configurations, with distinct characteristics depending on the used Multi-Objective Evolutionary Algorithm (MOEA). The first step in the research was, therefore, to identify which MOEA (a Single Objective Evolutionary Algorithm (SOEA) that linearly aggregates the objective functions, the Strength Pareto Evolutionary Algorithm (SPEA2) and Non-dominated Sorting Genetic Algorithm (NSGAII)) provides the best set of solutions considering the convergence, but also the scatteredness of the

solutions. In that context, in Chapter 3, we formulated two OSPF weight setting problems, each with two objectives. The first aims to find solutions that optimize the network resources utilization in two distinct moments in time modeled by distinct traffic necessities. The second problem aims to find solutions that optimize the network's resources utilization before and after a single link failure. By comparing the solutions provided by the SOEA, SPEA2 and NSAGII for both problems, we concluded that, in both cases, the NSGAII is the MOEA that delivers the best set of solutions.

- **Research Question 2: How to optimize resources utilization in hybrid IP/SDN networks?**
  - Hybrid IP/SDN networks combine the benefits of traditional link state routing with the flexibility of Software Defined Networking (SDN). While traditional link state routing provides scalability and robustness to networks configuration, SDN allows to overcome some Interior Gateway Protocols (IGP) restrictions. SDN enables to forward traffic along non-shortest paths and at the same time makes possible to perform non-equal traffic load balancing, thus improving the network's resources utilization. In Chapter 4, we proposed two routing models for hybrid IP/SDN networks and experimentally showed that they are able to provide a better usage of the network available resources than optimized OSPF routing configurations. We also showed that by introducing a relative small number of SDN enabled devices in the network (less than 20% of the devices) it is possible to capitalize on the advantages offered by both technologies.
- **Research Question 3: How to optimize resources utilization and minimize congestion that results from changes in traffic demands and link failures in hybrid IP/SDN networks?**
  - Traffic necessities vary over time and a hybrid IP/SDN configuration suited for a particular traffic demand may not be adequate to another. To overcome a poor utilization of network resources due to changes in traffic requirements we extended in Chapter 4 our hybrid IP/SDN routing model with a mechanism able to correct traffic load balancing. We experimentally showed that by optimizing a set of real values, node-p values, it is possible to correct in most cases hop-by-hop traffic load balancing thus improving the network performance. Additionally, we showed that by optimizing the initial hybrid IP/SDN configuration with two traffic matrices that bound a set of possible

traffic demands, such a mechanism provides a good solution to address the traffic variation problem in a network without performing any changes on the installed IGP configuration.

  - When a topology link fails, the IGP automatically reroutes the affect traffic by recomputing the shortest paths. As a consequence, some topology links may become overutilized leading to a poor network performance. We showed that by correcting hop-by-hop traffic load balancing it is possible to reduce the impact of such fault events and reset the network to an acceptable operating state. Here again, the traffic load balancing correction is achieved by optimizing the set of node-p values. Additionally, such an optimization can be performed in a relative short amount of time, thereby shortening the duration of the network instability period.

- **Research Question 4: How to optimize resources utilization in Segment Routing networks while minimizing label stack depth?**

  - Taking as a starting point the same principles used for the optimization of hybrid IP/SDN networks, we proposed in Chapter 5 an Evolutionary Computation optimization procedure for Segment Routing (SR), the Single Adjacency Label Path (SALP). Besides being able to optimize the network resources utilization, the SALP do so by using at the most three labels, or segments, to configure each edge-to-edge SR forwarding path. A solution configuration provided by the optimization model encompasses the IGP link weights, the SR paths and the split ratios of traffic between parallel paths.

