AS Domain Tunnelling for User-Selectable Loose Source Routing

Habiba Akter

PhD Thesis

School of Electronic Engineering and Computer Science
Queen Mary University of London

2020

Abstract

The use of the Internet as a ubiquitous means of e-commerce, social interaction and entertainment is well established. However, despite service diversity, all traffic is treated the same. Although this clearly "works" and is considered "fair" in terms of net neutrality, there are times when it would be particularly beneficial, if the end-user could have some control over the path his or her traffic takes, either avoiding geographic regions or exploiting lower latency options, should they exist.

In this research work, we propose to design and evaluate a scheme that allows end-users to selectively exploit a sequence of tunnels along a path from the source to a chosen destination. The availability of such tunnels is advertised centrally through a broker, with the cooperation of the Autonomous System (AS) domains, allowing end-users to use them if so desired. The closest analogy this scheme is that of a driver choosing to use one or more toll roads along a route to avoid potential congestion or less desirable geographic locations. It thus takes the form of a type of loose source routing. Furthermore, the approach avoids the need for inter-operator cooperation, although such cooperation provides a means of extending tunnels across AS peers.

In particular, we aim to ascertain the benefit in terms of delay and reliability for a given degree of tunnel presence within a portion of the Internet. The expectation is that a relatively small number of tunnels may be sufficient to provide worthwhile improvements in performance, at least for some users. Based on this premise, we first design and implement a simulation tool that uses Dijkstra's Algorithm to calculate the least cost path(s) for differing percentages of randomly placed intra-AS tunnels. We consider end-to-end delay as the cost metric associated with each route and a number of experiments have been performed to confirm the improvement in delays using the tunnels.

We then consider the inclusion of a small financial cost that the user would be expected to pay in order to use selected tunnels. Details of the payment mechanism is outside the scope of this thesis, however, the financial burden is taken into account when choosing a route. There is thus a trade-off between delay reduction and a financial penalty. First we explore a heuristic approach using a Genetic Algorithm (GA) we create whereby these conflicting goals are combined into a weighted fitness

score associated with the alternative routes, allow a near-optimal compromise to be found, based on the weighting.

The downside of this approach is that there is typically a single solution for a given selected weighting. It may be that the user wishes to see the spectrum of alternatives and decide a suitable "sweet spot" based on their current preferences. As such, we then design, implement and evaluate an end-user path selection tool using Multi-Objective Evolutionary Algorithm (MOEA). Unlike the GA, this approach presents a set of optimal solutions for different compromises between the performance objectives, which form a Pareto front. This scheme currently takes into account cost and delay but provides an extensible mechanism for other fitness factors to be considered.

Acknowledgements

First and foremost, I would like to thank the Almighty for blessing me with the strength and patience to accomplish the research work.

I am immensely thankful to my supervisor, Dr Chris Phillips. Not only he has been a great mentor to me, he was also the friendliest of supervisors any PhD student could ask for. I cannot thank him enough for his guidance and assistance towards improving my academic knowledge and research skill. Whatever I have been able to produce so far would not be possible if I did not have his support by me during every instance ever since the start of my PhD life.

I acknowledge the support and encouragement from my second supervisor, Professor Steve Uhlig. Without his support, it would have been difficult for me to attend a good number of conferences to present my work.

I am grateful to my independent assessor, Dr John Schormans, who has initially inspired me to aim for a PhD in Networks and has helped me with his suggestions for the past few years.

I also thank Dr Raul Mondragon for his valuable time and guideline. In addition, I would also like to thank Dr Damilola Ibosiola for his contribution.

I am thankful to everyone from Networks Research Group, QMUL. All my colleagues and friends have been very supportive.

Finally, I would like to thank my parents, Md Hamidul Islam and Mrs Shirin Akter and my brother, Shihab Shonglap for their support and encouragement on the research journey.

Contents

List of Figures

List of Tables

List of Algorithms

List of Equations

List of Abbreviations

AF	Assured Forwarding
AS	Autonomous System
ASBR	Autonomous System Border Router
ATM	Asynchronous Transfer Mode
BA	Barabasi-Albert
BGP	Border Gateway Protocol
BRITE	Boston university Representative Internet Topology gEnerator
CAGW	Citizens Against Government Waste
DSB	Directory Service Broker
DSCP	DiffServ Code Point
EA	Evolutionary Algorithm
EF	Expedited Forwarding
EP	Evolutionary Programming
EGP	Exterior Gateway Protocol
FCC	Federal Communications Commission
FEC	Forwarding Equivalence Class
FIB	Forwarding Information Base
GA	Genetic Algorithm
GMPLS	Generalised Multi-Protocol Label Switching
GT-ITM	Georgia Tech Internetwork Topology Models
HOT	Heuristically Optimal Topology
IETF	International Engineering Task Force
IGP	Interior Gateway Protocol
Inet	Internet Topology Generator
IP	Internet Protocol
ISP	Internet Service Provider
ITU	Internet Telecommunication Union
L2F	Layer 2 Forwarding
L2TP	Layer 2 Tunnelling Protocol
LER	Label Edge Router

LSP	Label Switched Path
LSR	Label Switched Router
MOEA	Multi-objective Evolutionary Algorithm
MOGA	Multi-objective Genetic Algorithm
MOOP	Multi-Objective Optimisation Problem
MPLS	Multi-Protocol Label Switching
NSGA	Nondominated Sorting Genetic Algorithm
OSPF	Open Shortest Path First
PAES	Pareto Archived Evolution Strategy
PCE	Path Computation Element
PCAT	Path Computation Algorithm for Tunnels
PD-FE	Policy Decision Function Entity
PFP	Positive Feedback preference
PHB	Per-Hop Behaviour
PLOD	Power Law Out-Degree
PLRG	Power Law Random Graph
PoP	Point of Presence
PPP	Point-to-Point Protocol
PPTP	Point-to-Point Tunnelling Protocol
QoE	Quality of Experience
QoS	Quality of Service
RACF	Resource and Admission Control Function
RIP	Routing Information Protocol
RSVP	Resource reSerVation Protocol
SOOP	Single Objective Optimisation Problem
SPEA	Strength Pareto Evolutionary Algorithm
TCP	Transmission Control Protocol
ToS	Type of Service
TRAI	Telecom Regulatory Authority of India
TRC-FE	Transpot Resource Control Function Entity
VEGA	Vector Evaluated Genetic Algorithm
VPN	Virtual Private Network
WAN	Wide Area Network

Chapter 1

Introduction

Although the Internet has been both robust and flexible due to its federated nature, providing delivery of time-critical data that needs traversing multiple Autonomous System (AS) domains is still challenging [3]. This is hampered by the unwillingness of network operators to support inter-operator signalling coupled with the control of the associated forwarding infrastructure. Mechanisms for such signalling exist with functional entities such as the ITU Resource and Admission Control Function (RACF) [4, 5] and the IETF Path Computation Element (PCE) [6]. Although these operator-owned control plane entities have been proposed and refined over many years, their adoption outside of the academic community is no closer [3, 7].

The signalling mechanism to support reliable end-to-end delivery either remains limited to exploiting AS path information provided by the Border Gateway Protocol (BGP) [8] or the construction of overlay networks [9]. Despite much published material on the RACF and PCE concerning how these entities could function across multiple inter-provider domains [4, 7, 10, 11] practical schemes are no nearer adoption.

However, the focus of this research is not concerned with establishing end-to-end paths, or tunnels spanning multiple AS domains. Rather we aim to design a scheme where the tunnels available in the network will be advertised by a "Service Broker" to the end-users, giving the users (i.e. typically via some automated path selection algorithm) the opportunity to select which of those to use, if any, while sending their data from a source to destination address. We assume that at least some operators will be cooperative, letting the broker know about the available tunnels and their characteristics, since we also consider a mechanism of financial recompense. Two different versions of a tool, us-

ing Genetic Algorithm (GA) and Multi-Objective Evolutionary Algorithm (MOEA) are also proposed to be present at the users' access point, which will select the most "appropriate" path for a given data stream and this selection will be made depending on constraints such as the amount of money the user is ready to pay and the end-to-end delay.

1.1 Research Background and Motivation

The motivation for tunnelling over segments or the entire end-to-end path across the Internet is to overcome limitations inherent in the traditional next-hop forwarding mechanism. With next-hop forwarding the path taken by the traffic is determined by the router node at each "hop" point using information held in its Forwarding Information Base (FIB). The FIB data is typically constructed based on automatically configured routing information obtained via intra- and inter-gateway routing protocols along with operator policy filtering [12]. This presents two key issues.

First, the end-user has no say in how their data is forwarded. Here, end-users can be defined as the users of the computer network to send their data over.

Second, the lack of traffic differentiation means that information flows along paths based on a simple "least cost" metric leading to load imbalances and "best effort" equal treatment of all traffic irrespective of its importance to the user.

A signalling mechanism, e.g., a classification and label switching mechanism can be used to address both of these issues. Tunnelling has already been implemented using various technologies [13]. We aim to give end-users some control over choosing the path their data flows by making the presence of these tunnels visible, advertised centrally using a broker, along with a means of steering traffic in sequence between them. Although the broker we have proposed is expected to know where the tunnels are, along with the characteristics, it does not need to know how the tunnels are established or operated. Operator security is not compromised as the details of the technology used to provide the tunnels are hidden, and their establishment and maintenance remain fully under the control of the operator.

Users can choose to use the tunnels, if they wish, for a nominal fee. The idea of charging customers for better service is not new [14]. However, in our case electing to use the tunnels is optional and it is

up to the user which flow(s) are directed through them. As such, some customers may be happy to pay to obtain flow transport with a better Quality of Experience (QoE).

In our research work, we have proposed the end-user as the one to decide whether specific tunnels will be used or not, knowing the "financial cost" and the expected benefits. Operators are expected to cooperate as they receive extra revenue by providing the tunnels. However, these tunnels, only straddle ingress to egress points of specific AS domains between AS Border Routers (ASBRs). The location, delay, cost and perhaps resilience of these tunnels (comprising an IP address of the ingress ASBR and additional information) are passed to the broker. A tool at the end-user's access point can see the information advertised by the broker and offers suitable paths optionally deciding to direct traffic flows via one or more tunnels if the perceived benefits are sufficient relative to the cost involved.

1.2 Claims of Novelty

The work outlined in this thesis is based on a number of existing concepts. The concept of tunnelling, label switching via Multi-Protocol Label Switching (MPLS), IP-in-IP encapsulation, and the use of Internet brokers already exist and are well understood. However, we exploit these technologies in a novel way:

- The first novel aspect of this proposal is the framework consisting of a Directory Service Broker (DSB) that operates with the cooperation of a number of individual network operators that provide end-users with selectable access to one or more AS tunnels between their traffic source and destination. This enables users to make a choice of their paths in order to avoid certain Internet regions or to receive preferential treatment if their packets are to flow through them, assuming a tunnel alternative is available.

- A second novel element is two versions of a path selection algorithm, PCAT (GA and MOEA based) at the end-user that dynamically chooses suitable or fittest paths across the Internet based on two constraints, the need for the traffic to be delivered within a certain time, and financial cost.

- We further explore the hypothesis that providing even a small density of tunnels to the customers will be advantageous in terms of

Quality of Service (QoS) to the extent that customers be prepared to pay for a better service and operators would be motivated to provide such a service. To this end, our third contribution is to assess the magnitude of this benefit as a function of the tunnel density under different traffic loading conditions.

Based on the novelty that we have claimed here, a framework is built to investigate the benefits of using different percentages of tunnels present in a part of the Internet for sending data from one AS to another. The observed results confirm that even for a small number of tunnels, the end user can obtain some benefits depending on the location of the tunnels in the network.

1.3 Overview of the Thesis

Chapter 1 provides an introduction to the research topic. It also presents the research background and motivation for the research, stating the proposed novelty.

Chapter 2 serves as a brief introduction to Internet architecture, the impact of latency and neutrality on the Internet and Loose Source Routing. The chapter also gives a description of tunnelling mechanism explaining the basic concepts of one of the possible ways of implementing tunnels, via Generalised / Multi-Protocol Label Switching (G/MPLS), as well as defining intra- and inter-domain tunnelling and the issues with the latter. It then discusses about some existing internet topology generators, explaining the topology generator tool PFP (Positive Feedback Preference) that we are using for our tool. Next, it describes Dijkstra's algorithm illustrating how it works, which is implemented to provide a baseline route selection algorithm for this research. The chapter ends with the explanation of the Multi-Objective Optimisation Problem (MOOP) since the research employs a MOOP, giving further introduction to Evolutionary Algorithms, one of which is implemented in the developed path computation tool.

Chapter 3 explores the framework that is implemented to reach the goal of this research, including an explanation of the role of the network operator, the directory service broker (DSB) and end-users. It also gives a view of the selection method of the best or least cost

route selection. Finally, the chapter discusses the process of ticketing service for the proposed framework.

Chapter 4 has two main sections. *Firstly,* the section of design and implementation part describes the development of the intial simulation tool that has been used to evaluate the proposed framework. The sections in the chapter gives a brief explanation about how topologies at Autonomous System Border Router (ASBR) level is generated from that of an Autonomous (AS) level which was predeveloped by PFP topology generator tool. Then it introduces the mechanism of generating tunnels and calculating least cost path by implementing Dijkstra's Algorithm. A flowchart and a pseudocode are included to give an easy idea about how the tool is developed. Some useful example outputs of the code used to develop the framework are included in the appendix.

Secondly, It is shown that the framework developed, successfully calculates the least cost path for no tunnels in the network topology and for the presence of different percentages of tunnels. From these least cost paths, the benefits of tunnels present in the internet is observed. Chapter 4 evaluates the efficacy of the tunnels after running several simulations for a number of regional topologies. Average and standard deviation of the benefits for the use of tunnels are calculated and graphical representations are included to show the results. This chapter then ends with a summary of the observed results that eventually confirms the benefit of even a small number of tunnels.

Chapter 5 describes the path computation tool, PCAT–I (Path Computation Algorithm for Tunnels) developed using Genetic algorithm (GA) with a brief introduction about how the MOOP is converted to a Single Optimisation Problem (SOOP) where GA is applied.

The design part shows how a PFP-generated AS-level topology is altered into an input format for the GA, from which the connections of the nodes are obtained to use for the path computation. Then the adaption of GA for the tool is explained in the problem context by describing all the steps included in finding the least cost path. The parameters of the GA are also described here as they vary between different engineering applications.

The PCAT–I is then validated comparing the outputs with manually calculated paths for a small topology. Finally, Chapter 5 includes

the results showing appropriate path computation by the tool. The observation from the results matches the claimed benefit of using Internet tunnels by showing improvement in the average end-to-end delay.

Chapter 6 describes the path computation tool, PCAT–II developed using Strength Pareto Evolutionary Algorithm (SPEA). The main difference in the application of GA and MOEA is described here after stating the requirement of finding path(s) where both the cost metrics i.e., the fitness values for the paths, are considered without converting the problem into a SOOP.

The results presented in Chapter 6 confirms that the PCAT–II calculates one or more optimal paths which are the outputs of the end user software, from which a suitable compromise can be selected.

Chapter 7 concludes the thesis. This chapter discusses the observations from the developed tools explaining how the research goals have been achieved. It also confirms the novel contributions, providing the scope of potential future work.

Chapter 2

Background

The focus of this research is on improving the end-to-end communication performance of the Internet, driven from an end-user perspective. To provide a suitable context for this work the Section 2.1 summarises the Internet architecture and some of its relevant protocols, including the review of several issues in Sections 2.2 and 2.3. Section 2.5 covers the use of tunnelling protocols, referring to recent publications, as appropriate. Section 2.6 discusses about the existing internet topology generator, arguing for the chosen one for performing the required tests for this research work. This chapter also introduces the algorithms which have been designed and implemented to prosper the research framework.

Finally, a summary provides a lead into this research, aimed at addressing the identified shortfalls.

2.1 Internet Architecture

An "internet" is a group of interconnected networks or a collection of rapidly growing network connections [15]. The Internet, on the other hand, is the collection of networks that permits communication between most research institutions, universities, and many other organisations around the world. In addition, it is defined as a global system of interconnected computers networks, which are usually termed as Autonomous Systems (ASes).

2.1.1 Autonomous System

One of the best basic descriptions of an Autonomous System (AS) can be found in the IETF document, RFC 4271 [16]:

"The classic definition of an Autonomous System is a set of routers under a single technical administration, using an interior gateway protocol (IGP) and common metrics to determine how to route packets within the AS, and using an inter-AS routing protocol to determine how to route packets to other ASes. Since this classic definition was developed, it has become common for a single AS to use several IGPs and sometimes several sets of metrics within an AS. The use of the term Autonomous System here stresses the fact that, even when multiple IGPs and metrics are used, the administration of an AS appears to other ASs to have a single coherent interior routing plan and presents a consistent picture of what destinations are reachable through it [16, 17]."

In short, an AS is a connected group of one or more IP prefixes run by one or more network operators which has a *single* and *clearly defined* routing policy. In other words, the routers which are used to move information through one particular group of networks under the same administrative authority and control are known as an AS (Autonomous System). ASes are organised hierarchically into tiers. Tier-1 AS domains provide support for transit traffic over large geographical areas. Tier-2 domains typically provide regional inter-connectivity. Lower tiers then function as access networks. If they have a single point of contact to the rest of the Internet, they are referred to as stub networks or domains.

The routers used for information exchange within a single Autonomous System are called interior routers and the routing protocol used for this purpose is called Interior Gateway Protocol (IGP). In the case of an AS, the route between a pair of nodes is usually selected using distance vector protocols, e.g., Routing Information Protocol (RIP) or link state protocols, e.g., Open Shortest Path First (OSPF) based on detailed intra-AS information. However, when it comes to inter-AS routing, the intra-AS information is filtered before passing to other ASes for security and aministrative reasons. Routers that move information between ASes are called exterior routers and they use an Exterior Gateway Protocol (EGP) such as Border Gateway Protocol (BGP) [18, 19]. The basic routing architecture is shown in Figure 2.1.

The IGPs of an AS are not aware of the internet topology outside their local AS. They still know, through route redistribution, which of their edge routers, i.e., the Autonomous Systems Border Routers (ASBRs), to

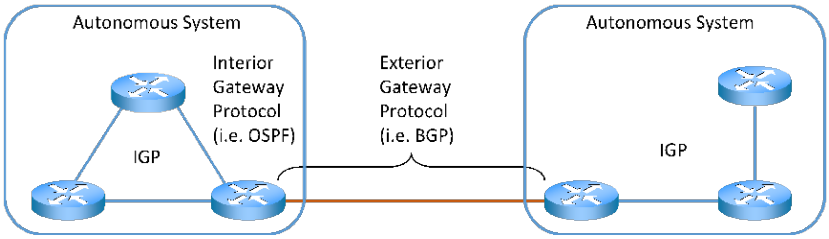


Figure 2.1: Internet architecture

transfer packets to for delivery outside of the local AS. These ASBRs are the routers to provide connectivity to other ASes [12].

2.1.2 Border Gateway Protocol (BGP)

Border Gateway Protocol (BGP) is the de facto protocol for routing across multiple ASes in the Internet [17]. In short, it can be described as "the core routing protocol that makes the Internet work" [20]. Unlike the RIP or OSPF, it is a policy-based path vector protocol that is designed to perform the exchange of routing and reachability information among ASes across the Internet [20].

Every BGP router contains a table to store all the possible reachable destination address prefixes in the Internet and it is more than 600,000 entries long [21]. For BGP, network prefixes, i.e., a block of IP addresses, indicate the reachable networks via alternative paths learned from the neighbours. A BGP router chooses the best path for the data to be sent through to reach the particular prefix or destination IP address based on path length or policy requirements. It also lets the neighbours know about any changes in the path (if there is any) through BGP update messages [20].

BGP sessions can be established to pass information either between neighbours of the same AS or between neighbours of different ASes. The first one is called Internal BGP (IBGP) and the latter is External BGP (EBGP). Neighbours in the same AS use IBGP and neighbours in different ASes use EBGP [22]. Figure 2.2 illustrates this.

Figure 2.2: Internal and external BGP

In Figure 2.2, AS2 is such an example where an IBGP session is implemented to pass the BGP information in between its routers. Moreover, the traversal of the information from a router of AS1 to a router of AS2 is made possible by an EBGP session. It is same for the case in between the routers of AS2 and AS3. A combination of IBGP and EBGP sessions helps the router from AS1 to be able to advertise a route to the router of AS3. IBGP sessions included in the routers enable packets to be forwarded with externally learned destination addresses. BGP -learnt route information is distributed to the interior gateway protocol, such as OSPF.

However, let us consider another example with the AS2 having three routers named as RTR1, RTR2 and RTR3, as shown in Figure 2.3, and a problem of looping might arise if each of the routers makes an entry in the AS path. Hence, to avoid this problem, a router adds its AS number to the AS path only when the route needs to be sent to an EBGP neighbour and avoids doing so for the routes to be sent to an IBGP neighbour. Since BGP sessions are run over the Transmission Control Protocol (TCP), which is a unicast point-to-point protocol, the risk of looping along the transit path between BGP routers is avoided. Moreover, if only EBGP routers (RTR2 and RTR3 in Figure 2.3) are aware of the destination address, the router RTR2 might drop the packet. This can be solved with the presence of IBGP session in all the routers of the AS2.

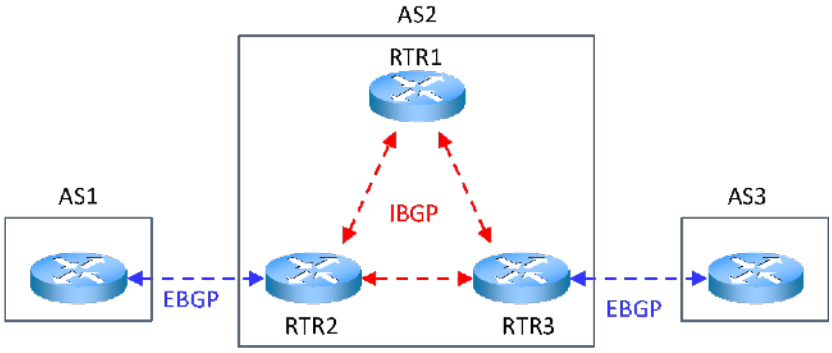


Figure 2.3: Loop prevention

For more reliable routing, a full mesh of IBGP sessions is established amongst all the routers in an AS, ensuring that all of these routers have the information they need to direct the packets to the next hop correctly. After any ASBR of a specific AS gets any information from its neighbour AS, it passes that to all the routers of the same AS. An example fully

meshed IBGP Autonomous System topology is shown in Figure 2.4.