- **Research Question 5: How to optimize resources utilization and minimize congestion that results from changes in traffic demands and link failure in SR networks while minimizing label stack depth?**

  - The strategy to address traffic variations that undermine the network performance is similar to the one proposed for hybrid IP/SDN networks, but instead of correcting hop-by-hop traffic load balancing ratios, the corrections of traffic splitting is performed at the edge nodes by adjusting the division of traffic between parallel paths. In Chapter 5, we showed that such a procedure is capable of improving, in most cases, the network resources utilization while preserving IGP configurations and SR paths. As a consequence the imposed limit of three labels to configure SR paths is maintained.
  - The solution proposed by the Internet Engineering Task Force (IETF) to address the rerouting of traffic due to a link failure, the Topology Independent

Link-Free Alternate (TI-LFA) while able to reestablish connectivity in less than 50 milliseconds, is unable to ensure a good utilization of network resources after the IGP convergence. As referred earlier, after a link failure, the IGP automatically reroutes the traffic affected by the fault by recomputing the shortest paths, which may lead to a poor distribution of traffic. In such a context we compared TI-LFA with other possible alternatives and experimentally showed that a SALP recomputation conjugated with an optimization of traffic load balancing between parallel paths, provides a good recovery solution which can quickly be implemented. Furthermore, we also proposed an alternative offline approach, whose solutions need to be pre-computed and stored into a database managed by a Path Computation Element (PCE) or SDN controller. By performing a multiplane optimization, the offline approach is able to further improve the SALP results but can only be applied in the case of a single link failure. After a link failure, connectivity can be reestablished using TI-LFA, and as soon as the IGP converges, optimized SR paths can be installed at edge nodes, achieving this way both desired goals: a shorten reaction time and a good utilization of network resources.

- **Research Question 6: Is it possible to develop an useful application framework able to assist network administrators in the optimization of routing configurations in OSPF, SDN and SR based scenarios?**
  - The NetOpt framework was not only developed as a tool to run the experiments whose results were presented in this thesis, but also as a tool that can be used by other researchers and network administrators. Besides being able to replicate all the experiments, a researcher can freely adopt, adapt and modify the source code to attend its needs. Network administrators, on the other hand, can use the NetOpt framework to investigate if the provided solutions are suited to solve the day to day problems that he must face, and if its the case, hopefully deploy such solutions in the real world networking scenarios.

7.2 Final Remarks

Traffic engineering is an important mechanism for Internet network providers seeking to optimize network performance and traffic delivery. Routing optimization plays a key role in traffic engineering, finding efficient routes so as to achieve the desired network performance. The aim in this thesis was to study and propose new solutions in the context of distinct intradomain routing technologies, namely for traditional Link-State protocols, hybrid IP/SDN networks as well as for networks with SR capable devices. The guideline that ties this work together, to improve network resources utilization by providing near optimal routing configurations, is specially important when unforeseen events disrupt the efficiency of the installed configuration.

In such a context, we proposed and evaluated different models and solutions which provide a better network performance than the one obtained when only traditionally configured intradomain routing protocols are implemented. Capitalizing in single and multi-objective Evolutionary Algorithms, we not only proposed solutions for the optimization of traffic routing in network with known conditions, but we also proposed solutions able to address unforeseen circumstances such as traffic variations and link failures. All the proposed methods were assembled into an optimization tool, NetOpt, made publicly available.

7.3 Future Research Directions

Considering the work developed in this thesis and its broad research scope, there are several topics that may benefit from further development in order to pursue a better utilization of networks resources. These topics are presented nextly:

- **Improve the computation time required for node-p values optimization**

  The node-p values optimization is performed by Evolutionary Computation algorithms, whose processing time can be excessively long to be used in larger networks. Thus, a possible future topic would consist in investigate if rather the use of linear programming can reduce the time required to obtain an optimized node-p valued configuration.

- **Node-p values extension**

  Each topology node has a same node-p value for all destinations. It is possible however to set distinct node-p values at each node that consider the distinct traffic destinations. A possible research topic may consider such differentiation of node-p values an evaluate if obtained results justify the significantly greater number of decision variables that need to be optimized. A satisfactory solution that can be obtained in a lesser amount of time may be better in practice than an optimize solution that requires more time to be deployed.

- **Node failure**

  In this work we addressed the failure of topology links in hybrid IP/SDN as well as in Segment Routing networks. There are, however, other faults that should be addressed to provide networks with a higher level of resiliency. According to the reports released by the Network Reliability Steering Committee, node failure is an important factor causing network faults. How to respond to such events is thus an important research topic. In particular, how to provide network recovery paths that enable networks to continue operating reliably after such events.

- **Multi-Topology Segment Routing**

  Multi-Topology Segment Routing has been recently proposed by the IETF enabling to compute different paths for unicast traffic, multicast traffic and different classes of service. Such paths may be assigned based on flexible criteria or an in-band network management. A future topic might consider the combination of two of our proposals, namely the optimization of routing configurations for Multi-Topology Routing in OSPF and the Segment Routing optimization model proposed in this thesis. Besides allowing to achieve a better usage of networks resources, such a routing solution may also be used to address fault events such as a node or a link failure.

- **Virtual Network Function embedding**

  The use of Network Function Virtualization (NFV) and SDN provides opportunities to offer services with lower CAPEX/OPEX for service providers and deploy new services quickly. One of the main challenges is an optimized placement of the virtualized functions based on the characteristics and available resources of the network. Another challenge is to optimize the utilization of link capacities to deliver intended traffic to nodes that perform NFV while managing the processing capacity of such nodes.

- **Multi-Objective Evolutionary Algorithm alternative configurations**

  Crossover and mutation operators as well as selection mechanisms have a major influence on the convergence of any Evolutionary Algorithm. In such a context, alternative configurations with different operators and matting selection schemes can be evaluated and results compared with the already obtained. Additionally, new EAs such as the third version of the Non-dominated Sorting Genetic Algorithm (NSGA-III) can also be evaluated in the context of the addressed problems and optimization models.

Bibliography

[1] Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications.* Prentice hall.

[2] Al-Fares, M., Radhakrishnan, S., Raghavan, B., Huang, N., and Vahdat, A. (2010). Hedera: Dynamic flow scheduling for data center networks. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI'10, pages 19–19, Berkeley, CA, USA. USENIX Association.

[3] Albrightson, B., Garcia-Luna-Aceves, J., and Boyle, J. (1994). EIGRP - a fast routing protocol based on distance vectors. In *Proceedings of the Networld/Interop 94*, pages 192–210.

[4] Alim, M. A. and Griffin, T. G. (2011). On the interaction of multiple routing algorithms. In *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies*, CoNEXT '11, pages 7:1–7:12, New York, NY, USA. ACM.

[5] Alizadeh, M., Edsall, T., Dharmapurikar, S., Vaidyanathan, R., Chu, K., Fingerhut, A., Lam, V. T., Matus, F., Pan, R., Yadav, N., and Varghese, G. (2014). Conga: Distributed congestion-aware load balancing for datacenters. *SIGCOMM Computer Communication Review*, 44(4):503–514.

[6] Altin, A., Fortz, B., and ímit, H. (2012). Oblivious ospf routing with weight optimization under polyhedral demand uncertainty. *Networks*, 60(2):132–139.

[7] Altin, A., Fortz, B., Thorup, M., and Umit, H. (2013). Intra-domain traffic engineering with shortest path routing protocols. *Annals of Operations Research*, 204(1):65–95.

[8] Applegate, D. and Cohen, E. (2003). Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and*

Protocols for Computer Communications, SIGCOMM '03, pages 313–324, New York, NY, USA. ACM.

[9] Argyraki, K., Baset, S., Chun, B.-G., Fall, K., Iannaccone, G., Knies, A., Kohler, E., Manesh, M., Nedevschi, S., and Ratnasamy, S. (2008). Can software routers scale? In *Proceedings of the ACM Workshop on Programmable Routers for Extensible Services of Tomorrow*, PRESTO '08, pages 21–26, New York, NY, USA. ACM.

[10] Atlas, A., Swallow, G., and Pan, P. (2005). Fast Reroute Extensions to RSVP-TE for LSP Tunnels. RFC 4090.

[11] Atlas, A. and Zinin, A. (2008). Basic Specification for IP Fast Reroute: Loop-Free Alternates. RFC 5286.

[12] Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., and Xiao, X. (2002). Overview and principles of internet traffic engineering. RFC 3272.

[13] Awduche, D. O., Berger, L., Gan, D.-H., Li, T., Srinivasan, D. V., and Swallow, G. (2001). RSVP-TE: Extensions to RSVP for LSP Tunnels. RFC 3209.

[14] Bashandy, A., Filsfils, C., Decraene, B., Litkowski, S., Francois, P., Voyer, D., Clad, F., and Garvia, P. C. (2018). Topology Independent Fast Reroute using Segment Routing. Internet-Draft draft-ietf-rtgwg-segment-routing-ti-lfa-00, Internet Engineering Task Force. Work in Progress.