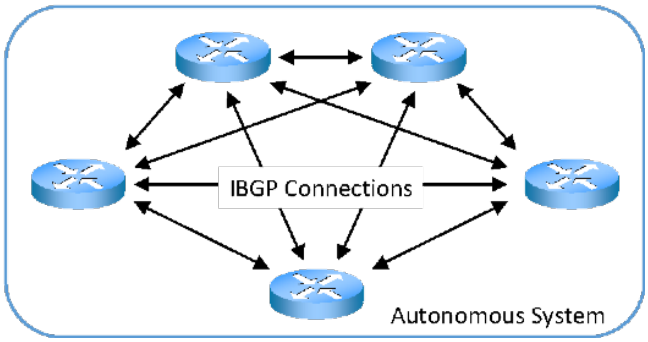


Figure 2.4: A fully meshed IBGP autonomous system topology

2.2 Latency

Implementation of tunnels in the network contributes to reducing network latency. This section discusses the impact of latency in network.

2.2.1 Definition of Latency

In networks, latency is an important constraint. It measures the time taken by some data to reach its destination address from the source. Network latency can be measured as either one-way (this is the time taken for sending data from the source to start to being received at the destination) or round-trip (this is the time taken for any information to get to its destination and back again) [23]. The round-trip delay has an important impact on network services, because a computer using a TCP sends a certain amount of data to its destination and then the next data is sent only after receiving an acknowledgement. In AT&T, latency is measured as the round-trip time of a packet which is sent to the destination and returned to the sender [24].

In simple words, latency of the packets can be defined as "the delay from the time of the start of packet transmission at the sender host to the end of packet transmission at the receiver host [23]."

The following are the main contributors to network latency[25]:

- *Propagation Delay:* It is the time taken by a packet to travel between two places, i.e., from sender to receiver. This type of delay depends

on the distance to be traversed and it is calculated as:

$$D_p = \frac{D}{S} \tag{2.1}$$

where D is the distance from sender to receiver and S is the propagation speed.

- *Transmission Delay:* This is usually the delay introduced by the medium itself, which varies depending on what the medium is. Moreover, the size of packet also has an impact on the round-trip delay since the larger packets need more time. It is calculated as:

$$D_t = \frac{L}{B} \tag{2.2}$$

where L is the number of bits in a packet and B is link capacity (B bits per second).

- *Processing Delay:* This is the time taken by each gateway node to examine the packet.
- *Queuing Delay:*It includes the waiting time for the lines to become available, due to presence of all other traffic that share the same medium in each hop. This type of delay depends on the link capacity, type of the traffic and arrival rate of incoming traffic.
- *Other computer and storage delays:* At the end of a journey in the network, a packet may experience storage or hard disk access delays. However, this delay is not considered as part of the network delay.

Ideally, latency is as close to zero as possible and is typically measured in milliseconds (ms).

2.2.2 Impact of Latency

Internet latency is a focus of attention at the leading edge of the industry. It has been found from several studies [26, 27] that even a slightly increased latency results in a significant reduction in visits to web pages from users and revenue.

In 2009, an experiment was run by Eric Schurman and Jake Brutlag to find the impact of latency on Bing and Google, respectively. With

Bing it was found that a 2 second slowdown changed queries/user by -1.8% and revenue/user by -4.3% [26]. In addition, it was observed that an increase of latency of 100 to 400 ms resulted into 0.2% to 0.6% fall in the number of daily searches via Google [27]. More importantly, even after returning the latency to its previous level, the users still had 0.21% fewer searches which indicated the long-lasting impact of latency on user-behaviour [26, 27]. More case studies of Google were also carried out by Marissa Mayer. One experiment resulted from a rise in the number of search results per page from 10 to 30, with a corresponding increase in page load times from 400 milliseconds to 900 milliseconds. This created a 25% drop in first result page searches. When a shopping cart was included, it resulted into further 2% slower performance that resulted in a 2% decline in searches. However, image optimisations in Google Maps made the page load times 2-3 times faster, with significant increase in user interaction with the site [26].

2.3 Net Neutrality and Quality of Experience

The net neutrality relates to the QoE and the work done for this research is not against the neutrality. This section discusses net neutrality briefly.

2.3.1 Net Neutrality

The term "net neutrality" was first used in 2003, by Tim Wu, as an augmentation of the idea of "common carrier" for telephone systems which transports data for any person or company with taking the responsibility of any possible loss [28].

Net neutrality is the idea stating that "all Internet traffic should be treated equally" [29]. According to this idea, Internet Service Providers (ISPs) and the governments regulating the Internet treat all the data equally, without making any discrimination or taking different charges based on user, content, website, platform, application, type of attached documents, e.g., emails, audio, video, or mode of communication [28, 30]. Hence, according to this policy, the ISPs cannot have the capability of prioritising any data over others while sending it from the source to the expected destination. In 2014, an information security architect, Hagai Bar-El, suggested a technical definition of net neutrality as follows [31]:

> "Network neutrality is the adherence to the paradigm that operation at a certain layer, by a network component (or provider) that is chartered for operating at that layer, is not influenced by interpretation of the processed data at higher layers".

2.3.2 Net Neutrality in Different Countries

Chile became the first country in the world to enact a law in support of net neutrality [32] by doing so on June 13, 2010. The second country in the world and also the first in Europe to do the same was the Netherlands on June 4, 2012. The second country in Europe to support net neutrality by enacting a law was Slovenia. Slovenia legislated a law of electronic communication at the end of 2012. In early 2016, the differential pricing of data services was prohibited in India by Telecom Regulatory Authority of India (TRAI). Nevertheless, violating net neutrality is a very common situation in India.

In Canada, a decision of using usage-based billing system was made by Canadian Radio-Television and Telecommunications Commission on January 25, 2011 [33]. In the UK, PlusNet used "deep packet inspection" in 2007 in order to implement limits and differential charges for peer-to-peer, file transfer protocols, and online game traffic, with a clarification by their network management philosophy that each package they sold was consistent between different websites [34].

The topic of net neutrality is mostly of focus in the USA. There has been a huge debate on whether the net neutrality law is something necessary. It has been claimed by the opponents of this law that investment for the improvement of broadband infrastructure is being deterred by the law [35]. Initially, in February 2015, the Federal Communications Commission (FCC) voted for regulating ISPs more strictly according to the law of net neutrality after receiving millions of comments (between July 2014- September 2014) in favour of changing the internet to a telecommunication service. Hence, FCC reclassified broadband carriers as "common carriers" [36].

However, AT&T and telecom industry did not support it. In April 2017, the then FCC chairman Ajit Varadaraj Pai does not support the net neutrality law and he has also stated that he has the plan to "modernise" FCC policies to "match the reality of the modern marketplace" [37]. His proposal also aims at reclassifying the broadband access as an information service and a loosening in the strict rules on ISPs. According to him, this would help the growth of infrastructural investment and

innovation of the broadband companies [38].

2.3.3 Issues with Net Neutrality

It can be argued that a few milliseconds delay while sending an email will not bother the sender or receiver much. On the other hand, the same amount of delay in a video streaming will leave negative impact on the Quality of Experience (QoE) of the user.

In 2016, the ITU (Internet Telecommunication Union) accepted the following definition, which was stated in 2013

> "The degree of delight or annoyance of the user of an application or service. It results from the fulfilment of his or her expectations with respect to the utility and / or enjoyment of the application or service in the light of the user's personality and current state [39]."

The issues related to net neutrality have made it an extensive research field. The argument against the regulations of net neutrality is supported by economists, ISPs, technologists. Even the operators like Comcast, AT&T, Verizon, IBM, Intel, Nokia, Broadcom, Juniper, QUALCOMM, D-link, Wintel, Alcatel-Lucent, Corning, Panasonic, and Ericsson also oppose net neutrality. In 2006, a website named "Hands Off The Internet" was created by some of the opponents in order to promote the arguments against net neutrality, which was mostly financially supported by AT&T. BellSouth, Alcatel, Cingular and "Citizens Against Government Waste" (CAGW) were included in the member of the site [28].

According to Eric Schimdt, the chairman of Google, from their point of view, although Google treats the similar type of data equally, it does not think that treating different types of data differently is wrong. And this view of Google is something both Google and Verizon generally agree on. The Wall Street Journal stated on February 24, 2015 that Schimdt told a white house official that President Barack Obama was making a wrong decision when he supported the net neutrality rules in the US.

Moreover, net neutrality hinders the investment from broadband companies. This is proved from the statement in a letter sent to FCC leaders from 60 major ISP technology providers that says:

> "instead of billions of broadband investment driving other sectors of the economy forward, any reduction in this spending will stifle growth across the entire economy. This is not

> idle speculation or fear mongering. . . . Title II (common carriers) is going to lead to a slowdown, if not a hold, in broadband build out, because if you don't know that you can recover on your investment, you won't make it [28]."

A motivation behind our research is that certainly from the point of view of an end-user, treating all the traffic in the Internet equally is creating problems. There is a lot of debate going on the topic of whether it is a good idea to give operators the chance to decide about how different traffic should be treated. However, our system is not about charging users for the services, rather it gives the users the opportunity to choose if they want to pay for getting a better service and also provides some control over how their traffic moves across the Internet. Hence, in a way, we are not against net neutrality, rather we are aiming to give more control to the users to decide how they want their traffic to be handled by the Internet.

2.4 Loose Source Routing (LSR)

2.4.1 Definition

Defined in RFC 791 [40], Source Routing is a methodology where the sender of a packet has some control over the route the packet is going to traverse through the network. As discussed before, in the traditional next-hop forwarding mechanism, the next hop is decided by each router that the packet encounters by examining the address of the destination node. In contrast, in source routing, the sender has some opportunity to make some of these decisions based on the information provided by the source [40, 41].

The two types of Source Routing are "Strict Source Routing" and "Loose Source Routing". In Strict Source Routing, each and every hop along the path is stipulated by the source as a sequence of addresses that form the route [40]. On the other hand, in Loose Source Routing, the source can identify a partial route or any number of nodes or intermediate gateways, i.e., some hop-by-hop behaviour mid-path to send its packet to the next address [40] and only the high-level path is specified by the source [41]. Effectively it identifies specific points along a path that the packet must pass through rather than dictate the complete path. This research supports the use of Loose Source Routing.

2.4.2 Use of Loose Source Routing

Although traditional Source Routing is no longer supported in the Internet due to the per packet overhead and security issues [42, 43], a number of research works have considered leveraging the technology of Loose Source Routing. This is due to the evident contribution of LSR in enhancing network flexibility, improving network scalability, and reducing traffic congestion by embracing different routing strategies to balance network traffic.

Source Routing contributes in the application of reverse path calculation and local link failure recovery [42]. IETF has proposed Segment Routing to achieve better traffic engineering and faster rerouting [44]. A noteworthy number of research works have been performed in the field of SDN (Software Defined Network) implementing source routing. AXON [45], SecondNet [46], Rain Man [47] are proposed models in datacenters using the idea of source routing. [45] proposes Ethernet-compatible devices, Axons which use source-routed Ethernet between themselves. This work also determines the advantages of source-routed Ethernet over the traditional switched Ethernet for better network scalability and flexible routing algorithm. The research work in [47] presents the idea of a datacenter network architecture, named "Rain Man" leveraging the benefits of source routing after a clear explanation of the reasons to choose source routing over hop-by-hop routing.

With the use of Loose Source Routing, a public-access site can have some control with filtering incoming traffic which helps during Distributed Denial of Service (DDoS) attacks [48]. [49] has proposed a system named "Binder" for aggregating multiple Internet gateways in a community network implementing finite Loose Source Routing avoiding any security issues. Moreover, [42] has explored the possibility of using this routing mechanism in SDN-based WAN (Wide Area Network) as a substitute for traditional routing as LSR helps improve SDN convergence performance as well as network scalability.

As mentioned in the Introduction, the routing mechanism in our research takes the form of Loose Source Routing where an end user is offered with the power to decide whether or not to use one or more tunnels to send its data over the network. Doing so allows a route to be partially "pinned". This is a separate benefit from any delay reduction packets would experience by traversing tunnels relative to the regular intra-AS sequence of nodes. Furthermore, in case of any network failure or traffic congestion, a tunnel can help a host to detour its traffic around

the hindrance avoiding affected intermediate AS domains if it wishes to.

2.5 Tunnelling Mechanisms

2.5.1 Layer 2 Tunnelling Protocol (L2TP)

Layer two Tunnelling Protocol (L2TP) is an Internet Engineering Task Force (IETF) standard tunnelling protocol. It is an extension of PPTP (Point-to-Point Tunnelling Protocol). L2TP is used to encapsulate Point-to-Point Protocol (PPP) frames for transmission over TCP/IP, X.25, frame relay, or Asynchronous Transfer Mode (ATM) networks [50, 51].

In 1996, L2TP was proposed as an extension of two protocols: PPTP (Point-to-Point Tunnelling Protocol) from Microsoft and, L2F (Layer 2 Forwarding) from Cisco, and combines their best features [52]. Since it is an IETF standard, unlike the PPTP and L2F, it facilitates compatibility between VPN (Virtual Private Network) vendors. So L2TP can be used for creating VPNs over public networks, such as the Internet.

The driving forces behind the development of L2TP include Microsoft and Cisco Systems; L2TP is supported on many Cisco Systems platforms and by Microsoft operating systems [51].

2.5.2 G-MPLS (Generalised) Multi-Protocol Label Switching

In a traditional Internet Protocol (IP) network, routing decisions are made using a longest prefix match. When a packet arrives, the router examines its header; the destination address is compared with information held in a FIB to find the most specific match. The router then sends the packet to the next hop depending on the destination address [12]. This is considered slow and complex. Although mechanisms have been introduced that ameliorate the delays in this look-up process, label switching has since been widely adopted due to its traffic engineering benefits. Conversely, label switching requires a simple lookup of a fixed-length label to determine the forwarding decision. This is typically much quicker. Moreover, when there are two traffic flows for the same destination address, once they intersect, they will be directed along the same path. Using Multi-Protocol Label Switching (MPLS) technology, different flows destined for the same address can be assigned to different paths and retain their uniqueness even if they intersect at a router. The operator has additional control both in terms of which flows are

assigned to particular paths and the sequence of hops each path takes, enabling a greater degree of traffic engineering. Indeed, some paths can be established and used redundantly. Depending on conditions the traffic can be directed along one or other pre-configured paths with little switch-over time.

MPLS defines a way to move data between network nodes by consulting a look-up table at each node using a fixed-length label or tag that is carried in each packet. The main reason for the IETF's interest in MPLS is that the signalling protocol, used to establish, maintain and release these Label Switched Paths (LSPs) is IP based and MPLS can operate alongside traditional IP forwarding [53].

Label Switched Paths

In hop-by-hop routing, the next hop is decided at each router. In MPLS, complete Label Switched Paths (LSPs) are pre-established between particular source-destination pairs. Therefore, unlike the hop-to-hop packet switching, MPLS is regarded as a form of circuit switching [54].

At the edge of the MPLS domain the Label Edge Router (LER) determines which LSP to assign an ingress packet to, if any. This operation is performed by consulting a Forwarding Equivalence Class (FEC) to Label Binding table. A FEC is a collection of analogous features associated with a class of packets which may be forwarded in a similar way [53]. If a match is found, then a label is "pushed" onto the packet and the packet is directed to a specific egress interface. If not, traditional IP forwarding can be invoked.

Assuming a packet is assigned a label, once it is received at the next hop Label Switched Router (LSR) the label is used to determine the next action. This is typically a forwarding operation where the packet label is updated, and the appropriate egress interface is selected. This is label swapping. The process repeats from LSR to LSR until the final LSP hop. In this instance, the label is "popped", i.e. removed and the packet is subsequently forwarded using traditional IP means.

The above process is illustrated in Figure 2.5. In particular, it shows an LSP (i.e. the purple line) carrying data from a host in subnet 47.3.0.0/16 to a host in subnet 47.1.0.0/16 where the MPLS network contains seven LSRs. At each router, there are interfaces and a corresponding label table. Each label number is associated with a particular LSP but its value is otherwise not significant.

A "FEC to Label Binding" table at the Label Edge Router, "A" is con-

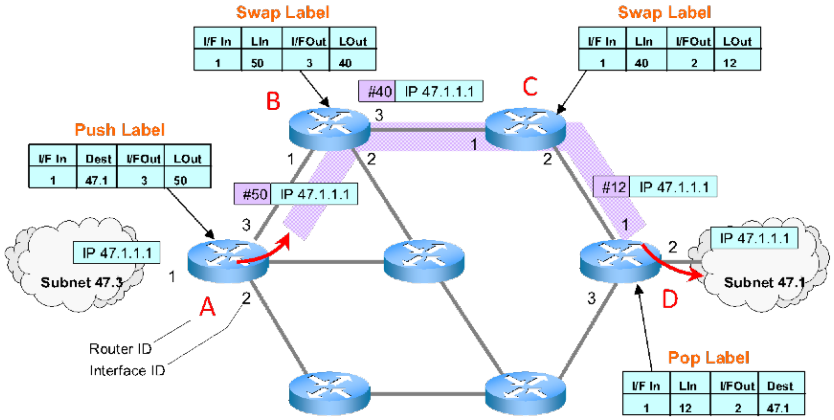


Figure 2.5: Example Label Switched Path (LSP)

sulted when traffic arrives from outside the MPLS domain. The LER selectively determines whether to assign the traffic to an existing LSP and, if so, pushes a label onto the packet(s). Alternatively, it can forward the traffic using traditional IP forwarding. It may also trigger the establishment of a new LSP.

Assuming the FEC to label binding operation pushes a label onto the packets of a particular flow, the traffic is passed along the LSP using the information held in the label swapping tables of the LSRs.

Figure 2.5 shows the swapping information in the LSRs for the "purple" LSP. At each hop the ingress interface and packet label are used to determine the label and egress interface for the next hop. However, at router "D", the popping action is invoked, removing the label from the packet and passing it to an egress interface from where it will be delivered using traditional IP forwarding.

The label swapping information is set up using a signalling protocol. This is typically Resource reSerVation Protocol (RSVP) for Traffic Engineering (RSVP-TE). In the case of the "purple" LSP the resulting entries are as shown in Table 2.1.

Table 2.1: Label data for the "Purple" LSP

LSR	Incoming interface	Incoming label	Outgoing interface	Outgoing label
A	1	-	3	50
B	1	50	3	40
C	1	40	2	12
D	1	12	2	-

Another useful feature of MPLS is the concept of label stacking. This

mechanism is useful since it allows many LSPs to be treated similarly over a truck-route whilst maintaining their individuality at the edges. It is akin to bundling multiple wires into a single conduit for routing between particular ingress and egress points. At either end of the LSPs can be unbundled and treated individually. This greatly enhances the scalability of MPLS.

DiffServ (Differentiated Services)

MPLS provides a mechanism for pre-configuring circuit paths along which particular packet flows are directed. However, it does not control how these packets are treated within a router. Differentiated Services (Diffserv) is a packet classification mechanism that labels each packet with a DiffServ Code Point (DSCP) that determines how it should be treated within a router [12]. Typically, the DSCP reuses part of the Type-of-Service field in the IP version 4 packet header. Its value determines the Per-Hop Behaviour (PHB) as follows:

- Expedited Forwarding (EF) PHB — Dedicated to low-loss, low-latency traffic, where packets are placed in a high priority queue for preferential forwarding.
- Assured Forwarding (AF) PHB — Gives assurance of delivery under prescribed conditions. Typically, packets are placed in a selection of weighted fair queues.
- Default PHB — Used for best-effort traffic. This class is for the lowest priority traffic.

A key benefit of DiffServ is that it can easily co-exist with MPLS. This allows operators to both control the flow path and how the flow is handled at each hop along that path. This provides considerable traffic engineering support. Most operators that use MPLS also use DiffServ, putting the DiffServ code in the header to indicate the priority of the traffic.

2.5.3 Intra-Domain Tunnelling

Intra-domain tunnelling means the use of tunnelling within an AS domain. This approach is widely adopted using MPLS as well as Generalised MPLS (G-MPLS) where the circuit paths are typically optical channels. The principal aim is to enable the operator to operate their

telecommunications infrastructure more effectively by load balancing and providing redundancy. The success of such tunnelling deployment results from the fact that the same operator owns and administers the entire infrastructure concerned, so trust and the use of a common signalling mechanism throughout the network are not issues.

The presence of such tunnels is not usually advertised to anyone outside of the AS domain administration. As such other operators and end-users have no knowledge of whether tunnels are being used, or the form they take, unless the operator wishes to make such knowledge known. For example, in some instances, a "large" end-user might arrange with an operator to link up multiple campus networks at the edge of the operator's domain using leased line tunnel(s). These may have dedicated network resources allocated to them.

In our framework we assume that some operators have chosen to setup a selection of tunnels across their own AS domain infrastructure between AS Border Routers. The form that these tunnels take need not be stated. Some of them may, for example, be implemented as LSPs with DiffServ class-based support. Additionally, some may provide redundancy using 1:N or 1+1 protection backup paths. Unlike the usual traffics ending process, in a 1+1 architecture (spoken of as "One Plus One"),dual copies are sent through two routes in parallel so that, in case of any network failure, the alternate route can be chosen for receiving the packet flow. Indeed, some may even be monitored to provide tunnel ingress to egress performance statistics. In all of these cases, the operator provides this information to the framework Broker.

To ensure only valid traffic is allowed to navigate a particular tunnel, a FEC to label binding (or selective optical channel traffic injection process) is needed. The detail of this process is not considered further at this stage. However, we assume traffic that wishes to traverse a given tunnel is supplied with a ticket, which is used for the authentication and the mapping process.

2.5.4 Inter-Domain Tunnelling

To date, many tunnelling concepts have been proposed. These include proposals for the formation and maintenance of cross-domain tunnels as well as complete end-to-end dedicated paths. However, given the degree of trust involved, the risk of exposing sensitive knowledge and the complexity of managing and funding such tunnels, we regard such schemes as far from commercial adoption. The mechanics of forming

cross-domain tunnels using devices such as Path Computation Elements with control plane signalling have received a reasonable amount of attention. Even so we limit ourselves to the possibility of adjacent AS-domain neighbours occasionally and selectively agreeing to allow tunnelled traffic to extend across a shared AS border. In this case several options are possible although we confine ourselves to just two:

- Traffic exiting a tunnel at one AS border router with an appropriate label, retains the label and uses it directly when it enters an adjacent tunnel in the neighbouring AS domain. Between the AS border routers the traffic uses classic IP forwarding.
- Unlike the first one, when traffic reaches the AS border router of the tunnel portion in the first domain, its label is simply swapped, and it is forwarded on to the AS border router of the second operator, thus allowing a contiguous LSP to exist across adjacent AS domains. How such an LSP is setup is beyond the scope of our work but possible schemes are discussed in [55].

2.6 Internet Topology Generator

Internet topology can be defined as the connectivity graph of a network, i.e., the arrangement of a network's hosts, and the connection of the Autonomous Systems or routers [15, 56]. Having a clear understanding of the internet topology is a basic requirement for many network researches [57].