[15] Bhatia, R., Hao, F., Kodialam, M., and Lakshman, T. V. (2015). Optimized network traffic engineering using segment routing. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 657–665.

[16] Braden, R. T., Zhang, L., Berson, S., Herzog, S., and Jamin, S. (1997). Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. RFC 2205.

[17] Briscoe, B. and Manner, J. (2014). Byte and Packet Congestion Notification. RFC 7141.

[18] Cardwell, N., Cheng, Y., Gunn, C. S., Yeganeh, S. H., and Jacobson, V. (2016). Bbr: Congestion-based congestion control. *Queue*, 14(5):50:20–50:53.

[19] Cheng, H. K., Bandyopadhyay, S., and Guo, H. (2011). The debate on net neutrality: A policy perspective. *Information Systems Research*, 22(1):60–82.

[20] Chu, C., Xi, K., Luo, M., and Chao, H. J. (2015). Congestion-aware single link failure recovery in hybrid sdn networks. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 1086–1094.

[21] Claise, B. (2004). Cisco Systems NetFlow Services Export Version 9. RFC 3954.

[22] Coello, C. A. C., Lamont, G. B., and Veldhuizen, D. A. V. (2006). *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Springer-Verlag, Berlin, Heidelberg.

[23] Coello, C. A. C., Pulido, G. T., and Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279.

[24] Corne, D. W., Jerram, N. R., Knowles, J. D., Oates, M. J., and J, M. (2001). Pesa-ii: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 283–290. Morgan Kaufmann Publishers.

[25] Cui, C. and Xu, Y. (2016). Research on load balance method in sdn. *International Journal of Grid and Distributed Computing*, 9:25–36.

[26] Dally, W. and Towles, B. (2003). *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[27] Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley-Interscience Series in Systems and Optimization. John Wiley & Sons, Chichester.

[28] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271.

[29] Dorigo, M. and Di Caro, G. (1999). New ideas in optimization. chapter The Ant Colony Optimization Meta-heuristic, pages 11–32. McGraw-Hill Ltd., UK, Maidenhead, UK, England.

[30] Ericsson, M., Resende, M., and Pardalos, P. (2002). A genetic algorithm for the weight setting problem in ospf routing. *Journal of Combinatorial Optimization*, 6(3):299–333.

[31] Evangelista, P., Maia, P., and Rocha, M. (2009). Implementing metaheuristic optimization algorithms with jecoli. In *2009 Ninth International Conference on Intelligent Systems Design and Applications*, pages 505–510.

[32] Feamster, N., Rexford, J., and Zegura, E. (2014). The road to sdn: An intellectual history of programmable networks. *SIGCOMM Computer Communication Review*, 44(2):87–98.

[33] Filsfils, C., Nainar, N. K., Pignataro, C., Cardona, J. C., and Francois, P. (2015). The segment routing architecture. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6.

[34] Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B., Litkowski, S., and Shakir, R. (2018a). Segment Routing Architecture. RFC 8402.

[35] Filsfils, C., Previdi, S., Leddy, J., Matsushima, S., and Voyer, D. (2018b). IPv6 Segment Routing Header (SRH). Internet-Draft draft-ietf-6man-segment-routing-header-15, Internet Engineering Task Force. Work in Progress.

[36] Floyd, S. (2000). Congestion Control Principles. RFC 2914.

[37] Floyd, S. and Allman, M. (2008). Comments on the Usefulness of Simple Best-Effort Traffic. RFC 5290.

[38] Ford, D. R. and Fulkerson, D. R. (2010). *Flows in Networks*. Princeton University Press, Princeton, NJ, USA.

[39] Fortz, B. and Thorup, M. (2000). Internet traffic engineering by optimizing ospf weights. In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies.*, pages 519–528. IEEE.

[40] Fortz, B. and Thorup, M. (2002). Optimizing ospf/is-is weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20(4):756–767.

[41] Fortz, B. and Thorup, M. (2003). Robust optimization of ospf/is-is weights. In *Proceedings of the International Network Optimization Conference*, pages 225–230.