Considering the importance of having a realistic internet topology for internet research, there have been a reasonable amount of research works on this topic and most of them immerse on the AS-level connection. According to [56], the main reasons for this are:

- AS-level is the highest granularity of the internet and the information of the AS-level connection does not harm the privacy of the operators.
- The information for the other levels are comparatively harder to obtain.
- AS-level topology is engineered by technical or economic constraints.

To benefit the network research, researchers have developed a number of internet topology generator tools.

2.6.1 Importance of Internet Topology Generator

In early 2000s, the difficulty of modelling a precise topology was considered as a crucial reason behind the lack of enough knowledge about Internet simulation since the physical and engineering properties of a network depends on the internet topology [58, 59]. The knowledge of the evolution of the internet topology is a must in order to have an understanding towards the internet architecture and how it interacts with social, economic and technical forces [60]. Moreover, the evaluation of network applications is dependent on how precisely the topology is generated [61]. It also plays important roles in network planning, optimal routing technique and detection of network failures [62].

In short, a precise topology generator renders the chances of designing more methodical protocols, creating accurate models for simulation, and estimating more topological parameters which help to create a real internet-like network.

However, this is not a part of our research, rather we have selected one generator among the existing ones.

2.6.2 Internet Topology Generating Models

A topology is the resultant of the cautious process of network design, applying the design guidelines [63–65]. Some topology models are briefly discussed in this section.

Early Models

The simplest model was the Pure Random model where a set of nodes is distributed in a plane, and an edge is added between each pair of the nodes with a probability p, where the value of p is fixed [56]. Paul Erdős and Alfred Rényi developed the theory of the random graph [66–68]. The first popular model was the Waxman mode [69], where the probability (p) to add edges between nodes is a function of the distance between the nodes. Based on these, Transit-Stub [70] method was developed with an aim to generate a large sparsely-connected internet-like topology. In this method, each node of an initially generated random graph is treated as a transit domain. Each of the transit domains then grow into another connected random graph which represents the backbone of the transit domain. Then a number of random graphs are generated representing the Stub domains attached to the Transit node. Finally, a pair of nodes, one from transit domains and another from the stub

domains or one node from two different stub domains are connected. The GT-ITM (Georgia Tech Internetwork Topology Models) tool is used to generate Transit-Stub networks. [71] proves that Waxman and Transit-Stub model lack the Power law as observed in the real network.

Pure Power Law Models

Since the Faloutsos brothers [72, 73] discovered the power laws, it has been under the limelight of the research on the internet topology. The work is later summarised in [74], discussing the topology at AS-level.If x and y are the measures of interest, then the power law would be:

$$y \propto x^a \tag{2.3}$$

where a is a constant.

The PLOD (Power Law Out-Degree) [75] and PLRG (Power Law Random Graph) [76] are two pure power law models.

Dynamic Growth Models

As mentioned in Section 2.1, internet is growing rapidly. The dynamic growth models generate the internet topology by simulating the internet growth [56]. The model developed by Barabási and Albert [77] mentions two main features of the internet topology:

- Incremental growth: with the continuous introduction of new nodes, the network keeps growing.
- Preferential attachment: while attaching a new node, preference is given to the nodes which are already well connected. It uses power-law link distribution [78]

Topology generator tools are designed for the different topology models.

2.6.3 Existing Topology Generator Tools

The AS-level Internet topology has been modelled for years from the information retrieved from BGP and traceroute data. Later this is proved not to be completely precise as these data sometimes do not have adequate information to form the peering links [60]. One of the main reasons behind this inaccuracy is the lack of the availability and visibility of the information at all levels of the internet [79].

Due to the high importance of a realistic internet topology, there has been a considerable amount of work to design topology generator tools. Hence, there are a number of internet topology generators available till date; e.g., BRITE [80], Inet [81, 82], GT-ITM [70], IGen [63], PFP [83] etc. [58, 84] claim that some of the topology generators like Tier, BRITE, GT-ITM and Inet do not always satisfy the power law distribution or large-scale hierarchy and evolution mechanism.

There are two main perspectives to generate topologies; named as equilibrium (top-down) and non-equilibrium (bottom-up). The equilibrium approach generates a group of graphs which replicates some specific properties and then derives other ones using standard methods. And the non-equilibrium method is about copying the real dynamics of the network and depending on its accuracy, an algorithm generates a network topology of a specified size keeping similarities with the observations. Examples of the non-equilibrium approach are BA Barabási-Albert), HOT (Heuristically Optimal Topology) and PFP (Positive Feedback Preference) [58].

2.6.4 Choice of Internet Topology Generator

The use of inaccurate connections of the ASes in network simulation can result into unrealistic routes or network paths [85, 86]. After doing some careful research, initially we narrowed down our choice in between IGen and PFP. Finally, PFP (Positive Feedback Preference) has been chosen to generate regional AS-level Internet topologies which are then fed into the bespoke tool we have developed.

We know that IGen [61] generates router-level end-to-end topology using heuristics and geographic constraints, where the routers are grouped into Point of Presence (POP) based on their geographical proximity and links are generated based on the network heuristics. Despite of its goal of generating a realistic topology, it does not answer the question of how the divergent ISP topologies should be interconnected. As our research concerns with the inter-AS and intra-AS tunnels, we are interested in the AS-level information only.

As mentioned in Section 2.6, generating AS-level topology is easier and less prone to errors. It is noteworthy that the AS-level internet topology not only undergoes a constant growth of links, but it is also impacted by the changes of the policy routing and hot potato routing at the IP-level [87]. We have chosen to use PFP as it takes care of this fact while generating synthetic AS-level internet topologies.

Moreover, The PFP model has been validated with the AS-level graph derived from traceroute data [83]. This data is claimed to generate more real-internet like graph compared to the measurement data obtained from BGP table [88].

2.6.5 PFP (Positive Feedback Preference)

PFP is a phenomenological model for AS-level Internet topology, which was developed by Raul Mondragon and Shi Zhou in 2004. It can precisely reproduce a number of topological characteristics, e.g., degree distribution, rich club connectivity, maximum degree, shortest path length, short cycles, disassociative mixing and betweenness centrality [83].

PFP [78] uses two mechanisms for topology generation [84]:

- **Interactive Growth:** The PFP tool starts from a small random AS-graph and keeps growing where at each step, new nodes are attached to old nodes and old nodes also peer with other old nodes,
  - with probability $p \in [0, 1]$, a new node is attached to one old node (we call it a *host*), and then the host develops new (internal) links to two other old nodes or peer nodes.
  - with probability $1-p$, a new node is attached to two hosts, one of which is linked to a peer node.

  Numerical simulation shows that when parameter $p = 0.4$, the PFP model can reproduce the best result.

- **Positive Feedback Preference:** It is important to note that the above mentioned mechanism of interactive growth, host nodes and peer nodes are not chosen uniformly and randomly, rather the nodes already having connections will be prioritised. The probability for a new node of choosing a node i with node degree k is:

$$\Pi = \frac{k_i^{1+\delta \ln k}}{\sum_j k_j^{1+\delta \ln k}} \tag{2.4}$$

  Where $\delta \geq 0$.

  According to the developers of the PFP, numerical simulation shows PFP performs better when parameter $\delta = 0 : 0.021$ i.e., $\delta = 0.048$.

In the PFP mechanism, preference is given to the more connected nodes. Thus Positive Feedback Preference means that with acquiring new links, the node will also get the advantage of getting prioritised while more new links are grown – this is a non-linear feedback loop. In 2009, Shi Zhou and other researchers have calculated the probability of a node gaining a new link, which is a function of the node degree as 0.48 [58]. The more links a node has, the more is its chance to obtain further links. The developers of PFP have explained the consequence as, "the rich not only get richer, but they get proportionately richer" [58, 84].

PFP-generated AS-level topology is used as an input of the research purpose.

2.7 Dijkstra's Algorithm

2.7.1 Definition

Dijkstra's algorithm [89, 90] was formulated by computer scientist Edsger W. Dijkstra in 1956, which was published in 1959 [91]. This is an algorithm for finding the shortest path from a starting node to a target node in a weighted graph. More precisely, it finds the least cost path from a selected node to all the other nodes, resulting into a shortest-path tree. It can also be stopped at a certain point to find least cost path from a single node to another specific node. Link state protocols are built around this algorithm. In short, it can be stated that the algorithm constructs (a) tree of minimum total length between the n nodes, where the tree is a graph with one and only one path between every two nodes.

In order to construct the tree, three types of **branches** can be explained as follows:

- Branches, which are definitely assigned to the tree under construction (they will be in a sub tree);
- Branches, from which the next branch to be added to set I, will be selected;
- Branches, which are still rejected or not considered.

In addition, the **nodes** can be divided into two sets as follows:

- Nodes, which are connected by the branches of set I;

- Remaining nodes (one and only one branch of set II will lead to each of these nodes).

Initially, the node that is the "source node", is the only member of set A. Set II is empty and the branches ending at the source node are placed in set II. Then two steps are followed repeatedly to construct the tree:

- *Step 1:* The least cost branch(es) from set II is/are erased from Set II and placed in Set I. This results into transfer of one node from Set B to Set A.
- *Step 2:* Find the branches from the recently transferred node (from Set B to Set A) to the nodes which are still there in Set B. If a corresponding branch is found in Set II which is shorter than the branch under construction, then the shorter branch will replace the longer one in Set II.

After completion of step 2, the process needs to be repeated until there is no branch left in set II and no node left in set B.

2.7.2 Adapting the Algorithm for Routers

While implementing the algorithm to routers, the routers can be divided into three sets:

- *Set I (The Tree Database):* In this set, the links are added to the shortest path tree by adding them here. After finishing the algorithm, the shortest path tree will be obtained here.
- *Set II (The Candidate Database):* Here, the links are copied from Set III, i.e., the link state database to this in a prescribed order, when they become candidates for adding to the tree.
- *Set III (The Link State Database):* All the remaining links are included in this set.

The routers can be placed in two sets as the nodes (as described in Section 2.7.1):

- *Set A* includes the routers, which are connected by the links in the tree database Set I (capitalise).
- *Set B* includes the rest of the routers.

At the end of the algorithm, set B will be vacant.

Router Links can be represented as triples (Router ID, Neighbour ID, Cost). A version of Dijkstra's algorithm adapted for routers [90] can be summarised as follows:

- A router, which is taken as the source node, is added to the tree database as the first router in that, having itself as the neighbour ID as well, resulting into a "0" cost.
- The triples from the link state database, which represents links to the root router's neighbours, i.e., to the routers that have a direct link from the source router, are added to the Candidate database.
- There is cost associated with the links from the source router to the other routers residing in the Candidate database. At this stage, these costs are calculated. Among all the links, the link that has the least cost associated is removed from the candidate list and placed in the tree database. In some cases, more than one link might have equally low cost from the root and any of those can be chosen.
- Now, the Neighbour ID of the link that is just added to the Tree database is examined. With the exception of any triples whose Neighbour ID is already in the Tree database, triples in the link state database describing that router's neighbours are added to the Candidate database.
- If there are still elements in the Candidate database, the process needs to be repeated. The algorithm is continued until there are no more triples in the Candidate database.

Dijkstra's algorithm terminates when there is one Neighbour ID entry in the Tree database to describe every router. At this point, the shortest path tree is complete.

2.7.3 Example of Dijkstra's Algorithm

An example is worked out in this section to explain how Dijkstra's algorithm is implemented. Figure 2.6 represents a network with seven routers, having links among them associated with different link costs. However, not necessarily the incoming and outgoing costs from the same router will be same; e.g., the link from router 2 (R2) to router 3 (R3) has a cost of 2, whereas the link from R3 to R2 has a cost of 5.

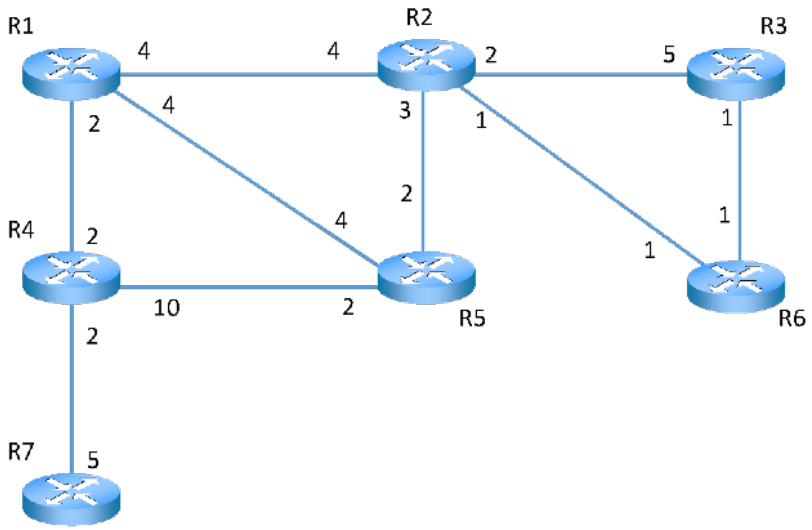


Figure 2.6: Example topology comprising seven routers with different costs

From Figure 2.6, Table 2.2 can be formed having three columns representing the three values of a triple: Router ID, Neighbour ID and Cost.

Table 2.2: List of routers, their neighbours and the cost to them

Router ID	Neighbour ID	Cost
R1	R2	4
R1	R4	2
R1	R5	4
R2	R1	4
R2	R3	2
R2	R5	3
R2	R6	1
R3	R2	5
R3	R6	1
R4	R1	2
R4	R5	10
R4	R7	2
R5	R1	4
R5	R2	2
R5	R4	2
R6	R2	1
R6	R3	1
R7	R4	5

Then, Table 2.3 shows how to get the shortest path from R1 to the other routers step by step by implementing Dijkstra's algorithm. The

highlighted elements of the first column indicates the routers which are removed from the candidate list and added to the tree list. The comments explain the tasks at every step.

Table 2.3: Implementing Dijkstra's Algorithm

Candidate	Total cost to root	Tree	Comments
	0	R1, R1, 0	• Router R1 adds itself to the tree as root.
R1, R2, 4 R1, R4, 2 R1, R5, 4	4 2 4	R1, R1, 0	• The links to all the neighbour routers of R1 is added to the candidate list.
(R1, R2, 4) R1, R5, 4 (R4, R1, 2) R4, R5, 10 R4, R7, 2	4 4 2+2 = 4 2+10 = 12 2 +2 = 4	R1, R1, 0 R1, R4, 2	• There are two triples having the **total** lowest cost of 4. When more than one have the same lowest cost, any of them can be chosen. The triple having the **total** lowest cost (R1, R4, 2) is added to the tree after removing it from the candidate database. • All the neighbours of R4 are added to the candidate list now.

Continuation of Table 2.3: Implementing Dijkstra's Algorithm.

Candidate	Total cost to root	Tree	Comments
(R1, R5, 4)	4		• The triple having the **total** lowest cost of 4, (R1, R4, 2) is added to the tree after removing it from the candidate database. • R4, R1, 2 is removed from the candidate list, since it is showing a route to R1, which is the source route. • All the neighbours of R2 are added to the candidate list now.
(R4, R5, 10)	12		
R4, R7, 2	4	R1, R1, 0	
(R2, R1, 4)	4+4 = 8	R1, R4, 2	
R2, R3, 2	4+2 = 6	R1, R2, 4	
(R2, R5, 3)	4+3 = 7		
R2, R6, 1	4+1 = 5		
(R4, R7, 2)	4		• R1, R5, 4 is the triple with the total lowest cost, which is taken to the tree from the candidate database. • R4, R5, 10 and R2, R5, 3 are removed from candidate database, since already a route to R5 with less total cost has been added to the tree, which is R1, R5, 4. • R2, R1, 4 is removed since it is representing a path to the source router, R1. • All the neighbours of R5 are added to the candidate database.
R2, R3, 2	6	R1, R1, 0	
R2, R6, 1	5	R1, R4, 2	
(R5, R1, 4)	4+4 = 8	R1, R2, 4	
(R5, R2, 2)	4+2 = 6	R1, R5, 4	
(R5, R4, 4)	4+2 = 6		

Continuation of Table 2.3: Implementing Dijkstra's Algorithm.

Candidate	Total cost to root	Tree	Comments
R2, R3, 2 (R2, R6, 1) (R7, R4, 5)	6 5 2+5 = 7	R1, R1, 0 R1, R4, 2 R1, R2, 4 R1, R5, 4 R4, R7, 2 R2, R6, 1	• The lowest total cost is 4 now and the triple having it is R4, R7, 2. Hence, this is removed from the candidate database and added to the tree database. • The triples R5, R1, 4 is remove from the candidate database since it represents a link to the source router, R1. • R5, R2, 2 and R5, R4, 2 are removed from the candidate database because routes to R2 and R4 are already there in the tree database. • All links to the neighbour routers of R7 are added to the candidate list.

Continuation of Table 2.3: Implementing Dijkstra's Algorithm.

Candidate	Total cost to root	Tree	Comments
R2, R3, 2 (R6, R2, 1) (R6, R3, 1)	6 1+1 = 2 1+1 = 2	R1, R1, 0 R1, R4, 2 R1, R2, 4 R1, R5, 4 R4, R7, 2 R2, R6, 1	• The triple having the lowest total cost is R2, R6, 1. This is added to the tree after removing it from the candidate database. • R7, R4, 5 is dropped since there is already a triple in the tree database with the least cost to the router R4. • All links to the neighbour routers of R6 from it are added to the candidate list.
		R1, R1, 0 R1, R4, 2 R1, R2, 4 R1, R5, 4 R4, R7, 2 R2, R6, 1 R2, R3, 2	• Both of R6, R2, 1 and R6, R3, 1 have the same lowest total cost, 2. But already least cost path to R2 and R3 are there in the tree database. So, both of these are simply dropped from the candidate database. • So, the only triple left is R2, R3, 2. This is now added to the database • No more candidates are left in the list. Therefore, this is the end of executing the algorithm. Also, the least cost path is complete.

The final tree database entries of the least cost paths from Router 1 is presented in Table 2.4:

Table 2.4: Final tree for the least cost paths from Router 1

Router ID	Neighbour ID	Cost
R1	R1	0
R1	R4	2
R1	R2	4
R1	R5	4
R4	R7	2
R2	R6	1
R2	R3	2

From this database information, the tree can be drawn as shown in Figure 2.7, which has router R1 as the source router. The paths to all the other routers are shown along with their associated least cost information.

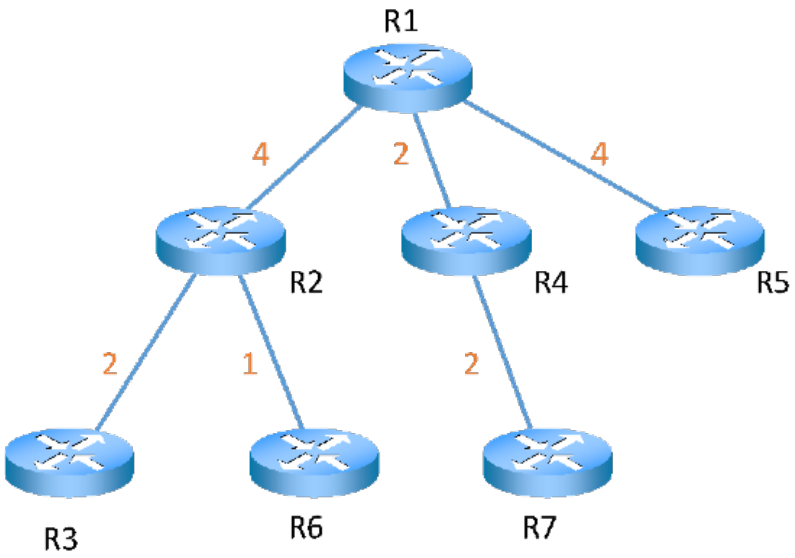


Figure 2.7: Routing tree showing the least cost path from router R1

The Dijkstra's Algorithm works successfully for a single metric. We have implemented the algorithm to develop our first simulation tool to find the least cost path from a source AS to a destination AS. Chapter 4 explains the design and development of the tool.

Later another tool has been developed that can do the same, but considers multiple constraints, which falls in the category of Multi-

Objective Optimisation Problem (MOOP). Section 2.8 gives a brief introduction to this.

2.8 Multi-Objective Optimisation Problem (MOOP)

In most real-world contexts, multiple and conflicting objectives are involved. For example, to make a production line, a number of competing number of factors/criteria need to be considered:

- Maximising the production rate
- Maximising the machine utilisation
- Minimising the throughput time
- Minimising the overall production time
- Minimising the overall cost

This can be defined as a Multi-Objective Optimisation Problem MOOP. Only if all of these objectives can be achieved concurrently, then the problem can be converted into a single objective problem which is very often difficult because of the conflicting nature of the objectives. Hence, we need to find a "compromise" among the objectives. This is required for the tool we are aiming to develop as well. Multi-objective optimisation is a method that optimises two or more objectives that conflict being subject to some constraints simultaneously [92].

Sections 2.8.1, 2.8.2, 2.8.3 and 2.8.4 present the basic concepts of MOOP.

2.8.1 MOOP Problem

A MOOP usually includes a set of decision vectors, a set of objective vectors and a number of constraints [1, 93].

Let us assume, a scenario where MOOP has:

- A set of *decision vectors* or *solutions*, x, which is composed of n number of decision variables $(x_1, x_2, x_3, \ldots, x_n)$, in n-dimensional decision space,X.

  i.e., $x = (x_1, x_2, x_3, ..., x_n) \in X \subseteq R^n$, where R is the set of real numbers.

- A set of m objective functions, $f(x) = (f_1(x), f_2(x), f_3(x), ..., f_m(x))$.

- A set of objective vectors, y that represents m objective variables $(y_1, y_2, y_3, ..., y_m)$ In m-dimensional objective space Y.

  i.e., $y = (y_1, y_2, y_3, ..., y_m) \in Y \subseteq R^m$.

- A set of k constraints $h(x)$.

  i.e., $h = h(x) = (h_1(x), h_2(x), h_3(x), ..., h_k(x))$,

And the aim of the MOOP is to maximise all the objectives.

Comparing two solutions in a Single Objective Optimisation Problem (SOOP) is not complicated. Considering the objective function, whichever solution performs better than the other, is the "optimised" solution, e.g., a solution $x_1 \in X$ is better than another solution $x_2 \in X$ if $y_1 > y_2$ Where $y_1 = f(x_1)$ and $y_2 = f(x_2)$. Even if more than one solution exists in the decision space, all of those will be mapped to the same objective value.

Figure 2.8 illustrates the decision and objective spaces in MOOP.

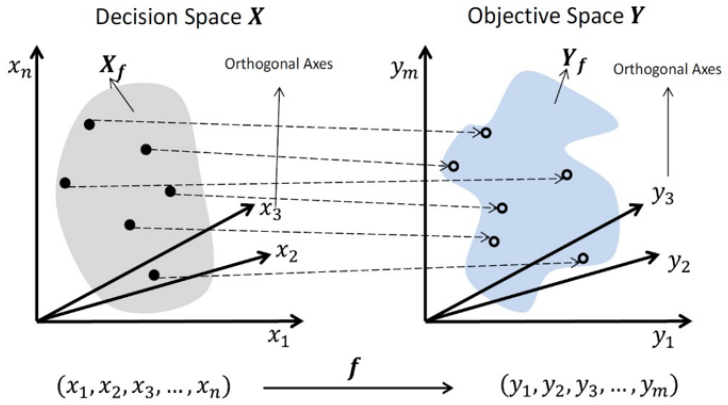


Figure 2.8: Decision and objective spaces in MOOP [1]

This is not the case for comparing in MOOP as the number of objectives to take into account is more. For making such comparisons among different solutions, the concept of Pareto Dominance is introduced, which was originally proposed by an Italian economist, Vilfredo Pareto in his economics studies. The next section gives a brief introduction about Pareto Dominance. The idea of using pareto fitness for solving a MOOP was introduced by Goldberg in 1989 [94, 95].

2.8.2 Pareto Dominance

This subsection explains the dominance for two decision vectors x_1 and x_2 in a **minimisation** problem.

- x_1 is better than the other solution x_2, i.e., x_1 **dominates** x_2 or ($x_1 > x_2$), if all the components of $y_1 = f(x_1)$ **dominates** $y_2 = f(x_2)$. In other words, if all the components of $y_1 = f(x_1)$ are **smaller** than the corresponding ones of $y_2 = f(x_2)$.

  In short, $x_1 > x_2$ (x_1 **dominates** x_2) *iff* $f(x_1) < f(x_2)$.

- Again, x_1 **weakly dominates** x_2 if at least one component of y_1 is **smaller** than y_2 and **not worse** than the other components OR if all the components of y_1 and y_2 are **equal**.

  In short, $x_1 \geq x_2$ (x_1 **weakly dominates** x_2) *iff* $f(x_1) \leq f(x_2)$.

- Solutions x_1 And x_2 are **indifferent** or **incomparable** if neither of them dominates the other and if they are not equal to each other. This will be true if one/more components of y_1 is/are smaller than the corresponding components of y_2 and at the same time one/more components of y_2 is/are smaller than the corresponding components of y_1.

  In short, $x_1 \sim x_2$ (x_1 is **indifferent** or **incomparable** to x_2) *iff* $f(x_1) \nleq f(x_2) \wedge f(x_1) \ngeq f(x_2)$

Figure 2.9 represents an example of the Pareto Dominance.

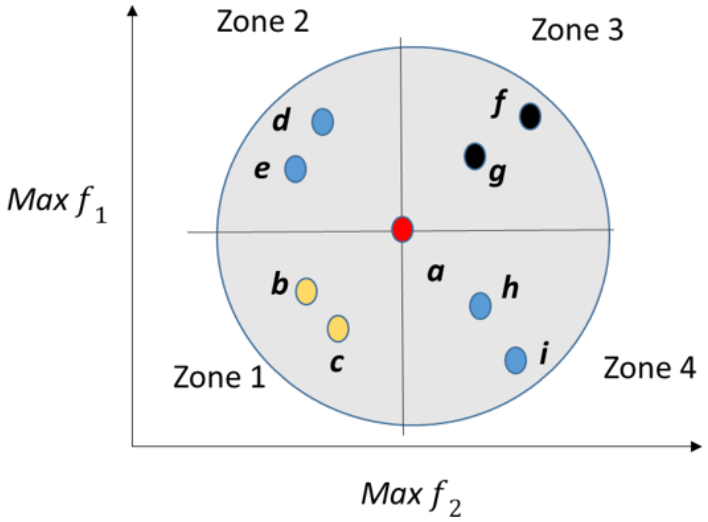


Figure 2.9: Pareto dominance

Let us assume that, two objective functions for the MOOP problem is f_1 and f_2 and both of these need to be **maximised** where objective vectors are ***a, b, c, d, e, f, g, h*** and ***i*** in the objective space. The

feasible set Y_f is represented by the circle and is divided into four zones according to the value of the two objective functions of ***a***.

From Figure 2.9, we can see that the values of the functions of ***b*** and ***c*** in zone 1 are smaller than ***a***. So the decision vector of ***a*** dominates the decision vector of ***b*** and ***c*** in zone 1. In zone 2, for both of the vectors ***d*** and ***e***, one of the objective functions has better value than ***a*** and the other has worse, which means they are indifferent. The same observation is true for zone 4. Now, for the vectors ***f*** and ***g*** in zone 3, it is clearly visible that their values are greater than ***a***. Hence, the corresponding decision vectors of ***f*** and ***g*** dominates the decision vector of a.

2.8.3 Pareto Optimality

Based on the previous explanation, the concept of Pareto optimality can be discussed too. The solutions which are not dominated by any other solution, are also known as the optimal solution. The optimality of the solutions is due to the fact that no solution can be considered as better than any other solution with respect to all the objective functions. In the example explained from Figure 2.9, the solution ***f*** has the ultimate optimality since no other vector has better value for any of the two objective functions involved and they also cannot have a better objective value without compromising the other. This is also known as **non-dominated solution**.

We can say, a decision vector, $x \in X_f$ is non-dominated in a set $S \subseteq X_f$ iff $s \in S : s > x$.

And if the set S is equal to X_f, then x is **Pareto optimal**.

The solutions are known as optimal solutions, which can be mapped to different objective vectors [96, 97]. It is important to note that, a MOOP gets converted to a SOOP if all the functions have the tendency to have better or worse values at the same time [1]. But this is not the case for most of the real cases where the number of optimal objective vectors can be more than one due to the fact that the compromise between the constraints can always be different. In such cases, rather than having a single optimal solution (as shown in the previous example), we will look for a set of trade-off or compromise solutions, which is typically known as a **non-dominated set**.

Figure 2.10 shows the Pareto Optimality in a general MOOP.

The set of optimal solutions in the decision space X is in general denoted as the Pareto set $X* \subseteq X$, and its image can be denoted in

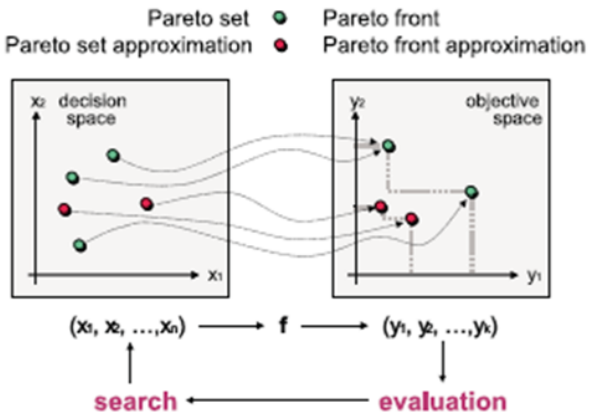


Figure 2.10: Pareto optimality in a general Multi-objective Optimisation Problem [2]

objective space as Pareto front $Y* = f(X*) \subseteq Y$.

2.8.4 Solving MOOP

For many multi-objective optimisation problems, the knowledge of the Pareto set helps solving them. Two main steps are involved in this [1]:

- Searching: Searching finds the Pareto-optimal solutions;
- Decision Making: Decision-making is actually about the decision to choose the solution from the Pareto-optimal set, that makes the best compromise in regard to the functions or constraints.

A number of approaches are there for the two steps [1, 96]:

- **Priori or Before search:** In order to find a Pareto-optimal solution, this method requires some preference information before of the search process.
- **Posterior or After search:** Having multiple objectives defined, the method searches (or approximates) the Pareto-optimal solutions without any preference information. Then, a decision maker chooses a suitable final solution.
- **Interactive or During search:** This method needs to search for the solution with the preference information provided interactively. During each of iteration, the decision maker receives some "trade-offs" for the selection and the search is carried on based on the preference information provided by the decision-maker.

- **Combination of the above:** A fourth approach can be made towards finding the solution that will actually combine the above strategies.

In order to solve MOOP, many conventional techniques have been proposed [92], including the following:

- **Weighted Sum Technique:** In this technique, multiple objectives are converted into single objective using linear combination of objectives.
- **Constraint Based Technique:** From the multiple given objectives, each time this technique takes only one into account i.e., for 'k' number of given objectives it will consider one and treat the rest as k-1 as constraints. The same is followed for all of the objectives step-by-step and the final solution is obtained. Knowledge about the constraints is a must for this technique.
- **Evolutionary Based Techniques:** These techniques are actually based on the idea of Genetic Algorithms and perform well for solving MOOP.

We have used the evolutionary based technique for developing our bespoke tool. Section 2.9 gives a brief idea about the Evolutionary Algorithms.

2.9 Evolutionary Algorithm (EA)

Being originally developed in the late 1950's, EAs can be described as a class of stochastic optimization method that simulates the process of natural evolution [96]. Since 1970s, a number of evolutionary methods have been suggested, the main three are: Genetic Algorithm (GA), Evolutionary Strategy (ES) and Evolutionary Programming (EP) [98]. All of these are based on biology-inspired mechanisms like mutation, crossover, and natural selection to refine a set of candidate solutions [1, 99]. Despite of their similarities in general, the algorithms are different from one another [100]. Genetic Algorithms were developed by Holland [101], and thoroughly reviewed by Goldberg [94]. Evolutionary Strategies were developed by Rechenberg [102] and Schwefel [103]; and Evolutionary Programming was developed by L.J. Fogel and D.B. Fogel [104].

2.9.1 Similarities and Differences

GA, ES and EP- all of these three operate on fixed length strings, which contain real values in ESs and EPs and binary numbers in the canonical GAs [98, 105–107]. Based on a set of initial population and search point, all the three algorithms incorporate a mutation operator, which is the mainspring for ESs and EPs. However, GAs and ESs also use a recombination operator, which is the primary operator for the GA [108]. They also use a selection operator which applies evolutionary pressure, either "instinctive" (in ESs and EPs, the operator determines which individuals will be excluded from the new population) or "preservative" (in the GA the operator selects individuals for breeding).

In GAs and EP selection is probabilistic. In contrast, ESs use a deterministic selection. ESs and meta-EP allow self-adaptation or self-learning mechanism [108], where parameters controlling mutation are allowed to evolve along with object variables. Finally, it is worth noting that the implementer is free to modify these algorithms. For example, the GA can be run using an integer alphabet.

2.9.2 Flowchart for EA

Figure 2.11 gives a preview of steps involved in a EA.

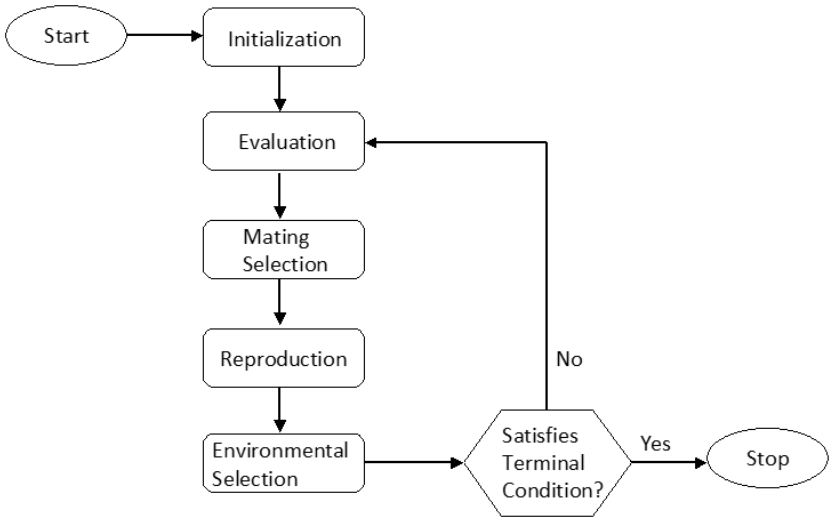


Figure 2.11: Evolutionary Algorithm flowchart

At the beginning of the evolutionary algorithm, in the "Initialisation" stage, the population can be generated by either a random or pre-defined method. At the "Evaluation" stage, evaluation of the candi-

date solutions of the population is done using a fitness function. This scores their merit and removes weaker solutions. Then comes the "Mating Selection" where a selection of the candidates with superior fitness is placed in the mating pool. This phase is stochastic, i.e., the fitter candidates have more possibility to become parents by being placed in the mating pool. The least fit ones will have been eliminated. After this, in "Reproduction", the candidate solutions, which are selected from the mating pool, produce offspring using crossover and mutation operators and this offspring form a new population. This population's quality is comparatively better since it is produced from the fitter candidates. At the "Environment Selection" phase, the new population is assessed in order to determine which candidates will have the opportunity to mate and produce offspring for the next generation.

The algorithm completes when the terminal condition is satisfied or a solution with enough quality is found. Otherwise, the evaluation-selection-reproduction process keeps repeating. Two main mechanisms, Exploration and Exploitation are beneficial for EAs. Exploration can be defined as the capability of creating population diversity by exploring new solutions in the search space. On the other hand, Exploitation decreases the population diversity by focusing on the fitter candidate solutions [109].

2.9.3 Key Components of EA

The key components of an Evolutionary Algorithm (EA) are explained in this section.

- **Individual Representation:** In the evolutionary algorithms, the initial task is about relating the actual problem scenario with the problem-solving space where evolution is supposed to take place. This relation or link is typically obtained by designing an individual representation. In an EA, an individual solution is the candidate solution and it can also be referred as "chromosome". A placeholder in an individual is called a variable, a locus (plural: loci) or gene and its value is addressed as a value or an allele. The possible solutions in the original problem context are referred to as phenotypes. Mappings of phenotypes onto individuals in the EAs, are called genotypes. The method of designing a mapping between phenotypes and genotypes is called representation. Individual representation is problem specific and it can be encoded in binary, float or integer.

- **Evaluation Function or Fitness Function:** The evaluation function or fitness function may differ based on the context of the problem and this is used for measuring the quality of the chromosomes. It is noteworthy that the idea of an objective function in optimisation problems is different from that of a fitness function. However, if the problem needs maximisation, then the objective function and the evaluation function can be the same since fitness is usually associated with maximisation. On the contrary, if the context requires minimisation, it needs a fitness assignment process for the transformation between the objective function and the fitness function.

- **Mating Selection:** Mating selection, or parent selection, focuses on selecting fitter individuals from the current generation and send them to a structure called mating pool to be the parents for reproduction. The mating selection is typically probabilistic. In this selection process, a fitter individual has a higher chance to be selected than a less fit one. This helps ameliorating the quality of the population [110, 111]. An individual with poor quality still has a small chance of being selected to maintain diversity in the population to allow exploration. Two basic selection mechanisms are: Proportionate and Ordinal-based selection [94, 112–114]. The proportionate one selects the fitter chromosomes. Examples of some schemes are roulette wheel selection, stochastic remainder and stochastic universal selection. In ordinal based selection, the selection pressure depends on the rank of the population which is not related to the individual fitness. Example schemes of ordinal-based selection are tournament selection, selection, truncation selection, and linear ranking selection. Commonly used mating selection schemes are proportional selection (roulette wheel selection, tournament selection [115], and rank-based selection.

- **Reproduction Operator:** The selected parent chromosomes undergo the reproduction to generate new individuals. Reproduction consists of two operators: crossover and mutation (recombination). The crossover operator produces offspring by exchanging gene information from two parents, which allows the best part of the selected two parents to combine in order to generate offspring. The selection of the best parts is usually done with a number of random choices. Crossover is stochastic in regard to what parts of the parents' chromosomes are combined. The mutation operator

helps to maintain the population's genetic diversity by amending some gene values randomly. Similar to the crossover operator, the mutation operator is stochastic. The frequency with which these two operators are invoked is controlled by crossover and mutation rates.

- **Environmental Selection or Survivor Selection:** Environmental selection or the survivor selection mechanism is used to select individuals based on their quality (fitness) which is similar to mating selection. As the size of a population is generally a constant, environmental selection is used to decide which individuals can survive through to the next generation depending on their fitness. Environmental selection is often deterministic. For instance, the current population and its offspring are merged, and the top segment is selected according to their fitness.

2.9.4 Advantages

The main reasons for the popularity of Evolutionary Algorithms are as follows [116]:

- The EAs do not need any derivative information.
- They are relatively simple to implement
- They have also been proved as robust and powerful well-functioning methods to solve problems which involve:
  - Multiple conflicting objectives;
  - Unmanageably large and highly complex search spaces [94, 96].
- Moreover, the use of population in EAs (explained Sections 2.9.2 and 2.9.3) also has a number of advantages [117]:
  - The EA procedure can function with a parallel processing power.
  - Because of the populations, the Evolutionary Algorithm can find multiple optimal solutions, which eventually facilitates the solution of multi-modal and multi-objective optimisation problems.
  - Finally, since there will be a number of evolving populations generated, hence any Evolutionary Algorithm will get the opportunity of normalising decision vectors (as well as objec-

tive and constraint functions) within those by using best minimum and maximum values in the population.

For our research purpose we have used the Genetic Algorithm.

2.9.5 Genetic Algorithm (GA)

The concept of Genetic Algorithm was originally developed by John Holland and his students at the University of Michigan in the 1960 [93, 99] which later became popular specially after his book Adaptation in Natural and Artificial Systems [101] was published in 1975 [99]. It was then thoroughly reviewed in 1980s [118, 119] and extended in 1989 by his student David Goldberg [94].

The GA has the same key components as described in the subsection 2.9.3. For multiple constrains, GA finds a solution based on a fitness score which is eventually a combination of the fitness values associated with the objectives.

The application of GA in computer network is very popular. [114, 120–129]

To make sure that we can get a number of possible set of solutions, we have then implemented Multi-Objective Evolutionary Algorithm (MOEA).The Multi Objective Evolutionary Algorithm applies the EA technique and is widely accepted and used as a solution towards MOOP problems. The next section discusses MOEA.

2.10 Multi-Objective Evolutionary Algorithm (MOEA)

During the World Congress of Computational Intelligence (WCCI) in Vancouver 2006, Multi-objective Optimization (MOOP) has been evaluated as one of the three fastest growing fields of research and application among all computational intelligence topics [116]. Multi-objective Evolutionary Algorithm (MOEA) is the most popular approach to solve MOOP [130–132]. It employs evolutionary methodologies to solve problems involving multiple conflicting objectives. MOEAs have been applied in many real-life multi-objective problems in different areas, such as economics and finance [133] and engineering [2, 134–141], sensor networks[142]. Many works have proposed application of MOEA in network [143–146].

The MOEAs aims at finding an approximate Pareto-optimal set in a single simulation run. However, it is not simple to get all Pareto-optimal solutions due to the fact that it is computationally expensive and sometimes it is even infeasible. For example, if the curve of Pareto-optimal front is continuous, the number of solutions is infinite. Hence, it can be claimed that the more realistic target for MOEA will be to find an approximate Pareto-optimal set that satisfies two sub-objectives:

- To minimise the distance from resulting solutions to the pareto-optimal front.
- The solutions should be uniformly distributed assuring the maximum diversity of the pareto front.

Hence, the goal itself is multi-objective. In order to realise the above objectives, some important issues need to be considered such as *fitness assignment, diversity preservation and elitism* [96]:

- **Fitness Assignment:** Compared with SOOPs, where the objective function is sometimes identical to the fitness function, MOOPs need a fitness assignment process after objective values have been calculated for each individual. There are many strategies that can be used for fitness assignment, such as aggregation-based (e.g., weighted sum), criterion-based (e.g., Vector Evaluated Genetic Algorithm (VEGA) [147]) and Pareto-based or dominance-based (e.g., Strength Pareto Evolutionary Algorithm (SPEA).
- **Diversity:** The diversity preservation issue is to ensure that the resulting solutions have a good distribution. This is usually achieved by incorporating density information in the selection process. For instance, an individual has less chance of being selected if it is a short distance from its neighbours.
- **Elitism:** Elitism aims to avoid losing good solutions during the optimization procedure due to some random effects. There are two common strategies to realize the elitism [148]: One is to merge the current population and its offspring after reproduction into a "temporary" population and then rank the individuals. The individuals in the top segment of the temporary population survive and become the next generation. The other is to establish a special population called the "archive" to hold the promising individuals. The archive is separated from the optimisation engine and is updated at each generation when reproduction completes.

2.10.1 Advantages

The main advantages of MOEA that make it more popular to solve MOOP are:

- They are easy to implement.
- They can return more than one solutions.
- There is less chance of the algorithms becoming deadlocked in local minima.
- The algorithms are flexible and robust.
- They do not require a prior knowledge of the problem (unlike the conventional techniques).

Moreover, [149, 150] claim that MOEAs can probably obtain (or approximate) Pareto-optimal solutions in a single optimisation run instead of obtaining one solution each run. In addition, MOEAs usually do not require weight, scale or prioritised objectives [149]. Therefore, MOEAs are one of the most powerful mechanisms to solve the multi-objective problems (often with conflicting goals) and have been successful applied to a wide range of practical problems [151].

2.10.2 Different Types of MOEA

Since the 1980s, there have been several approaches in the development of MOEA [152]. In the initial strategies, a MOOP was converted into a SOOP using an evolutionary methodology. These algorithms do not incorporate the concept of Pareto optimality. Later, in the mid-1980s, some algorithms started introducing Pareto optimality into the evolutionary algorithms. These algorithms include the Nondominated Sorting Genetic Algorithm (NSGA) [153], the Niched-Pareto Genetic Algorithm (NPGA) [154] and the Multi-Objective Genetic Algorithm (MOGA) [149, 155]. [95] gives a summary of different approaches in MOEA to solve MOOPs.

When elitism became a de facto standard mechanism in the late 1990s, the new stage of evolution MOEAs began. The landmark algorithm in the field is generally considered to be the Strength Pareto Evolutionary Algorithm (SPEA), which introduces an external population called the “elite” archive to retain suitable nondominated solutions. This is an important feature as it guarantees that the final solutions are

nondominated with respect to all other solutions across the total evolutionary process rather than the current population. After SPEA, many algorithms incorporating additional archive mechanisms have been proposed. The most representative of them are: Strength Pareto Evolutionary Algorithm 2 (SPEA2) [148], Pareto Archived Evolution Strategy (PAES) [156], and Nondominated Sorting Genetic Algorithm II (NSGA-)-II [157–159].MOEAs, such as NSGA-II and SPEA2 [148], have proven to be efficient for complex MOOPs with two or three objectives [160]. Most elitist MOEAs make use of a combination of dominance and density to make choice of the individuals that will be kept in the archive at every generation [2].

The idea of using MOEAs for routing in the network is not new. Some example works are, [161–168].