[42] Gay, S., Hartert, R., and Vissicchio, S. (2017). Expect the unexpected: Sub-second optimization for segment routing. In *Proceedings of the Conference on Computer Communications (INFOCOM)*, pages 1–9. IEEE.

[43] Glez-Peña, D., Reboiro-Jato, M., Maia, P., Rocha, M., Díaz, F., and Fdez-Riverola, F. (2010). Aibench: A rapid application development framework for translational research in biomedicine. *Computer Methods and Programs in Biomedicine*, 98(2):191 – 203.

[44] Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533 – 549. Applications of Integer Programming.

[45] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

[46] Gonçalves, J. and Resende, M. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525.

[47] Gredler, H. and Goralski, W. (2005). *The Complete IS-IS Routing Protocol.* Springer.

[48] Grossman, D. B. (2002). New Terminology and Clarifications for Diffserv. RFC 3260.

[49] Guedrez, R., Dugeon, O., Lahoud, S., and Texier, G. (2016). Label encoding algorithm for mpls segment routing. In *15th International Symposium on Network Computing and Applications (NCA)*, pages 113–117. IEEE.

[50] Handel, R., Huber, M. N., and Schroder, S. (1998). *ATM Networks: Concepts, Protocols, Applications.* Addison-Wesley Longman Ltd., Essex, UK, UK, 3rd edition.

[51] Hawkinson, J. and Bates, T. (1996). Guidelines for creation, selection, and registration of an autonomous system (as). RFC 1930.

[52] Hedrick, C. L. (1989). An introduction to igrp. Technical Report.

[53] Hong, C.-Y., Kandula, S., Mahajan, R., Zhang, M., Gill, V., Nanduri, M., and Wattenhofer, R. (2013). Achieving high utilization with software-driven wan. In *Proceedings of the ACM SIGCOMM 2013 Conference*, SIGCOMM '13, pages 15–26, New York, NY, USA. ACM.

[54] Hopps, C. (2000). Analysis of an Equal-Cost Multi-Path Algorithm. RFC 2992 (Informational).

[55] Iannaccone, G., Chuah, C.-n., Mortier, R., Bhattacharyya, S., and Diot, C. (2002). Analysis of link failures in an ip backbone. In *Proceedings of the 2Nd ACM SIGCOMM Workshop on Internet Measurment*, IMW '02, pages 237–242, New York, NY, USA. ACM.

[56] Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., Zolla, J., Hölzle, U., Stuart, S., and Vahdat, A. (2013). B4: Experience with a globally-deployed software defined wan. *SIGCOMM Computer Communication Review*, 43(4):3–14.

[57] Jansen, T. and Wegener, I. (2001). On the utility of populations in evolutionary algorithms. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, GECCO'01, pages 1034–1041, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[58] Jones, K. L., Lustig, I., Farvolden, J. M., and Powell, W. B. (1993). Multicommodity network flows: The impact of formulation on decomposition. *Mathematical Programming*, 62:95–117.

[59] Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, A. Y. (2002). An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892.

[60] Katta, N., Hira, M., Kim, C., Sivaraman, A., and Rexford, J. (2016). Hula: Scalable load balancing using programmable data planes. In *Proceedings of the Symposium on SDN Research*, SOSR '16, pages 10:1–10:12, New York, NY, USA. ACM.

[61] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *SCIENCE*, 220(4598):671–680.

[62] Knowles, J. D. and Corne, D. W. (2000). Approximating the nondominated front using the pareto archived evolution strategy. *Evol. Comput.*, 8(2):149–172.

[63] Lammich, P. and Sefidgar, S. R. (2016). Formalizing the edmonds-karp algorithm. In *ITP*, volume 9807 of *Lecture Notes in Computer Science*, pages 219–234. Springer.

[64] Lin, Y., Teng, H., Hsu, C., Liao, C., and Lai, Y. (2016). Fast failover and switchover for link failures and congestion in software defined networks. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6.

[65] Malkin, G. S. (1998). RIP Version 2. RFC 2453.

[66] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: Enabling innovation in campus networks. *SIGCOMM Computer Communication Review*, 38(2):69–74.

[67] Medhi, D. and Ramasamy, K. (2017). *Network Routing, Second Edition: Algorithms, Protocols, and Architectures.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition.