From different types of MOEAs, the Strength Pareto Evolutionary Algorithm (SPEA) has been used for the tool proposed in this work. SPEA was proposed by Zitzler and Thiele [169] based on a case study they conducted [170]. Depending on some surveys and comparisons done, [2] claims that SPEA has a positive approach towards solving MOOP problems and has been used in a number of applications too [171].

Chapter 6 explains the detailed implementation of SPEA.

2.11 Summary

This chapter gives an overall literature review, including the basic internet architecture and the discusses the motivations behind this research. Despite of the tunnelling mechanism not being a part of out research, it has been discussed in this chapter in order to give the idea of its protocols as well their possible implementation within one or more domains. After discussing the basic topics, the required tools and algorithms are discussed with an explanation of the choices made for developing the baseline framework and the path computation algorithm for tunnels.

Chapter 3

Overall Tunnelling Framework

The basic architecture of the AS-Domain tunnelling framework is shown in Figure 3.1. The tunnels shown are assumed to have been setup and maintained by the specific network operators using whatever means they wish. This could involve the use of PCE/RACF [4, 6] signalling; however, this is not essential and is beyond the scope of this research. The presence of the tunnels is advertised via the Directory Service Broker (DSB). The tunnels can be of any technology, though it is expected that many will be MPLS or based on optical channels. These tunnels can be both intra and inter AS in scope, in the latter case, this being achieved through operator peering. Some tunnels may offer 1+1 protection; others may exist between peering operators through LSP stitching. However, details of the construction mechanism are outside the scope of this research.

Initially customers for this service are assumed to be Small and Medium Enterprises including financial institutions that wish to transport data quickly without having to incur the costs associated with a leased end-to-end infrastructure. They will have awareness of the sequence of AS domains that their data is passing through and possible alternatives, particularly if BGP reachability information is made available to them via the DSB Internet Map.

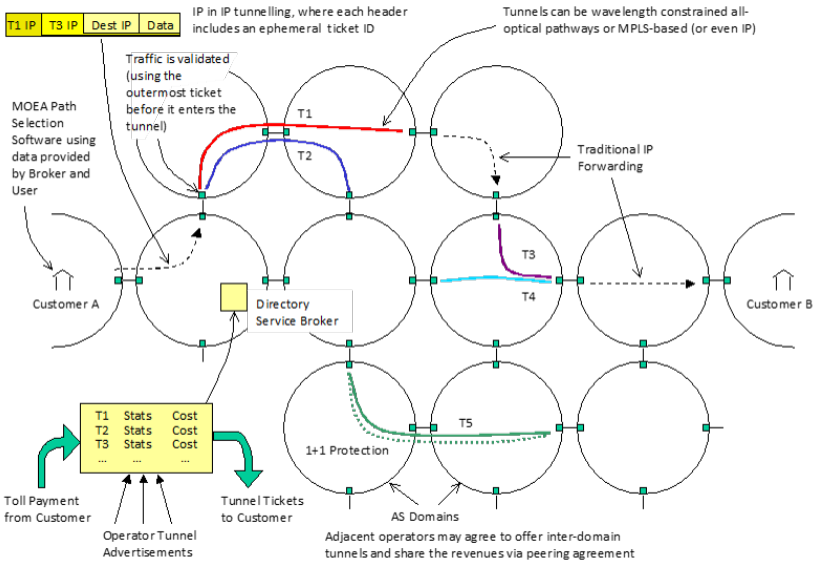


Figure 3.1: User-selectable AS-domain tunnelling framework

Their IT administration, which could be automated software that performs path selection based on cost and other requirements, may wish to choose a preferred path between their own site and a given destination, such as between Customer A and B in Figure 3.1. For example, by interrogating the information in the DSB, customer A wishes to use Tunnel T1 and T3 to hasten the delivery of data between the two sites, possibly avoiding a congested domain. Having informed the DSB of this decision, for a small fee Customer A is given tickets for each of the tunnels (i.e. T1 and T3) along with their ingress IP addresses. Tickets are ephemeral so it is unlikely that users can abuse the system extensively.

Software at the end user's terminal will be developed implementing an algorithm which will perform path selection by comparing alternatives given various constraints. For now, these are the delay associated in each link and the amount of financial cost that the user needs to pay for using tunnels. A customer's desire to avoid a certain AS can be a constraint too. In addition, the tunnels will be chosen to send the traffic only if the benefit exceeds the normal path. To have a clearer idea, let us assume that in Figure 3.1, the average delay experienced for using tunnels T1, T2, T3 and T4 are x, x, x, and 2x (in milliseconds) and the amount of money the user needs to pay for using each of them is 50 units, 25 units, 50 units and 20 units. Now, the option is to pass through tunnels T1 and T3 or T2 and T4. By paying 70 units, the customer can send its traffic via tunnels T2 and T4, where the total of

average delay experienced in the tunnels will be 3x and after making a payment of total 75 units, the total of average delay experienced in the tunnels will reduce to 2x. Now, if customer A provides the information to the software that it should not exceed 80 units of cost but wants as little delay as possible, then the software will choose T1 and T3 based on a simple comparison. However, the decision of making a choice depending on different constraints is typically not going to be as simple as this example.

3.1 Network Operator Functions

It is already mentioned in Chapter 1 that tunnels traversing multiple domains is hampered by the unwillingness of network operators to support inter-operator signalling coupled with the control of the associated forwarding infrastructure. However, our system does not depend on any information that the network operators will not share with the Directory Service Broker (DSB) due to "trust" issue. We assume that the cooperative operators will let the broker advertise the available tunnels centrally with some information such as where the tunnels are, how much the usage charge is and what performance they offer to the users. It does not include any sensitive information such as what tunnelling mechanism is being used by the operators to establish the tunnels or how they are being operated in the ASes. AT&T and CAIDA already provide some information concerning dynamic network performance [172–174].

For now, our system deals with the tunnels existing between Autonomous System Border Routers (ASBRs) belonging to the same AS. There is no need to know about the internal path of the tunnels. Furthermore, tunnels straddling AS domains are considered optional as they would require a peering relationship between operators. However, mechanisms for stitching together LSPs across AS domains could technically be provided if sufficient trust existed between adjacent operators.

What is required, is for the operators to cooperate with the broker in order to share information concerning the location of the tunnel(s) in terms of entrance and exit points, their operating performance (such as average delay, whether they are protected etc.) and the financial cost of using them.

3.2 Broker Function

The concept of a service broker is proposed here in order to provide a centralised resource for advertising the AS tunnels to the end-users giving them the opportunity to choose, to some extent, their desired path across the inter-network.

According to Gartner, brokerage means "any type of intermediation that adds value to the consumer's use of a service" and broker signifies "a person, company or piece of technology that delivers an instance of brokerage or, the specific application of a mechanism that performs the intermediation among consumers and providers" [175]. The research work in [176] states that there is no real-time demonstration of a brokerage system that deals with the practical technological challenges.

In our research, we have proposed a Directory Service Broker (DSB) assuming it has the map view of the ASes which helps to allow the broker to show the end users about which ASes are adjacent to each other and, in case of the cooperative ASes, information concerning their tunnels can also be made available. The broker presents the location of tunnels to the users superimposed on an AS view of the Internet (or a portion of it) and the users will have the opportunity to choose whether their traffic is directed through one or more tunnels in a particular sequence. This provides a form of loose source routing. Furthermore, certain ASes may show information concerning their degree of congestion. This allows the end users to selectively choose to use a tunnel to detour traffic away from the congestion, or to provide preferential treatment across the congested AS.

[177] proposes the idea of implementing a service broker in the field of telecommunication to make an offer of the best service against a customer's request. In [178], a tunnel broker system that minimises the job of the tunnel server by assigning the broker server that handles the user-requests and returns the prime configuration to both users and tunnel servers is discussed. However, this paper indicates that the tunnel server cannot offer the best service and also the user cannot switch its chosen path in case of failure.

In 2008, a scheme to configure AS paths was proposed as a solution of providing inter-domain IP/ (G) MPLS tunnels [13]. However, the proposed PC (Path Computation) mechanism does not handle the selection of the AS-paths. Conversely, in our proposal the end-user employs a path computation algorithm that aims to compute the most appropriate loose source-routed path across the network given various constraints.

This algorithm could operate continuously to allow path alterations to be made in response to changing circumstances.

However, according to [17], information passed by the Border Gateway Protocol (BGP) to the SB may not be enough to allow the end-user to make meaningful path choices. In [179] and [180] enhancements to BGP are proposed to advertise TE information and in [181] an overlay architecture and BGP extensions have been proposed for this. Even so, this is necessary for our research. At this initial stage we have decided on "traceroute" as an alternate means of assessing the status of various AS domains.

We naturally assume that operators that are willing to cooperate, will also pass some information saying whether the tunnels or their default forwarding environment are busy at a particular time and this information can be made available in the proposed broker's map view. In short, the Directory Service Broker (DSB) provides an Internet Map showing the tunnel locations, their usage charge and some statistics regarding the performance they offer.

However, the broker does not tell the user how to get across the network. It provides a view of the topology, with the cost. Even there can be AS present in the network which is not cooperating with the broker. Therefore, the broker does not necessarily have complete knowledge of the network and it certainly does not know anything about what is going on in the ASes. However, it does provide AS network topology information along with a measure of how busy they are. Moreover, we have assumed that the route between the ASBRs at the connection point between a pair of ASes is one-to-one. This is not always the case in the real Internet; rather it can be one-to-many.

The DSB also provides a single brokerage point whereby the user can request a sequence of tunnel permits (tickets) so that traffic can use a tandem arrangement of multiple tunnels between a source and a destination. The DSB is effectively the customer-facing entity where operators advertise their tunnels and transactions can be made.

3.3 End-User Functions

The user will have to install the software in its terminal. The software will obtain the visualization part of the Internet map from the DSB. It also needs to acquire information as follows:

- Number of tunnels in the network.

- Location of the tunnels.
- The amount of money the end-user needs to pay to use each given tunnel, if it wishes to do so.

The software will get some information from the user, e.g.,

- The source and destination ASes for the data to be sent.
- Expected service required by the user, where delay and other constraints will be taken into account.
- Amount of money the user is willing to pay

Knowing the preferences of the user, the software will be able to:

- Determine the least cost path for:
  - tunnel-paths, i.e., when tunnels are employed.
  - no-tunnel paths, i.e., when there is no tunnel.

  However, the tunnels present in the network topology may not always be used if their use does not contribute to a least cost path
- Compare the constraints.
- Suggest the better route(s) for the traffic.

Finally, the decision is made depending on the two factors:

- End-to-end delay and
- Financial cost

The end user then can select the path it wants its data to traverse through. Thus the overall framework we have designed, allows the user to employ a form of Loose Source Routing (LSR).

3.4 Best Route Selection

The user software obtains an AS domain performance map from the broker. It is not necessary to show the complete AS topology and ASBR topology to the user, but it is up to the user whether it wants to have knowledge about these. However, we assume that most users will be interested in obtaining the expected service provided limited by the financial cost it is willing to pay. In case the user wants to see the map,

it is not impossible for the software to do so. Moreover, there is no concern regarding trust or privacy since no internal details associated with this information are made available.

If the user is not interested in using tunnels, the expected least cost path is calculated with the Dijkstra's Algorithm without the use of tunnels and the traffic will be sent through this. The algorithm can also calculate the least cost path using the tunnels present in the network. A single constraint, delay, is considered as the cost in this case. It should be borne in mind that there is no guarantee that packets will travel this path (due to BGP policy information), though traceroute can attempt to determine the standard path and the per-hop delay along the route.

Chapter 4 shows the implementation of Dijkstra's Algorithm in the tool we have developed.

However, a comparison of multiple constraints has been made through the implementation of Genetic Algorithm (GA) and a Multi-Objective Evolutionary Algorithm (MOEA). We do not claim any novelty for the concept of this form of evolutionary algorithm, but the use of MOEA for least cost path calculation within the context of this application, where tunnels are implemented, is novel, along with the various constraints considered. Chapters 5 and 6 describe these implementations in the developed path computation tool, PCAT.

3.5 Ticketing Service

The information about the location and associated delay and financial cost of the tunnels are required to reach and traverse the tunnels by the data flow, if the user wishes so. One possible way to allow data to flow appropriately is to use IP-in-IP packet tunnelling. In our example (from Figure 3.1), this means the first IP packet destination would be the ingress point of tunnel T1. An option field would hold the ticket. Tunnel policing requires access functionality at Layer-3 to validate the ticket and, if necessary, count the number of packets transferred associated with a particular ticket number. If valid, this header is removed and the next IP header takes the packet through the tunnel and all the way to the ingress of tunnel T3. This header is also interrogated and removed allowing the packets to move along tunnel T3 and from there be delivered to the destination (Customer B) using traditional IP means.

Hence, somewhere in the header of the packet, some kind of authentication is required to make sure that the user has already paid to use

the specific tunnel. Now, in IPv4, the header is typically 20 bytes and it can be extended but not all operators will allow it to do so. In IPv6, the main (base) header is always of the same size, but additional headers can be added after that, which are called extension headers. Therefore, in our system, we can keep the required information of authentication in an extension header. According to the way IPv6 works, the presence of an extension header is acceptable even if a router does not know its purpose. If a router that finds the extension header knows its purpose, it will realise that the packet contains tunnel authentication to be processed. Therefore, the part of the internet that does not understand the extension header just ignores it and the part that understands it uses it. Moreover, in IPv4, we can either use spare bits in the header or an options field can be added.

Technically, IPv4 is obsolete, but almost half of the Internet still uses it. The way the Internet works at present, it supports both of IPv4 and IPv6. Sooner or later it will support IPv6 only. Our framework will work with IPv6 since it requires no changes to the IPv6 structure.

3.6 Summary

To summarise with, Chapter 3 explains how the proposed tunnelling framework looks like. It also discusses the main parts of the framework, i.e., network operation, Directory Service Broker (DSB), path computation tool at the end user, including how they function. Then the chapter ends with how the ticketing service works to ensure the authenticated end user's access to the tunnels.

Chapter 4 explains how we have designed our tool for the tunnelling framework.

Chapter 4

Design and Implementation of the Baseline Route Selection Tool

4.1 Design and Implementation

A framework is built to investigate the benefits of using different percentages of tunnels present in a part of the Internet for sending data from one AS to another.

The framework takes an AS topology as input and produces an Autonomous System Border Router (ASBR) topology assuming that there is a peering of border routers (formed by one from each of the connecting ASes) at the point where the two AS domains are connected to each other. We are aware that the route between the adjacent border routers of two connecting ASes does not necessarily have to be one-to-one, rather there can be one-to-several connections. At present, our tool does not consider this case, but later on the tool could incorporate this feature, if desired.

Moreover, in a single AS, the border routers are all inter-connected, but the connection will not necessarily be direct; rather more than one internal hops may exist between a pair of border routers. Our system does not require this knowledge as well as the fact that operators will not share this information. Therefore, the view of the topology the broker

has, is not necessarily a complete one. We can call it a "sanitised" or an "artificial" view of the internet map. It just shows how the various ASes are inter connected at the AS level. Depending on this AS view, the ASBR topology is produced.

The software at the user's machine knows the AS topology and the ASBR topology and calculates the least cost path. The framework developed initially, uses Dijkstra's Algorithm to calculate the least cost routes with tunnels and without tunnels, for traffic to be sent. The routes are given as output of the tool showing the sequence of the ASBRs for the traffic from the source AS to the destination AS.

Sections 4.1.1 and 4.1.2 explain how the framework creates the AS and ASBR topology.

4.1.1 AS Topology

Internet topology maps are important for characterising the internet infrastructure, understanding and analysing its properties, behaviour and evaluation. Construction of these maps is possible for different layers, e.g., fibre, IP address, router, Point-of-Presence, Internet Service Provider (ISP) and Autonomous Systems (AS) [174]. However, the AS-level topology is crucial for design and evaluation of future internet protocols [182].

As stated before in Section 2.1, the Internet consists of ASes where inter-domain routing information is exchanged via BGP. Many studies [183–185] have highlighted the importance of AS topology and many research works have been done to construct it [173, 185, 186]. Section 2.6 provides with a literature review for this.

For this research, we view the entire Internet as an AS-topology graph taking each AS as a node and each BGP connection between a pair of ASes as a link. Although research is still going on to find a precise way of constructing the AS topology, it is possible to get the information. The data published by CAIDA is an example where they have used three data sources: traceroute, BGP and IRR data to compare the AS topology graphs [174].

At this stage, we consider the AS topology graph for part of the Internet from which we can find infer the connections among the ASes based on their adjacency, as shown in Figure 4.1.

From Figure 4.1 we can generate Table 4.1

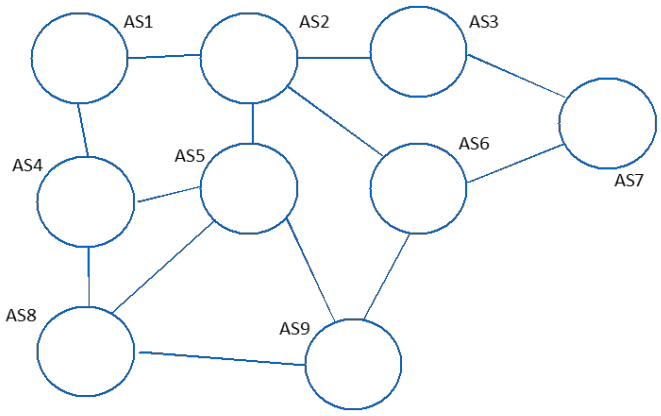


Figure 4.1: Graph of AS topology for a part of the Internet

Table 4.1: AS-Level topology from Figure 4.1

Source AS	Neighbour AS(es)
AS1	AS2, AS4
AS2	AS1, AS3, AS5, AS6
AS3	AS2, AS7
AS4	AS1, AS5, AS8
AS5	AS2, AS4, AS8, AS9
AS6	AS2, AS7, AS9
AS7	AS3, AS6
AS8	AS4, AS5, AS9
AS9	AS5, AS6, AS8

We have used the PFP network topology generator in order to generate a small regional AS-level internet topology that is later fed into the framework tool as an input file. The reasons behind the choice are briefly explained in Section 2.6.4.

PFP takes a small AS-level topology and then generates an expected number of nodes with a specific node degree distribution, which are pre-defined in the program. As explained in Section 2.6.5, for certain probability ratios, PFP generates new artificial nodes having connections with existing old/ host node and new links in between the already existing old nodes. The newly created nodes are named as "Artificial Nodes" and the old ones are classified as "Nodes".

A small topology of 7 ASes with their connections, as shown in Table 4.2 is given to the PFP tool.

The first column represents a source AS and the next one is another AS connected to it.

The PFP will then generate a topology with a certain number of nodes

Table 4.2: A small topology of 7 ASes

Source AS	Adjacent AS
Node-1	Node-2
Node-2	Node-3
Node-1	Node-4
Node-1	Node-3
Node-1	Node-5
Node-5	Node-6
Node-6	Node-1
Node-6	Node-7
Node-5	Node-3

it is asked for, having connection to the old nodes as well. E.g., *Artificial-Node-3 Node-5* means a new node **3** is created having a link to old node **5**. Again, *Artificial-Node-3 Artificial-Node-1* indicates that the new node **3** has a link to another new node **1**.

A sample of the output for 30 ASes topology is included in the appendix A. The developed framework takes the AS-level topology as an input and creates a linked-list data structure of that storing all the AS domain nodes there. The next job is to separate the source nodes and the destination nodes i.e., the ASes which are connected via links to the source AS and storing them in linked lists, where it keeps reading the nodes until reaching the end of the file.

However, we understand that the connections can be unidirectional or bidirectional and for our research the connection is assumed to be bidirectional, i.e., if there is a connection between ASes 1 and 7, traffic can be sent both from AS1 to AS7 and from AS7 to AS1.

So, the framework creates a data structure to store all the nodes there.

An image of the output source and destination ASes, using a PFP-generated topology as input is included in Appendix A.

4.1.2 ASBR Topology

The AS topology developed from the PFP is fed into the framework developed for this research, which then produces another topology at the level of ASBRs (Autonomous Systems Border Routers), assuming there is a peering of border routers (formed by one from each of the connecting ASes) at the point where the two AS domains are connected to each other. Moreover, within a single AS, the border routers are inter-

connected into a full mesh, but the connections need not necessarily be direct; rather more than one internal hop may exist between a pair of border routers. Our system does not require this knowledge, nor do operators need to share this information.

Table 4.3 can be formed from Table 4.1, assuming that the ASBRs are identified with a number as:
"SourceAS_DestinationAS"

Table 4.3: ASBR Topology for Table 4.1

Source ASBR	Destination ASBR(s)
1_2	2_1, 1_4
1_4	4_1, 1_2
2_1	1_2, 2_3, 2_5, 2_6
2_3	3_2, 2_1, 2_5, 2_6
2_5	5_2, 2_1, 2_3, 2_6
2_6	6_2, 2_1, 2_3, 2_5
3_2	2_3, 3_7
3_7	7_3, 3_2
4_1	4_1, 4_5, 4_8
4_5	5_4, 4_1, 4_8
4_8	8_4, 4_1, 4_5
5_2	2_5, 5_4, 5_8, 5_9
5_4	4_5, 5_2, 5_8, 5_9
5_8	8_5, 5_2, 5_4, 5_9
5_9	9_5, 5_2, 5_4, 5_8
6_2	2_6, 6_7, 6_9
6_7	7_6, 6_2, 6_9
6_9	9_6, 6_2, 6_7
7_3	3_7, 7_6
7_6	6_7, 7_3
8_4	4_8, 8_5, 8_9
8_5	5_8, 8_4, 8_9
8_9	9_8, 8_4, 8_5
9_5	5_9, 9_6, 9_8
9_6	6_9, 9_5, 9_8
9_8	8_9, 9_5, 9_6

Figure 4.2 shows the topology at the ASBR level.