[68] Medina, A., Lakhina, A., Matta, I., and Byers, J. (2001). Brite: Universal topology generation from a user"s perspective. Technical report, Boston, MA, USA.

[69] Medina, A., Taft, N., Salamatian, K., Bhattacharyya, S., and Diot, C. (2002). Traffic matrix estimation: Existing techniques and new directions. *SIGCOMM Computer Communication Review*, 32(4):161–174.

[70] Michalewicz, Z. (2010). *How to Solve It: Modern Heuristics 2e*. Springer-Verlag, Berlin, Heidelberg.

[71] Mijumbi, R., Serrat, J., Gorricho, J., Bouten, N., Turck, F. D., and Boutaba, R. (2016). Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys and Tutorials*, 18(1):236–262.

[72] Mirtorabi, S., Nguyen, L., Roy, A., Psenak, P., and Pillay-Esnault, P. (2007). Multi-Topology (MT) Routing in OSPF. RFC 4915.

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

[74] Movsichoff, B. A., Lagoa, C. M., and Che, H. (2005). Decentralized optimal traffic engineering in connectionless networks. *IEEE Journal on Selected Areas in Communications*, 23(2):293–303.

[75] Moy, J. (1998). OSPF Version 2. RFC 2328.

[76] Nucci, A., Cruz, R., Taft, N., and Diot, C. (2004). Design of igp link weight changes for estimation of traffic matrices. In *IEEE Infocom*, Hong Kong.

[77] Nucci, A., Sridharan, A., and Taft, N. (2005). The problem of synthetically generating ip traffic matrices: Initial recommendations. *SIGCOMM Computer Communication Review*, 35(3):19–32.

[78] O'Dell, M. D., Malcolm, J., McManus, J., Awduche, D. O., and Agogbua, J. (1999). Requirements for Traffic Engineering Over MPLS. RFC 2702.

[79] Papagiannaki, K., Taft, N., and Lakhina, A. (2004). A distributed approach to measure ip traffic matrices. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, IMC '04, pages 161–174, New York, NY, USA. ACM.

[80] Pereira, V., Rocha, M., Cortez, P., Rio, M., and Sousa, P. (2013a). A framework for robust traffic engineering using evolutionary computation. In *Proceedings of the 7th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security: Emerging Management Mechanisms for the Future Internet - Volume 7943*, AIMS'13, pages 1–12, Berlin, Heidelberg. Springer-Verlag.

[81] Pereira, V., Rocha, M., and Sousa, P. (2015a). Automated network resilience optimization using computational intelligence methods. In *IDC*, volume 616 of *Studies in Computational Intelligence*, pages 485–495. Springer.

[82] Pereira, V., Rocha, M., and Sousa, P. (2016). Optimizing load balancing routing mechanisms with evolutionary computation. In *Intelligent Environments (Workshops)*, volume 21 of *Ambient Intelligence and Smart Environments*, pages 298–307. IOS Press.

[83] Pereira, V., Rocha, M., and Sousa, P. (2017a). Optimizing segment routing using evolutionary computation. In *FNC/MobiSPC*, volume 110 of *Procedia Computer Science*, pages 312–319. Elsevier.

[84] Pereira, V., Rocha, M., and Sousa, P. (2018). Segment routing single link failure congestion optimization. In *ICETE (1)*, pages 242–249. SciTePress.

[85] Pereira, V., Sousa, P., Cortez, P., Rio, M., and Rocha, M. (2013b). Robust optimization of intradomain routing using evolutionary algorithms. In *DCAI*, volume 217 of *Advances in Intelligent Systems and Computing*, pages 201–208. Springer.

[86] Pereira, V., Sousa, P., Cortez, P., Rio, M., and Rocha, M. (2015b). Comparison of single and multi-objective evolutionary algorithms for robust link-state routing. In *EMO (2)*, volume 9019 of *Lecture Notes in Computer Science*, pages 573–587. Springer.

[87] Pereira, V., Sousa, P., and Rocha, M. (2017b). Evolutionary computation at work for the optimization of link state routing protocols. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 299–300. ACM.

[88] Perrot, N., Benhamiche, A., Carlinet, Y., and Gourdin, E. (2018). *Future Networks: Overview of Optimization Problems in Decision-Making Procedures*, pages 177–207. IGI Global.

[89] Pietrabissa, A., Priscoli, F. D., and Lombardi, V. (2004). Edge router congestion control with congestion estimation. In *2004 43rd IEEE Conference on Decision and Control (CDC)*, volume 3, pages 2370–2371 Vol.3.

[90] Rekhter, Y., Hares, S., and Li, T. (2006). A Border Gateway Protocol 4 (BGP-4). RFC 4271.

[91] Rocha, M., Sousa, P., Cortez, P., and Rio, M. (2011). Quality of service constrained routing optimization using evolutionary computation. *Applied Soft Computing*, 11(1):356–364.

[92] Roughan, M. (2005). Simplifying the synthesis of internet traffic matrices. *SIGCOMM Computer Communication Review*, 35(5):93–96.

[93] Sá, T., Rocha, M., and Sousa, P. (2010). Tools for traffic engineering on ip networks. In *10ª Conferência sobre Redes de Computadores*, pages 13–18.

[94] Salsano, S., Ventre, P. L., Prete, L., Siracusano, G., Gerola, M., and Salvadori, E. (2014). Oshi - open source hybrid ip/sdn networking (and its emulation on mininet and on distributed sdn testbeds). In *2014 Third European Workshop on Software Defined Networks*, pages 13–18.

[95] Sandhya, Sinha, Y., and Haribabu, K. (2017). A survey: Hybrid sdn. *Journal of Network and Computer Applications*, 100:35–55.

[96] Schaffer, J. D. (1984). *Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms (Artificial Intelligence, Optimization, Adaptation, Pattern Recognition)*. PhD thesis, Nashville, TN, USA. AAI8522492.

[97] Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 93–100, Hillsdale, NJ, USA. L. Erlbaum Associates Inc.

[98] Sousa, P., Cortez, P., Rio, M., and Rocha, M. (2011). Traffic engineering approaches using multicriteria optimization techniques. In *Wired/Wireless Internet Communications*, pages 104–115, Berlin, Heidelberg. Springer Berlin Heidelberg.

[99] Sousa, P., Pereira, V., Cortez, P., Rio, M., and Rocha, M. (2017). A framework for improving routing configurations using multi-objective optimization mechanisms. *Journal of Communications Software and Systems*, 12(3):145–156.

[100] Sousa, P., Rocha, M., Cortez, P., and Rio, M. (2008). Multiconstrained optimization of networks with multicast and unicast traffic. In *Proceedings of the 11th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services: Management of Converged Multimedia Networks and Services*, MMNS '08, pages 139–150, Berlin, Heidelberg. Springer-Verlag.

[101] Sousa, P., Rocha, M., Rio, M., and Cortez, P. (2006). Efficient ospf weight allocation for intra-domain qos optimization. In *Proceedings of the 6th IEEE International Conference on IP Operations and Management*, IPOM'06, pages 37–48, Berlin, Heidelberg. Springer-Verlag.

[102] Sousa, P., Rocha, M., Rio, M., and Cortez, P. (2007). Class-based ospf traffic engineering inspired on evolutionary computation. In *Wired/Wireless Internet Communications*, pages 141–152, Berlin, Heidelberg. Springer Berlin Heidelberg.

[103] Sridharan, A., Guerin, R., and Diot, C. (2005). Achieving near-optimal traffic engineering solutions for current ospf/is-is networks. *IEEE/ACM Transactions on Networking*, 13(2):234–247.

[104] Srinivas, N. and Deb, K. (1994). Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248.

[105] Srivastava, S., Agrawal, G., Pioro, M., and Medhi, D. (2005). Determining link weight system under various objectives for ospf networks using a lagrangian relaxation-based approach. *IEEE Transactions on Network and Service Management*, 2(1):9–18.

[106] Stevens, W. R. (1993). *TCP/IP Illustrated : The Protocols*, volume 1. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[107] Suchara, M., Xu, D., Doverspike, R., Johnson, D., and Rexford, J. (2011a). Network architecture for joint failure recovery and traffic engineering. In *Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '11, pages 97–108, New York, NY, USA. ACM.

[108] Suchara, M., Xu, D., Doverspike, R., Johnson, D., and Rexford, J. (2011b). Network architecture for joint failure recovery and traffic engineering. In *Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '11, pages 97–108, New York, NY, USA. ACM.