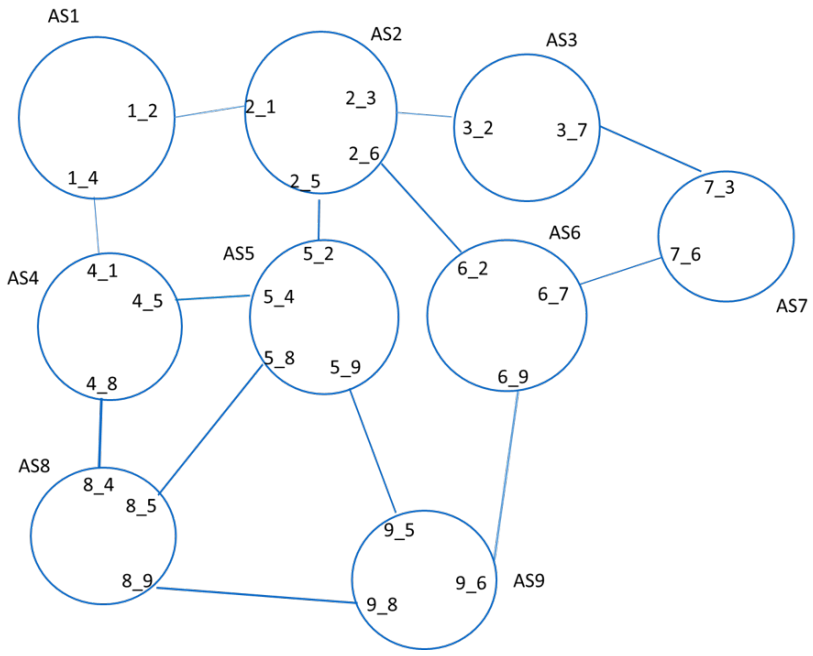


Figure 4.2: Graph of ASBR topology obtained from Figure 4.1

Appendix A includes the output ASBR topology for the AS-level topology of Appendix A.

4.1.3 Presence of Tunnels

The tool takes the expected number of tunnels as user input and generates the tunnels randomly in different ASes.

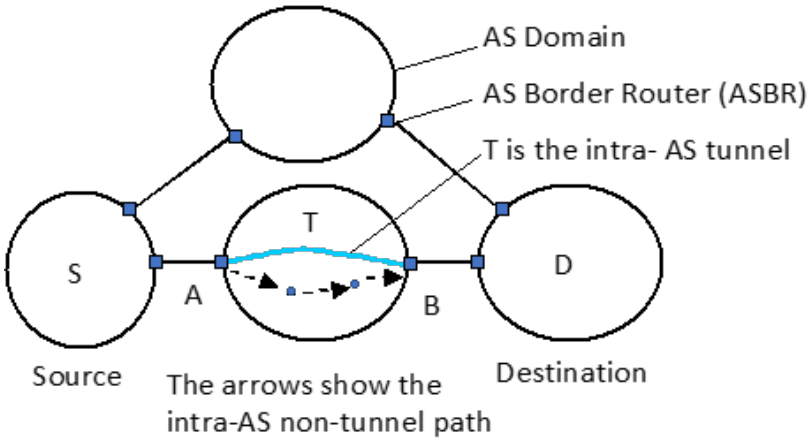


Figure 4.3: Example Intra-AS paths with and without tunnels

Figure 4.3 illustrates a simple example of alternate no-tunnel and tunnel paths within an AS. Here, the source and destination ASes are S and D and the traffic is assumed to traverse through another AS to reach the destination, which has a tunnel T with ingress point A and egress point B. The dotted lines represent normal intra-AS pathway including routers inside the AS.

Generating tunnels

While developing the basic route selection tool, two different ideas have been tested to implement it.

Firstly, the tool can generate the exact number of tunnels randomly every time it is asked to. The steps will be as follows:

1. Ask the number of tunnel(s).
2. Generate tunnel(s) allocating them in some randomly picked AS(es).
3. Ask for a number greater than the last one.
4. Delete the previous tunnel(s).
5. Generate this number of tunnels again in random AS(es).

Secondly, the tool takes the number of tunnels from the user and adds them to the existing ones obtained from the previous run. The following steps make this clearer:

1. Ask the number of tunnel(s).
2. Generate tunnel(s) allocating them in some randomly picked AS(es).
3. Keep the existing tunnel(s) present.
4. Ask for a number greater than the last one.
5. The difference of the numbers will be the new tunnel(s) required to be generated
6. Generate tunnels in AS(es) which do not already have tunnels in them.

Upon doing some test runs, it was decided to use the second idea for generating tunnels.

After the tool takes the number of tunnels, it generates the exact number of intra-AS tunnels. The program randomly creates a number

either 0 or 1, representing the probability of having tunnels where 1 indicates that there will be a tunnel and 0 means there is no tunnel. (As shown in Figure 4.4)

```
The probability of having tunnel in the AS domain 1 is 0.
There is NO tunnel in AS domain 1
The probability of having tunnel in the AS domain 6 is 1.
There is a mesh tunnel in AS domain 6
The probability of having tunnel in the AS domain 2 is 0.
There is NO tunnel in AS domain 2
The probability of having tunnel in the AS domain 3 is 0.
There is NO tunnel in AS domain 3
The probability of having tunnel in the AS domain 4 is 0.
There is NO tunnel in AS domain 4
The probability of having tunnel in the AS domain 5 is 1.
There is a mesh tunnel in AS domain 5
The probability of having tunnel in the AS domain 7 is 0.
There is NO tunnel in AS domain 7
```

Figure 4.4: Randomly generated tunnels in a small topology

Here, two tunnels are randomly generated and placed in AS6 and AS5.

Figure 4.5 gives a visual representation of the tunnelled topology.

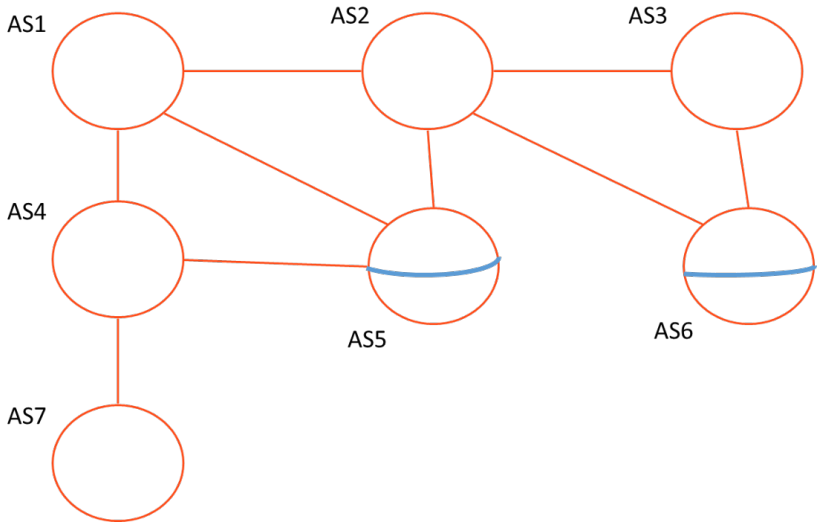


Figure 4.5: A small topology with tunnels in two ASes

The blue lines represent two intra-domain tunnels.

4.1.4 Least Cost Path

We have used Dijkstra's Algorithm to calculate the least cost routes for traffic to be sent from any source AS to any destination AS. The paths

include the ASBRs that the traffic needs to traverse to reach the destination.

Assigning cost

The adjacency matrix for the Dijkstra's Algorithm is built by the program assigning a set of costs to the normal no-tunnel paths and the tunnels. For now, the cost of the routes is considered using the metric of "delay" in milliseconds. A data packet typically needs to go through 4 to 6 hops within a given AS while traversing across a number of ASes to reach to the destination [187]. Hence, an intra-AS tunnel having the ingress and egress points in the same AS can reduce the delay that is experienced relative to the normal no-tunnel intra-AS links. This is particularly true if the normal pathways are congested and some form of priority is given to the tunnels, be that through the use of separate optical channels or queueing priority along shared links.

Here, the associated cost along each of the links is the (mean) delay in milliseconds. The four types of delay contributing to the total end-to-end delay are: transmission delay (Tx), propagation delay, processing delay and queueing delay. The propagation delay between the ASBRs A and B will be same for the no-tunnel normal path and the tunnel. [188] shows that the processing delay matters although both of processing and transmission delay are proportionately small. Hence, queueing delay is the one that typically contributes most to the delay experienced. [188] also confirms that processing and queueing delays are the ones that are usually considered in terms of measurements and simulations.

The amount of delay experienced via tunnels versus no-tunnel intra-AS paths and the corresponding cost ratio have been chosen carefully after doing some research on Internet delay measurements [188–191].

Note that, in the sections discussion related to the initial route selection tool we have developed, we use the term "cost" as a short form of "cost metric", which is actually the average end-to-end delay.

Calculating least cost path

Next, Our tool uses Dijkstra's Algorithm to calculate the no-tunnel least cost path depending on these allocated costs. After that, the tool generates a given percentage of tunnels in the produced AS topology. Taking the expected number of tunnels as user input, the tool places different percentages of tunnels in randomly chosen ASes and calculates the

least cost path again considering the tunnels in and the least cost path included the tunnels if and only if the delay cost of the tunnels is less than that of the no-tunnel paths. For now, we assume an AS which is selected for hosting tunnels, has them arranged in a full mesh between the ASBRs of the AS.

Both for no tunnels and certain percentages of tunnels being present in the topology, the least cost path is calculated for sending traffic from each of the ASes to every other ASes, i.e., for 30 ASes, a least cost path will be calculated to send the data to each of the rest twenty-nine ASes. Most importantly, the tool considers the tunnels only if their use makes the cost cheaper than the normal no-tunnel paths.

Appendix A has the example output file with the least cost path calculated from AS1 to all the other twenty-nine ASes of AS topology showed in Appendix A.

4.1.5 Flowchart

Section 4.1 gives the idea about what the developed framework is doing. For a clearer representation, a flow chart is attached here.

For a better understanding of the tool, the steps can be explained briefly:

- A text file containing simple AS topologies (which can be designed from a graph of the internet) is given as an input to the tool.
- From the data in the text file, the tool separates the source AS and destination ASes and prints it out on the terminal.
- The tool creates a text file containing the AS topology clearly indicating the source AS and the destination ASes from it.
- Then the sequence of ASBRs is worked out by the tool and it is also printed out on the terminal showing the source ASBR and the destination ASBRs (which have a direct link to the source ASBR) from it.
- Similar to the AS topology, the tool creates a text file with the ASBR topology.
- Next, the tool takes the expected number of tunnels as inputs, to be present in the topology generated.
- It generates the exact number of tunnels randomly placed in different ASes and creates a text file containing this information.

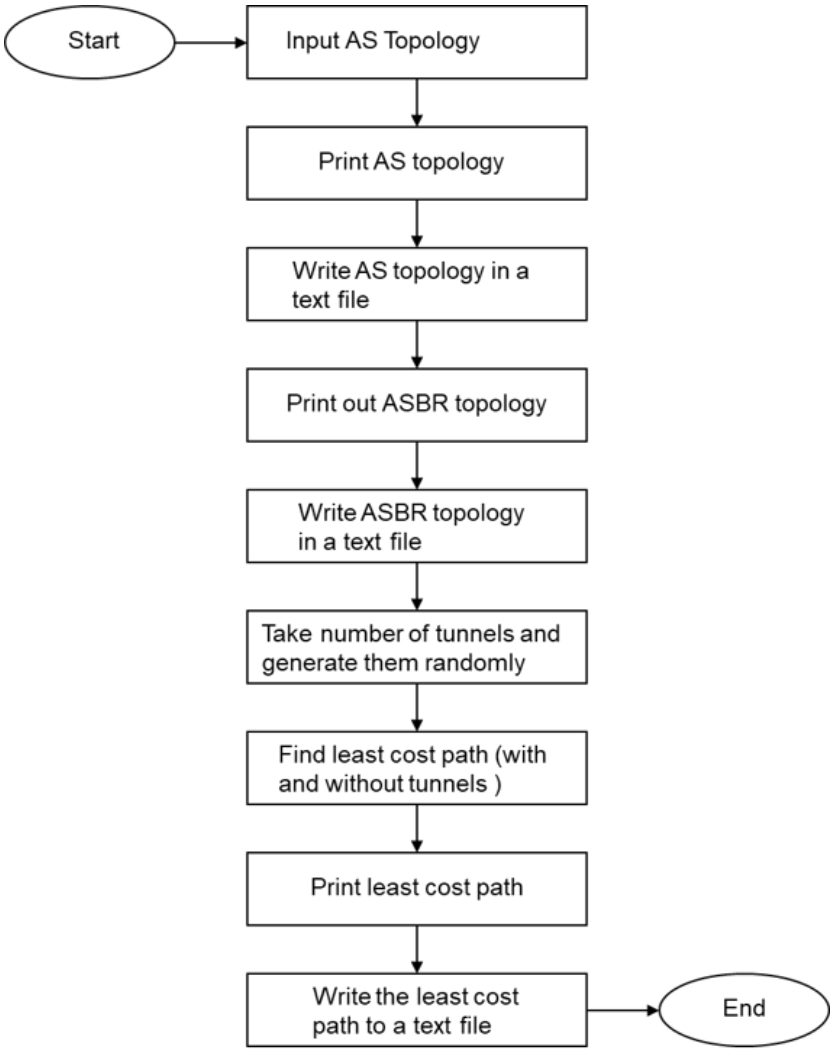


Figure 4.6: Flowchart showing the steps to validate the baseline framework

- Then the tool implements the Dijkstra's algorithm on the source and destination ASes to find the least cost path from each AS of the topology to every other AS both for the having the tunnels present and not.
- The least cost path is printed out on the terminal.
- A final text file is created containing the least cost paths.

However, although a small topology was generated by the framework after being provided with an input text file, now we have chosen an

existing internet topology to generate a larger AS-level topology which is then used as the input for our framework.

4.1.6 Pseudo Code

This section presents a pseudo code that gives an idea about how the framework is programmed (in C).

Algorithm 1 Underlying Framework

INPUT: *Graph,* $G(V, E)$
Number of tunnels, N_T
Delay cost for tunnel link, d_t
Delay cost for no-tunnel (intra-AS) link, d_{nt}
Delay cost for inter-AS link, d_{as}

OUTPUT: *Least Cost Path,* P'

```
while G ← TRUE do
  G_AS ← Generate tool-AS topology
  G_ASBR ← Generate ASBR topology
  T ← Tunnels to generate in random ASes
  N_AS ← Number of AS nodes
  Generate random number, rnd, where rnd is either 0 or 1
  for i ← 1 to N_p do
    if rnd = 0 then
      Generate tunnel
    else
      Do not generate tunnel
    end if
  end for
  N_p ← Number of possible S − D paths
  Produce cost matrix
  for i ← 1 to N_p do
    d_π ← 0
    if L is inter AS then
      d_π ← d_π + d_as
    else if L is intra-AS then
      d_π ← d_π + d_nt
    end if
    Use Dijkstra's Algorithm and sort d_π
  end for
  P' ← P with minimum d_π
end while
return P'
```

4.1.7 Data Structure

As explained in the previous sub-sections, the code uses data structures to save ASes, ASBRs, tunnels and the cost assigned. Figure 4.7 is a representation of all the data structures with their properties and associations that we have used in programming the framework.

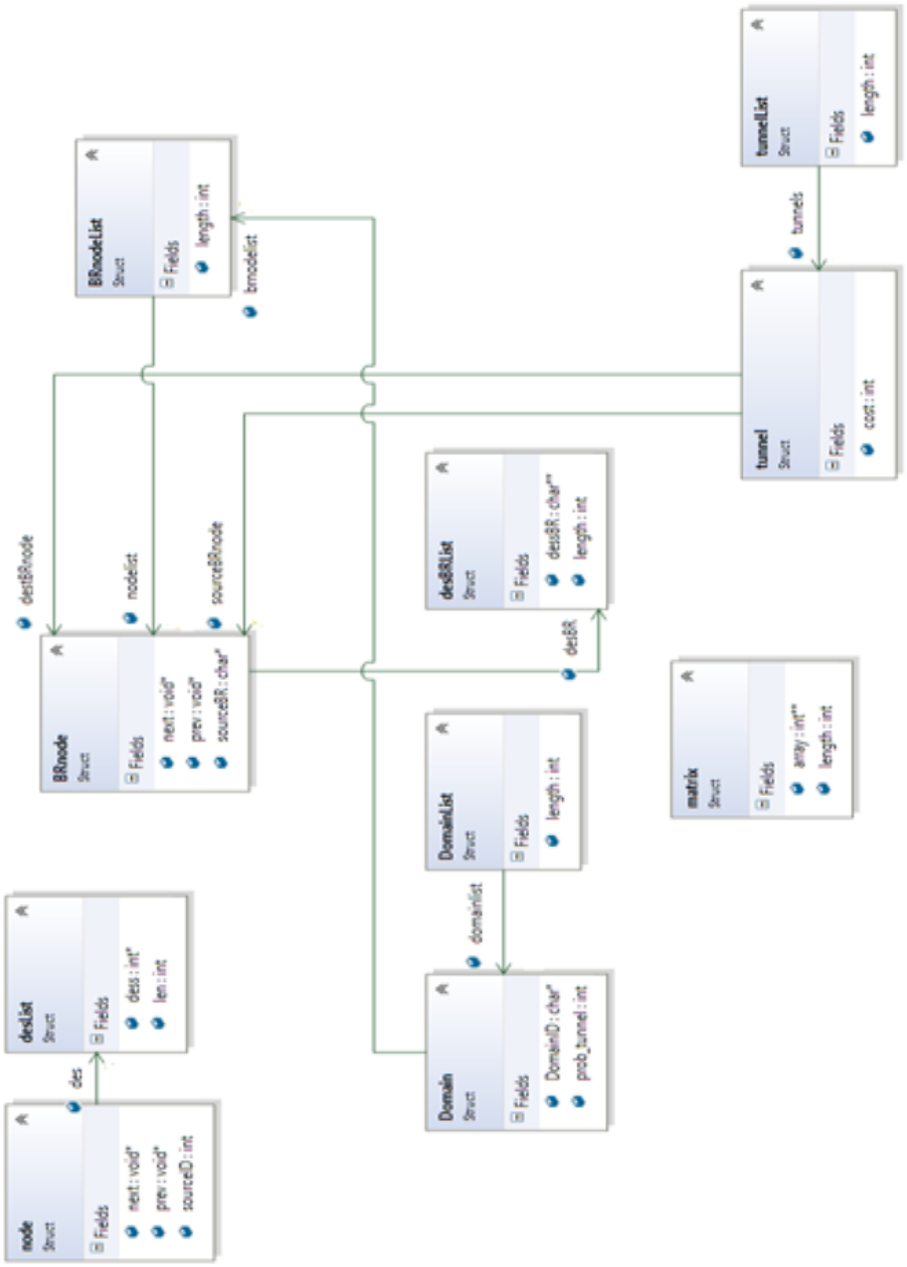


Figure 4.7: Illustration of the data structures used for developing the framework

4.2 Results and Evaluation

We have performed a number of simulations in order to assess the benefits of using tunnels in a regional network topology.

To start with, the PFP generator is used to generate different topologies with same network properties and the probability of a node gaining a new link, which is a function of the node degree, is 0.048 [58]. Section 2.6.5 gives a brief idea about how PFP works and what characteristics are to be considered while using it.

A small topology having 7 ASes is provided to the PFP tool, which later gives a larger topology of 30 ASes as asked. We did that for two different node degrees- **3** and **4** to investigate the benefit for both. Here, the cost associated to each link is the average end-to-end delay.

Taking the PFP-generated AS-level topology as input, the developed framework produces a topology at the ASBR level. Next Dijksta's Algorithm calculates the least cost path from every AS to all the remaining ASes. Then the presence of 5%, 10%, 15%, 20%, 25% and 30% tunnels are consequently added to the topology and least cost paths are again calculated for every percentage.

Here, no inter-domain tunnels have been considered and the cost of a link between the peering border routers of two adjacent ASes is set to 1 millisecond.

The benefit of the tunnels present is calculated as follows:

$$\begin{aligned}&\text{Benefit from AS } \mathbf{A} \text{ to AS } \mathbf{B} \text{ for } x\% \text{ tunnels} \\ &\quad = \Big[\text{cost from } \mathbf{A} \text{ to } \mathbf{B} \text{ using no tunnels} \\ &- \text{the cost from } \mathbf{A} \text{ to } \mathbf{B} \text{ when } x\% \text{ tunnels are present}\Big] \text{ milliseconds}\end{aligned} \tag{4.1}$$

The costs are automatically calculated using Dijkstra's algorithm for each least cost path and then the average and standard deviation of these differences is calculated. It should be noted that in many cases there would be no cost benefit of going via one or more tunnels when they are remote from the original no-tunnel pathway. This tunnel-placement process is repeated 10 times for a given overall AS topology and the average and standard deviation of the benefits are calculated. Then, the results are plotted in graphs to make a clearer representation of the benefits.

Table 4.4 includes the parameters used for the tool.

Table 4.4: Parameters used in the baseline route selection tool

Parameter	Value
Delay cost for tunnel	1
Delay cost for no-tunnel link	3 or 4
Delay cost for tunnel during peak time	1
Delay cost for tunnel during peak time	15
Delay for inter-domain link	1

4.2.1 Results for Different Topologies

The five PFP generated topologies that we have used for the experiment, are included in the Appendix A. At first, for all the experiments, unless otherwise stated, the ratio of the cost of a tunnel in an AS to that of a normal no-tunnel path is set as 1:3 and the Dijkstra's algorithm calculates the least cost path considering that the cost of using a tunnel is 1ms while that of a no-tunnels intra-AS path is 3ms. And then the average and standard deviation are calculated after 10 runs for each topology as mentioned above.

For the different percentages, the expected exact number of tunnels in a 30 AS topology are shown in Table 4.5.

Table 4.5: Number of tunnels in the AS topology consisting of 30 ASes

Percentage	Calculated Number in 30 ASes	Number of Tunnels Generated
5%	1.5	1 or 2
10%	3	3
15%	4.5	4 or 5
20%	6	6
25%	7.5	7 or 8
30%	9.5	9 or 10

Table 4.6 summarises the results for the different topologies having no tunnels as well as different percentages of tunnels implemented in them.

Table 4.6: Average and standard deviation of the benefits of using tunnels

Tunnel Percentage	**Topology1** Avg/Std (ms)	**Topology2** Avg/Std (ms)	**Topology3** Avg/Std (ms)	**Topology4** Avg/Std (ms)	**Topology5** Avg/Std (ms)
5%	0.32506/ 0.49298	0.10698/ 0.42596	0.30943/ 0.60180	0.19172/ 0.46187	0.05333/ 0.28789
10%	0.43035/ 0.64778	0.43813/ 0.73694	0.46006/ 0.79010	0.38161/ 0.74438	0.21839/ 0.57428
15%	0.82242/ 0.89388	0.59418/ 0.89978	0.65701/ 0.95350	0.49885/ 0.83423	0.55840/ 0.78606
20%	0.98318/ 1.04425	0.84782/ 1.05771	0.81839/ 1.04909	0.77471/ 1.04058	0.70391/ 0.92151
25%	1.21760/ 1.12095	0.94515/ 1.13862	0.87218/ 1.09265	0.95035/ 1.14451	0.77793/ 0.97733
30%	1.27256/ 1.13427	1.10450/ 1.23033	0.99816/ 1.17400	1.07172/ 1.19032	0.91494/ 1.02871

As expected, as the proportion of tunnels increase so does the average benefit. When the percentage of tunnels is small, the average benefit is marginal. However, from the standard deviation, we can see that some users, located close to the tunnels can still obtain a considerable benefit.

These results are plotted in graphs, where X-axis shows the different percentages of tunnels present in the network topology and Y-axis represents the benefits.

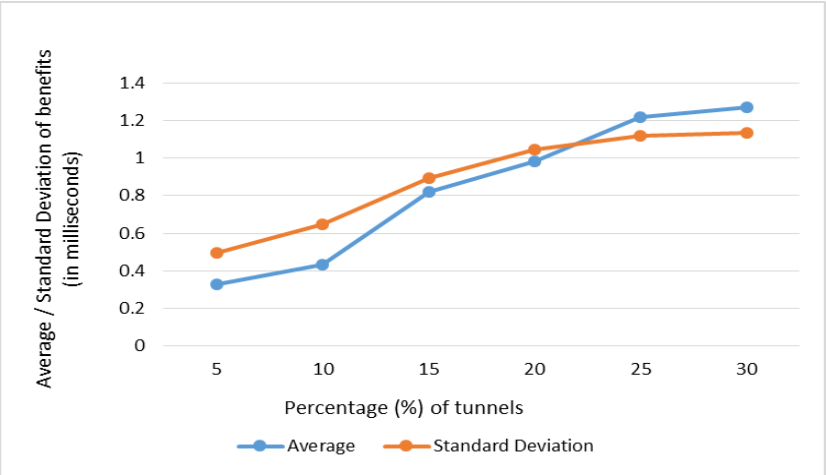


Figure 4.8: Average and standard deviation of cost benefit for Topology 1 (Ratio of tunnel to no-tunnel cost = 1:3)

As shown in the Graph 4.8, even for only 5% tunnels present in the topology, there is at least some benefit, although it is not that high. For one tunnel randomly placed in any one AS amongst the thirty ASes of topology 1, there is an average benefit of almost 0.33 milliseconds which increases to 0.8 milliseconds for 7 tunnels present in random ASes. For every increment in the number of tunnels, the average benefits keep increasing and finally, for 9 tunnels, the average benefit is almost 1.3 milliseconds.

For topology 2, Figure 4.9 shows that the average benefit for 5% tunnel present is approximately 0.11 milliseconds which is less than half of the average benefit for same percentage of tunnels present in topology 1, same number of ASes and with same network properties. This is due to the fact that not necessarily the tunnel will make same benefit for all the source- to-destination routes for sending a data over the network. However, as observed for topology 1, the average benefits keep increasing as the number of tunnels increases.

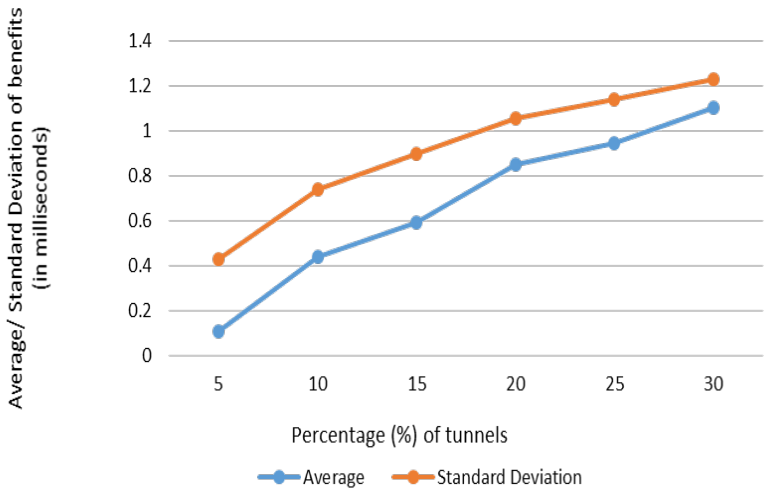


Figure 4.9: Average and standard deviation of cost benefit for Topology 2 (Ratio of tunnel to no-Tunnel cost = 1:3)

For topology 3, the average benefit keeps increasing with the number of tunnels present. The graph becomes slope as shown in Figure 4.10. This behaviour is because of the fact that although there is at least one new tunnel added to the topology, not necessarily it will be used by the end user. So, for a specific occurrence of sending a traffic over the network, use of tunnels will not always make noticeable benefits.

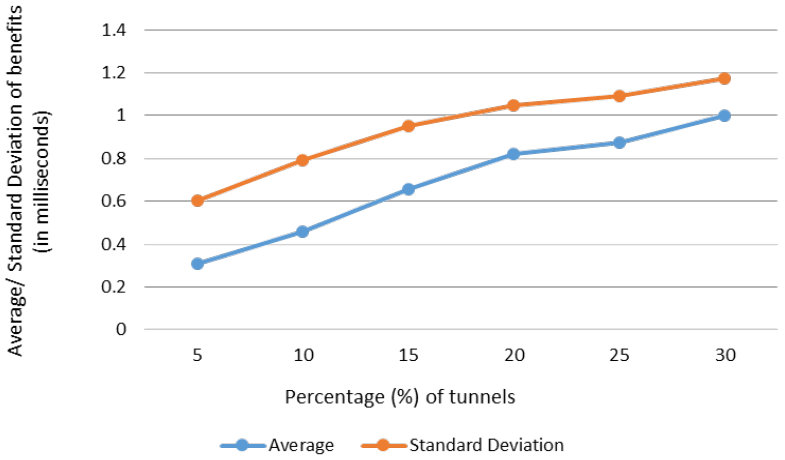


Figure 4.10: Average and standard deviation of cost benefit for Topology 3 (Ratio of tunnel to no-Tunnel cost = 1:3)

In Figure 4.11, the line showing the average benefit reaches at the point of 1.07 milliseconds starting from 0.19 milliseconds ascertaining that there is benefit of the tunnels being present in the network topology 4.

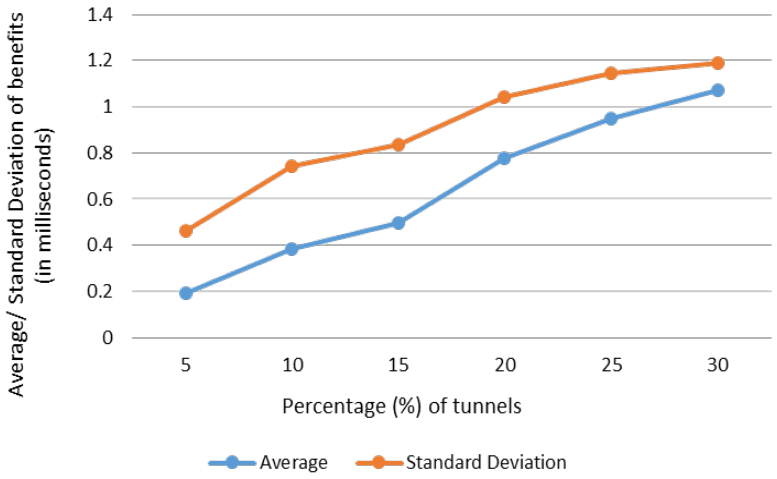


Figure 4.11: Average and standard deviation of cost benefit for Topology 4 (Ratio of tunnel to no-Tunnel cost = 1:3)

For topology 5, Figure 4.12, for the first two increase of 5% tunnels, the blue line showing average benefit increases by large dimen-

sion. Then the increment becomes a little slower for the next two sets of increment and again it shows rise in the performance for 30% tunnels.

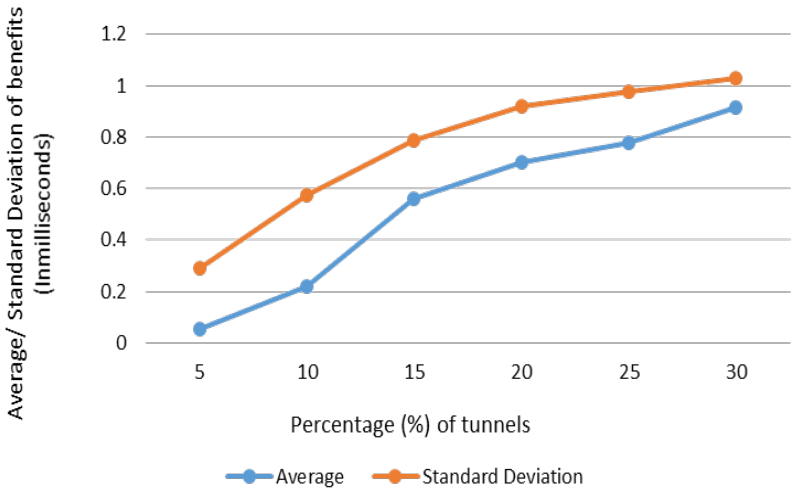


Figure 4.12: Average and standard deviation of cost benefit for Topology 5 (Ratio of tunnel to no-Tunnel cost = 1:3)

Finally, it is clear from the graphs that the benefit increases, as there is an increase in the percentage of tunnels present in the Internet. As seen, the average improvement is relatively small. This is not surprising, as many paths would incur a costly diversion to reach tunnel(s), particularly when they are few in number. However, the increasing standard deviation shows that between a smaller numbers of source-destination pairs, the cost benefit can be substantial.

4.2.2 Results for Different Cost Ratio

We have observed how the benefit of using tunnels changes with the change in the cost ratio of tunnels and no-tunnel links in a particular topology. We have used Topology 4 for this.

For this, different cost ratios are considered for 10 runs. While choosing the cost ratios, at first, we have been conservative and considered the average delay cost for no-tunnel paths as $3x$ milliseconds and $4x$ where the average delay for a tunnel is x millisecond. Then we have considered a situation representing traffic congestion where the average no-tunnel link's cost is $15x$. At certain times, the Internet can be busy, impacting on the end-to-end delay. Usually, queueing delay makes a greater contribution in such cases [188].

Figure 4.13 presents a graph plotting the average benefit for different percentages of tunnels for Topology 4.

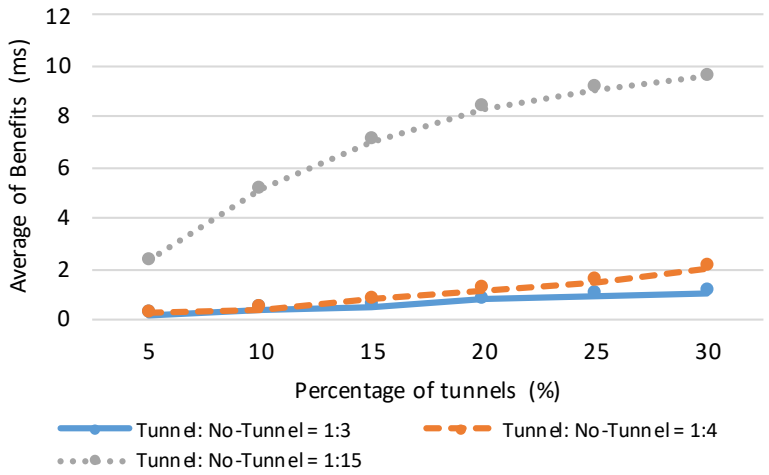


Figure 4.13: Average of cost benefit for different cost ratio

It is clear from the graph that for all cost ratios, the benefit increases as there is an increase in the percentage of tunnels present in the Internet. With a ratio of 1:3, the average delay for sending data in the topology 4.97ms which is reduced by a minimum of 0.19ms when 5% of ASes have tunnels in them. The average benefit gradually reaches almost 1.08ms for 30% tunnels. For a tunnel/ no-tunnel ratio set to 1:4, the average end-to-end delay without the use of tunnels is 5.97ms. With 30% tunnels, this end-to-end delay goes down by 2.03ms.

It can be seen that the average improvement is relatively small when the tunnel's average delay cost is one-third or one-quarter of the normal no-tunnel average delay. This is not surprising, as many paths would incur a costly diversion to reach tunnel(s), particularly when they are few in number. Even so, a decrease of almost 2ms compared with almost 4ms to 6ms could still be of attraction to at least some end-users for specific application services.

Conversely, for the "busy" period, the average benefit associated with greater cost ratios are noticeably high. When we consider the cost associated with a no-tunnel link in a congested AS as 15ms, the average end-to-end delay for the same topology is calculated to 16.97ms. As expected, exploiting tunnels within this AS lowers the delay to a great extent resulting in more average benefit. For 10% tunnels the average benefit is more than 5ms and for 30% it reaches almost 9.6ms.

The standard deviation of the benefit is plotted in Figure 4.14.

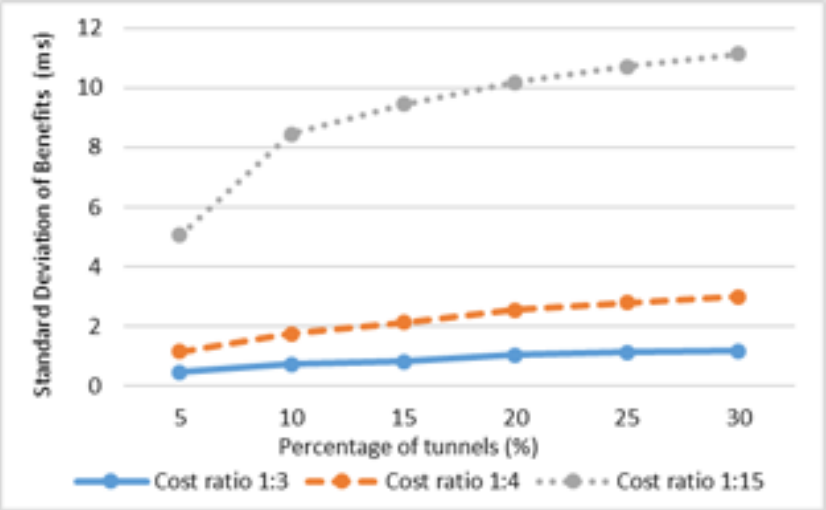


Figure 4.14: Standard deviation of cost benefit for different cost ratio

The increasing standard deviation shows that between a smaller numbers of source-destination pairs, the cost benefit can be substantial. Indeed, it is worth noting that during peak hours or when specific high-demand events occur, the intra-AS queueing delay can be tens of milliseconds if not more. If tunnels bypass such "hot spots" the delay cost benefit could be orders of magnitude providing end-users considerable benefit in terms of delay.

4.2.3 Results for Different Node Degree

The PFP generator is used again to generate an AS-topology from the same initial 7-node seed graph that has been used to generate the topologies used above. However, this time the graph evolution is altered by setting the average node degree to 4.

As with the previous simulations, we have considered the use of tunnels under normal traffic conditions and during a period of localised congestion where the ratio of the cost for tunnel to that of no-tunnel path is 1:4 and 1:15 within the specified ASes.

Then, for both cases the average of the delay cost benefit for the presence of 5%, 10%, 15%, 20%, 25% and 30% tunnels were calculated.

It can be seen from the graph that the greater inter-AS connectivity (since the node degree is greater) has a marginal improvement of the no-tunnel paths and thus the benefit of the tunnels is slightly reduced.

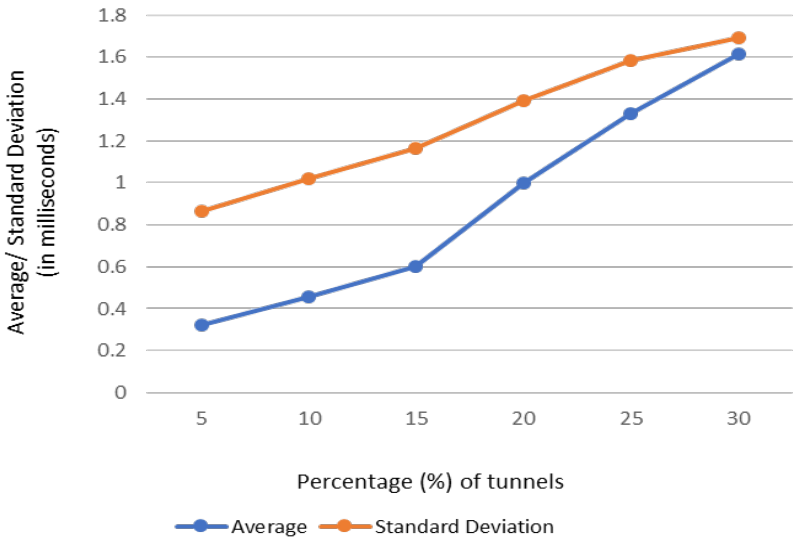


Figure 4.15: Average and standard deviation of cost benefit for different node degree

Now, from Figure 4.15, we can observe that for the AS level topology, the average end-to-end delay without any tunnel is 5.97ms which decreases when tunnels are available to end users. If the tunnel has an average delay cost of 1/4th of the normal intra-AS link path then it gives an average end-to-end delay benefit of 0.32ms, which increasing number of tunnels and for 30% tunnels reaches 1.62ms.

For the busy period conditions, we make the assumption that the tunnel will have an average delay of 1/15th of the average normal intra-AS link delay. For the no-tunnel topology, the average end-to-end delay is 14.94ms. Clearly the graph plotted in Figure 4.16 shows that the availability of different percentages of tunnels adds benefit by improving the average delay cost. For 15% of tunnels the average reduction in delay is 4.73ms and for 30% it is almost double, 8.95ms and it is approximately 6 times more than the benefit we observed for the ratio of tunnel/ no tunnel cost of 1:4.

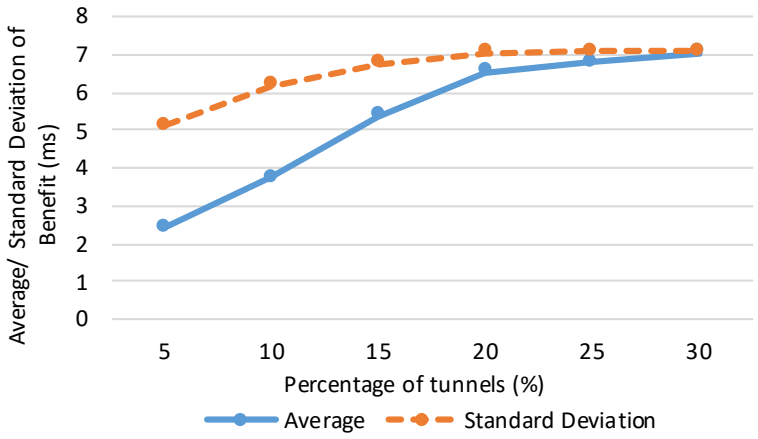


Figure 4.16: Average of cost benefit for different node degree and cost ratio

Hence, it is clear from the graphs that during peak times, even the presence of a small percentage of tunnels can provide noticeable benefit to many users by decreasing the delay cost.

4.2.4 Considering "Hotspot" Area

In Section 1.1, we have mentioned that the lack of traffic differentiation can lead to load imbalances and "best effort" equal treatment of all traffic irrespective of its importance to the user. Keeping this situation in mind, we conduct a series of simulations for a situation where all the source ASes want to send their data traffic to a particular destination AS over the Internet. This approximates the situation where the destination AS hosts a popular server farm or datacentre.

In this case, the framework is changed in such a way that upon calculating the expected number of tunnels for a specified tunnel-percentage, it generates the tunnels in ASes adjacent to the destination AS first. If the number of expected tunnels is more than the number of adjacent ASes, then the rest of the tunnels are generated to the ASes which are one hop away and so on. Thus the tunnels are organised into approximately concentric rings around the destination AS.

Noting the benefits of tunnel-usage are more pronounced and meaningful for "peak time" situations we have again run 10 simulations for the same topology with 30 ASes, used in Section 4.2.3 (with a node degree of 4) where the average delay cost for tunnel is 1ms and that of

normal path is 15ms. Taking AS2 as the destination AS, and assuming each of the remaining 29 ASes act as the source domains, Figure 4.17 presents the graph of average and standard deviation of the benefit for using tunnels around a "hotspot" destination.

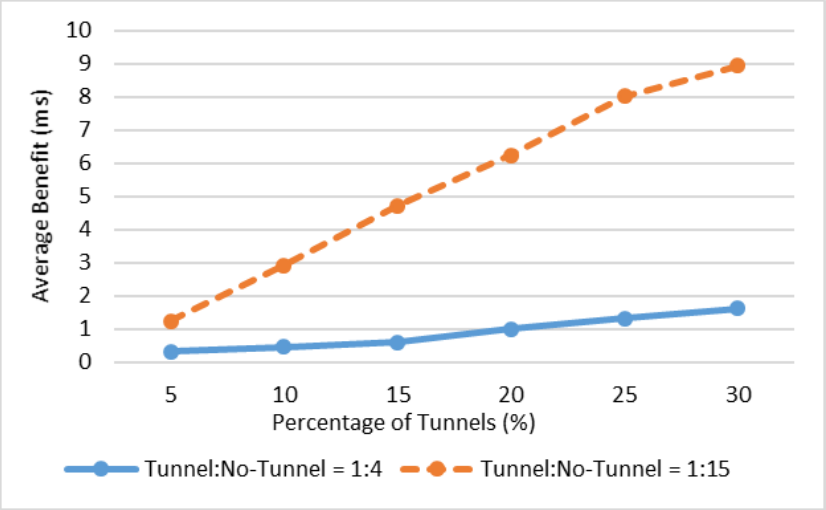


Figure 4.17: Average and standard deviation of cost benefit for tunnel-no tunnel = 1:15)

The baseline average end-to-end delay for sending data to AS2 from all the other domains is calculated to be 9.5ms. It is clearly observed that both of the average and standard deviation of the delay cost benefit of employing a given percentage of tunnels is relatively high compared the ones we have observed in Figures 4.13,4.14, 4.15, and 4.16, even for only 5% of tunnels near a hotspot destination in the network topology. Hence in case of known "hot spots" in terms of desirability and possible congestion, access to low delay tunnels become particularly attractive.

4.3 Summary

Using the developed framework we are able to examine the delay benefits that intra-AS tunnels might bring to the Internet. It shows that there is a benefit for even 5% tunnels in the network for some users, though this is dependent on how close the tunnel alternatives are to the default traditional pathway.

To show the variation in benefit between source-destination tuples we provide the standard deviation. However as one standard deviation only encompasses about 68% of a normally distributed population, it is

worth noting that for some uses the cost benefit would be appreciable. When the standard path experiences delays brought about by "hot-spot" congestion then tunnel alternatives become much more attractive.

The results observed have been published in [192, 193].

Chapter 5

Path Computation Algorithm for Tunnels using GA (PCAT–I)

A path computation tool has been developed (using Python) that is able to take more than one constraint in account and to calculate the best suitable path for an end user to send its data to the desired destination. The problem is Multi-Objective, where the two objectives are:

1. To minimise the average end-to-end delay.
2. To minimise the financial cost.

The tool uses an Evolutionary Algorithm (EA) to find the suitable path(s). The idea of implementing EAs to find shortest path is not novel, but their use in finding benefits of tunnels in the network is novel.