[109] Sun, X., Jia, Z., Zhao, M., and Zhang, Z. (2016). Multipath Load Balancing in SDN/OSPF Hybrid Network. In Gao, G. R., Qian, D., Gao, X., Chapman, B., and Chen, W., editors, *13th IFIP International Conference on Network and Parallel Computing (NPC)*, volume LNCS-9966 of *Network and Parallel Computing*, pages 93–100, Xi'an, China. Springer International Publishing. Part 3: Scheduling and Load-Balancing.

[110] Tanha, M., Sajjadi, D., Ruby, R., and Pan, J. (2018). Traffic engineering enhancement by progressive migration to sdn. *IEEE Communications Letters*, 22(3):438–441.

[111] Thomas, B., Andersson, L., and Minei, I. (2007). LDP Specification. RFC 5036.

[112] Tilmans, O. and Vissicchio, S. (2014). Igp-as-a-backup for robust sdn networks. In *10th International Conference on Network and Service Management (CNSM)*, pages 127–135.

[113] Tiso, J. (2011). *Designing Cisco Network Service Architectures (ARCH).* Foundation Learning Guide. Cisco Press, 3rd edition.

[114] Tootoonchian, A., Ghobadi, M., and Ganjali, Y. (2010). Opentm: Traffic matrix estimator for openflow networks. In *Proceedings of the 11th International Conference on Passive and Active Measurement*, PAM'10, pages 201–210, Berlin, Heidelberg. Springer-Verlag.

[115] Valiant, L. G. (1982). A scheme for fast parallel communication. *SIAM Journal on Computing*, 11(2):350–361.

[116] Vissicchio, S., Cittadini, L., Bonaventure, O., Xie, G. G., and Vanbever, L. (2015). On the co-existence of distributed and centralized routing control-planes. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 469–477.

[117] Vissicchio, S., Vanbever, L., Cittadini, L., Xie, G. G., and Bonaventure, O. (2017). Safe update of hybrid sdn networks. *IEEE/ACM Transactions on Networking*, 25(3):1649–1662.

[118] Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.

[119] Xu, D., Chiang, M., and Rexford, J. (2007). Deft: Distributed exponentially-weighted flow splitting. In *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pages 71–79.

[120] Xu, D., Chiang, M., and Rexford, J. (2008). Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering. In *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*, pages 466–474.

[121] Xu, H., Huang, H., Chen, S., Zhao, G., and Huang, L. (2018). Achieving high scalability through hybrid switching in software-defined networking. *IEEE/ACM Transactions on Networking*, 26(1):618–632.

[122] Yu, C., Lumezanu, C., Zhang, Y., Singh, V., Jiang, G., and Madhyastha, H. (2013). Flowsense: Monitoring network utilization with zero measurement cost. In *Proceedings of the 14th International Conference on Passive and Active Measurement*, PAM'13, pages 31–41, Berlin, Heidelberg. Springer-Verlag.

[123] Yu, M., Rexford, J., Freedman, M. J., and Wang, J. (2010). Scalable flow-based networking with difane. *SIGCOMM Computer Communication Review*, 41(4):–.

[124] Zhang, Y., Roughan, M., Duffield, N., and Greenberg, A. (2003). Fast accurate computation of large-scale ip traffic matrices from link loads. In *Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '03, pages 206–217, New York, NY, USA. ACM.

[125] Zhang-Shen, R. and McKeown, N. (2008). Designing a fault-tolerant network using valiant load-balancing. In *Proceedings of the 27th Conference on Computer Communications (INFOCOM)*, pages 2360–2368. IEEE.

[126] Zhao, Q., Ge, Z., Wang, J., and Xu, J. (2006). Robust traffic matrix estimation with imperfect information: Making use of multiple data sources. *SIGMETRICS Performance Evaluation Review*, 34(1):133–144.

[127] Zitzler, E., Laumanns, M., and Thiele, L. (2001). Spea2: Improving the strength pareto evolutionary algorithm.

[128] Zitzler, E. and Thiele, L. (1998). An evolutionary algorithm for multiobjective optimization: The strength pareto approach. Technical Report 43.