The tool is named as Path Computation Algorithm for Tunnels (PCAT). For this research purpose, two different versions of PCAT are developed:

- PCAT–I: The first version of the tool is developed using a Genetic Algorithm (GA). It converts the Multi-Objective Optimisation Problem (MOOP) into a Single Objective Optimisation Problem (SOOP).
- PCAT–II: The second and final version is developed using a Multi-Objective Evolutionary Algorithm (MOEA), Strength Pareto Evolutionary Algorithm (SPEA).

In the rest of the thesis, the tool is referred as Path Computation Algorithm for Tunnels (PCAT).

This chapter discusses the PCAT–I.

The idea of adapting the features of GA to design routing algorithms is not new. One of the earliest algorithms, Genetic Adaptive Routing Algorithm (GARA) was introduced in the late 1990s by Munetemo [194–198]. To-date, the application of GA for shortest path routing has been proved to be a point of interest for the research community [114, 120–128].

5.1 Design and Implementation

The initial design of of the PCAT is same for both the versions and the main difference is how the path computation is done using two different algorithms.

5.1.1 AS Topology

A small python program is written to get an AS topology in the expected format of a csv file which is then fed into the main PCAT. At first, the code takes an AS-level topology generated using PFP [83]. Table 5.1 shows a sample PFP-generated topology.

Table 5.1: PFP-generated AS-level topology of 7 ASes

Source AS	Destination AS
Node-1	Node-2
Node-2	Node-3
Node-2	Node-5
Node-1	Node-4
Node-5	Node-3
Node-1	Node-5
Node-2	Node-6
Node-6	Node-3
Node-5	Node-4
Node-4	Node-7

Taking this as an input, the code generates an output file showing bi-directed connections between each AS. The output file for Table 5.1 is shown in Table 5.2. Here, in each row, the first node is a source AS and the second node is a destination AS.

Table 5.2: Source and destination ASes from Table 5.1

Source;Destination
1;2
2;1
1;4
4;1
1;5
5;1
2;3
3;2
2;5
5;2
2;6
6;2
3;5
5;3
3;6
6;3
4;5
5;4
4;7
7;4

The final AS topology is built from the connections of the ASes, which is then ready to provide to the main PCAT. Table 5.3 shows the final AS topology obtained from the PFP-generated topology in Table 5.1. In each row, the first node is a source AS and the rest of the nodes are the ones directly connected to the source AS.

Table 5.3: Input format AS-topology for the PCAT

1;2;4;5
2;1;3;5;6
3;2;5;6
4;1;5;7
5;1;2;3;4
6;2;3
7;4

Figure 5.1 illustrates the topology with the AS-level connections.

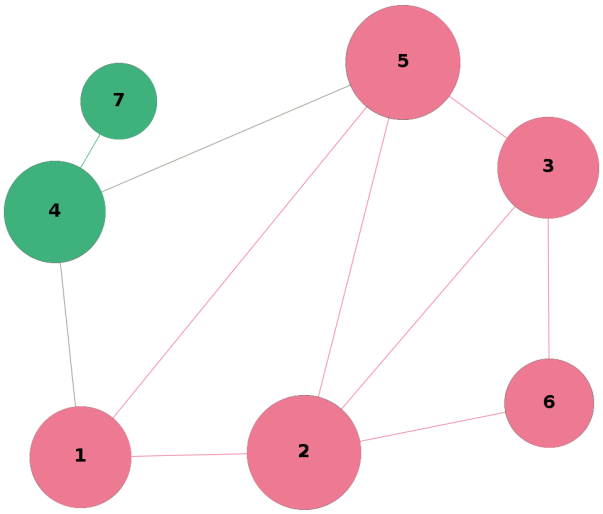


Figure 5.1: AS topology of 7 ASes

5.1.2 Generating Tunnels

The tool takes the expected percentage of tunnels to be present in the network topology from the user. It then generates the exact number of tunnels in randomly selected ASes. If more tunnels are expected, the tool keeps adding new tunnels to the existing ones obtained from the previous run. The steps involved in the process are:

1. Ask the percentage of tunnel(s).
2. Calculate the exact number of tunnel(s) to generate from the provided percentage.
3. Generate tunnel(s) allocating them in randomly picked AS(es).
4. Keep the existing tunnel(s) present.
5. Ask for a percentage greater than the last one.
6. The difference of the previous percentage from the new one will be the new tunnel(s) percentage required to be generated.
7. Generate new tunnels in AS(es) which do not already have tunnels in them.

5.1.3 Calculating the Best Suitable Path

Each tunnel and no-tunnel link is associated with a delay and a financial cost. Section 4.1.4 has the explanation of the choice made for the ratio of the average delay experienced using tunnel path and no-tunnel path. It is assumed that the network operators will be willing to implement tunnels upon being benefited financially. End users will pay for each tunnel they select to use along the path from the source the AS to the destination AS. There is also a small notional cost for each tunnel, which has been made the same for all existing tunnels for a specific topology. The PCAT–I finds one best path, which is alternatively called as the "elite chromosome" of GA.

All the paths are found at the ASBR-level. It is assumed that there is a one-to-one peering connection between the ASBRs of two adjacent ASes. Also the routers within a single AS are assumed to have a full mesh connection. Section 4.1.2 provides a brief description of this. In the PCAT–I, the two metrics are converted into one, weighing both of the assigned metrics. The elite path represents a path having the least possible end-to-end delay with least possible financial cost to be paid by the end user.

Section 5.1.4 explains how the GA is implemented for path computation.

5.1.4 Implementation of GA

This section briefly discusses how the main components of an EA are designed and implemented in the developed path computation tool, PCAT–I.

Initial Paths (Population Initialisation)

Each of the solution or population is encoded as a chromosome. A chromosome just contains a set of values which represents one specific solution to a particular problem. The chromosome encodes what a solution will be. It can be a certain combination of colours or a certain sequence of links. So, the encoding describes a specific solution. It is not necessary for the chromosomes to be of the same length.

While generating the initial population for GA, two issues are considered [94, 114, 199]:

Firstly, the size of the initial population: Initially, it was thought that a large population is required to obtain a better solution. It was

later proved that a larger number of population is expensive in terms of memory and time [200–202], whilst being of limited benefit. The works presented in [114, 196, 203] have used the number of nodes present in the topology as the size of initial population. [127] used the number of neighbour nodes as the population size.

The developed PCAT–I was tested and evaluated with different sizes of population and then the number of AS nodes in the topology was selected as the initial population size.

Secondly, the procedure of initialising the population: This can be done heuristically or randomly. Heuristic initialisation of the population can make the GA perform faster, but this incorporates a high chance of compromising the solution space which reduces the population diversity making the algorithm unlikely to find the to optimal solution in the end [199]. The loss of diversity is a noteworthy issue when implementing the standard GA in shortest path problems [127].

In this work, the initial population is generated randomly. However, care is taken so that the same solution is not revisited several times. Moreover, during "environmental selection", new random chromosomes are injected in every iteration which assures the diversity in the population.

The possible paths between a source AS and a destination AS initial paths are the initial population or "chromosomes" for the GA. A possible path from source S to destination D is obtained using the following steps:

- Start from a link adjacent to S.
- Form a link to a randomly chosen node adjacent to S.
- With equal probability, choose the next node adjacent to the current one.
- If the node is not being revisited, then form a link to it. The links in between need to be adjacent to each other to make the connections valid. These links are the "genes" for mapping the initial chromosomes. Each valid link connects the ASBRs of a pair of adjacent AS nodes. If an invalid link is encountered while mapping the chromosome, the path is not valid and the algorithm comes up with another one.

  If a link between ASBR nodes N_1 and N_2 is L_{N_1,N_2}, then it can be

defined using Equation 5.1.

$$L_{N_1,N_2} = \begin{cases} 1 & \text{if } N_1 \text{ and } N_2 \text{ are adjacent to each other} \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

- There should not be any loop i.e., duplicated nodes in the path. If there are any, the algorithm will compact the duplicate ones and the nodes in between them into one. Figure 5.2 shows an example of this.

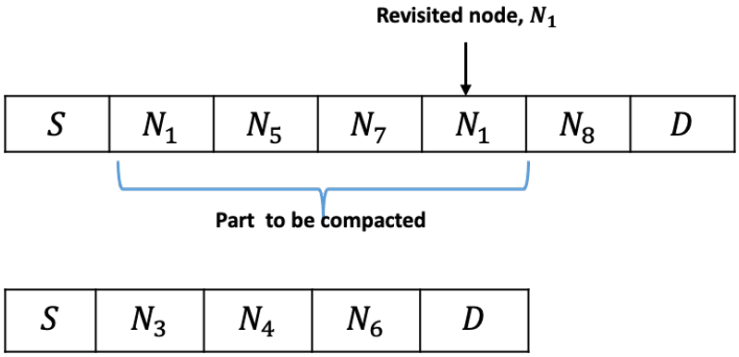


Figure 5.2: Example of compacting duplicate nodes

- For each tunnel link present in a valid path, the probability of choosing it to use it is a design parameter and is set to 50%.
- End when a link adjacent to D is found.

Equation 5.2 explains whether the path P is going to be considered or not.

$$P_{S,D} = \begin{cases} 1 & \text{if the path from } S \text{ to } D \text{ is valid} \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

Figure 5.3 represents the genetic encoding of a path from S to D. Note that, in the rest of the thesis, N_i represents ASBR nodes, not AS nodes.

This encoding considering the feasible paths only easily portrays the path length as well as successfully reducing the search space [126, 204, 205].

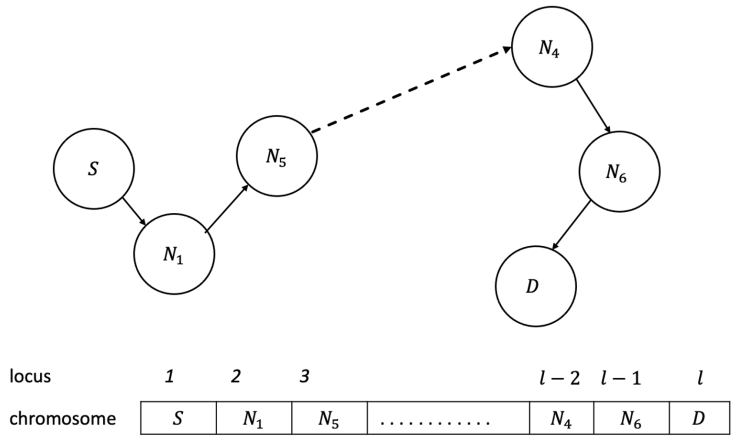


Figure 5.3: Example encoding of an initial population

Calculate Fitness (Evaluation)

The problem given to PCAT–I is to find a path between a source to a destination having a minimum end-to-end delay and a minimum total cost. Hence, this is a minimisation problem.

The algorithm has a fitness assignment process for the transformation between the objective function and the fitness function. The calculation of the fitness score is done considering the average end-to-end delay and financial cost for each path. The two objective functions are: delay fitness and cost fitness. Since the final fitness score is a single value, so this is calculated by applying some weight to delay fitness and cost fitness. The component terms are decided in such a way that higher values are either both better or both worse (not a mix). For this minimisation problem, the smaller the value is, the fitter the path is.

Delay Fitness: In a network graph G with nodes V and edges or links E, the delay metric along a path from a source AS S to a destination AS D, is calculated by adding up each of the average end-to-end delay associated with each tunnel and no-tunnel links. This can be mathematically represented using Equation 5.3.

$$d_\pi = \sum_{(i,j)\in E} d_{ij} \tag{5.3}$$

where d_{ij} represents the delay metric for a link (i, j) used in the path.

The algorithm finds the maximum possible delay, d_{max}, to be experienced for the particular pair of source and destination ASes. Then the delay fitness, f_1 for any path from the S to D is calculated using

Equation 5.4.

$$f_1 = \frac{d_\pi}{1 + d_{max}} \tag{5.4}$$

In the denominator, 1 is added to d_{max} so that the value of f_1 is never equal to or greater than 1.

Cost Fitness: The calculation of cost fitness is based on the total financial cost the end user will need to bear for choosing a path for its data packets to traverse through from a source AS S to a destination AS D. The cost of using one intra-AS tunnel in the network graph is c_t. Equation 5.5 represents the financial cost of a path.

$$c_\pi = \sum_{(i,j) \in E} c_{ij} \tag{5.5}$$

Finally, the cost fitness for a path from S to D, f_2 is calculated using Equation 5.6.

$$f_2 = \frac{c_\pi}{c_t + c_\pi} \tag{5.6}$$

The addition of c_t to c_π in the denominator confirms that the value of f_2 is always less than 1.

Final Fitness Score: The final fitness score is then calculated giving a particular weight to delay fitness and cost fitness. Each path is then evaluated based on the final fitness score. This way the problem is converted to a SOOP for the GA. The weighting is controlled by an alpha term (α). Equation 5.7 calculates the final fitness score.

$$F = \alpha \times f_1 + (1 - \alpha) \times f_2 \tag{5.7}$$

Rank Paths (Mating Selection)

Besides the basic selection schemes (as discussed in Section 2.9.3), some improved selection schemes have been proposed as well. E.g., [111] proposes a method combined of roulette wheel and elitist mechanisms. In this work, the selection process is rank-based and uses the idea of elitism so that the fittest path from each iteration is never lost in the process of evolution. The paths are ranked based on their overall

fitness score. For a minimisation problem, the smaller value of fitness score a path has, the fitter it is. A certain percentage (80%) of the paths are then selected as fitter paths which are suitable for mating. This way ensures that while selecting the fitter paths, not necessarily all the less fit paths are discarded. Rather some of them can still be selected for mating which may result into fitter paths after going through the process of reproduction. The fittest path here is the "elite".

Crossover (Reproduction)

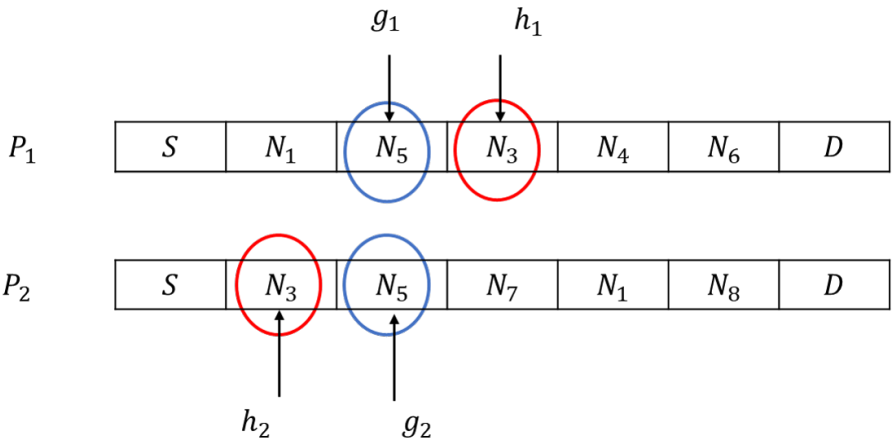
Crossover is the first step for the fitter paths to go through the reproduction procedure. In the traditional crossover, only the information of the gene located at the crossover point is exchanged between two selected chromosomes [124]. Unlike this, in path crossover, some part of the genes (nodes taking part in forming the paths) are exchanged between the paths. The crossover can be single point or double-point. In a single point crossover, there is only one point of exchange in of information in the paths [206]. In double point cross over, each path has two points from where it exchanges the information with another path. This PCAT-I has been developed using single point crossover, which combines parts of two parent paths and creates two new children paths. The probability of a path going through a crossover is set for the algorithm. The crossover probability suggested in the research works varies from 0.5 to 1.0 [115, 124, 207, 208]. In most of the works proposing the use of GA for shortest path problem in the network, the crossover probability is high; e.g., [121] uses 0.99, [122] uses 1 and [123] uses 0.7. After doing some research and number of simulations, the probability for PCAT–I is set as 0.6.

The following steps explain the crossover mechanism:

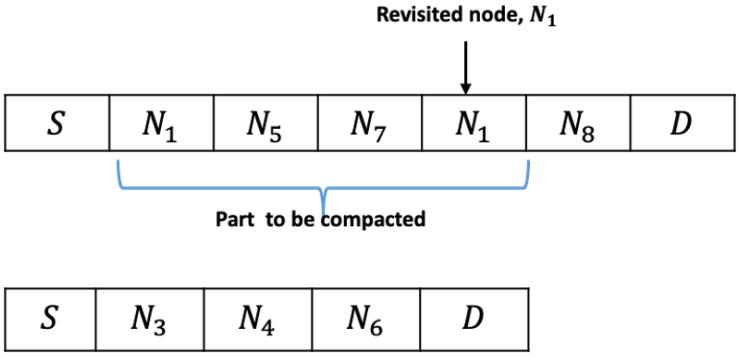
- Select a pair of parent paths P_1 and P_2 randomly.
- Generate random number, rnd
- If the crossover probability, $rnd \leq \rho_c$, then do crossover.
- Read the ASBR nodes (N) in P_1 one by one and look for the similar node in P_2.
- If found, denote that node of P_1 as g_1 and the one in P_2 as g_2.
- Replace all nodes between g_1 and destination node D in P_1 by those between g_2 of the parent path P_2 and destination D in P_2.

- If any node is revisited, the algorithm will compact the duplicate ones and the nodes in between them into one.
- This forms the children path after crossover, C_1.
- The path P_1 remains unchanged and represents C_1 in two possible cases, if:
  - there is a loop in the newly formed path.
  - there is no common node.
- Read the ASBR nodes (N) in P_2 one by one and look for the similar node in P_1.
- If found, denote that node of P_2 as h_2 and the one in P_1 as h_1.
- Replace all nodes between h_2 and destination node D in P_2 by those between h_1 and D in P_1.
- If any node is revisited, the algorithm will compact the duplicate ones and the nodes in between them into one.
- This forms the crossed over path, C_2.
- If there is no common node, the path P_2 remains unchanged and represents C_2, if:
  - there is a loop in the newly formed path.
  - there is no common node.
- Randomly select the next pair.
- If the pair has been selected before, generate a new pair and follow from the step of generating crossover probability.

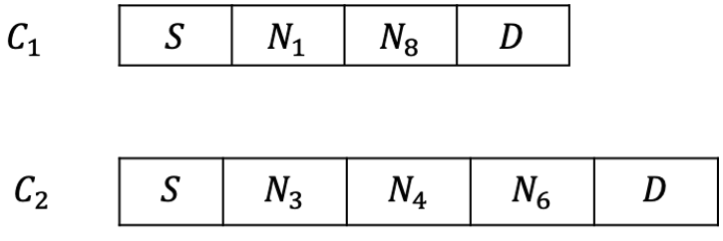
Figure 5.4 represents the overall procedure of the crossover on two randomly selected paths P_1 and P_2 for a specific crossover probability.



(a)



(b)



(c)

Figure 5.4: The overall procedure of crossover: (a) Parent paths; (b) Paths after crossover with loop; (c) Children paths after crossover.

Mutation (Reproduction)

The children paths after crossover, C then undergo mutation which helps to maintain the diversity in the paths by making some amendments. Like the crossover over probability, probability of mutation or mutation probability is also pre-specified. Although in biology, the rate of mutation can be as low as 0.001 [115, 124, 207, 208], this is not proven to be the best to produce the best result for the shortest path calculation. [122] and [123] have used 0.05, whereas in [121] the mutation probability is set as 0.1. For the developed PCAT–I, it has been set as 5%. The mutation is performed as follows:

- Select a path, C_1.
- Generate a random number, rnd.
- If $rnd \leq \rho_m$, then do mutation on the selected path.
- Generate a random mutation position n_1, except the position for S and D.
- If the node at position n_1 is x_1, generate a valid path from x_1 and D.
- Replace the previous part of C_1 with the newly generated one.
- If any node is revisited, the algorithm will compact the duplicate ones and the nodes in between them into one.
- This will form the mutated path, M_1
- If there is no other valid path than the existing one, then keep C_1 unchanged.
- Select the next path from C.
- Follow the steps for all the paths in C.

Figure 5.5 gives an overall idea of mutation on a specific path.

Inject Random Paths (Environmental Selection)

To assure the diversity of the solutions and the possibility of less fit paths to be a part of the next generation, a number of newly calculated possible paths for the same pair of source and destination are injected. These paths are obtained following the same steps as for initialisation. n number of paths to be generated at this step so that the final number

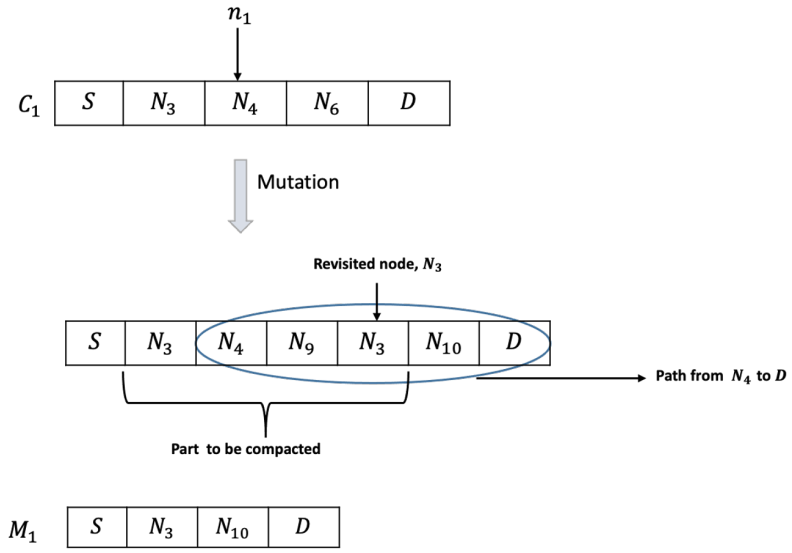


Figure 5.5: Mutation operation

of output paths is same as the size of initial population. These paths are described as "secondary paths" in this thesis.

Finally, a set of paths is sent to the next generation. The elite path, resultant mutated paths and the secondary paths are merged to create the next population.

Terminating Condition

The algorithm is iterated a number of times (i.e., generations) to get the final output and it needs a terminating condition to stop the iterations. In this work, it is set as the maximum number of iterations. The algorithm has been run with setting different number of iterations as the terminating condition. In every trial, the number was set as the multiples of 10 and it was tested for network topologies of different sizes. It was observed that for small topologies, having 7 and 10 ASes, only 5 and 10 iterations can generate the best solution. For topologies having uptp 50 ASes, a maximum of 30 iterations generate a good solution and the further iterations could not find a better solution. Hence, the number of iterations, that works as the terminating condition, is set as 30. Once the number of iterations reaches the maximum, the algorithm stops and the elite path produced for the last iteration is the final solution path produced using the algorithm.

5.1.5 Flowchart

Flowchart of PCAT–I

Figure 5.6 represents a high level flowchart of the GA-based Path Computation Tool.

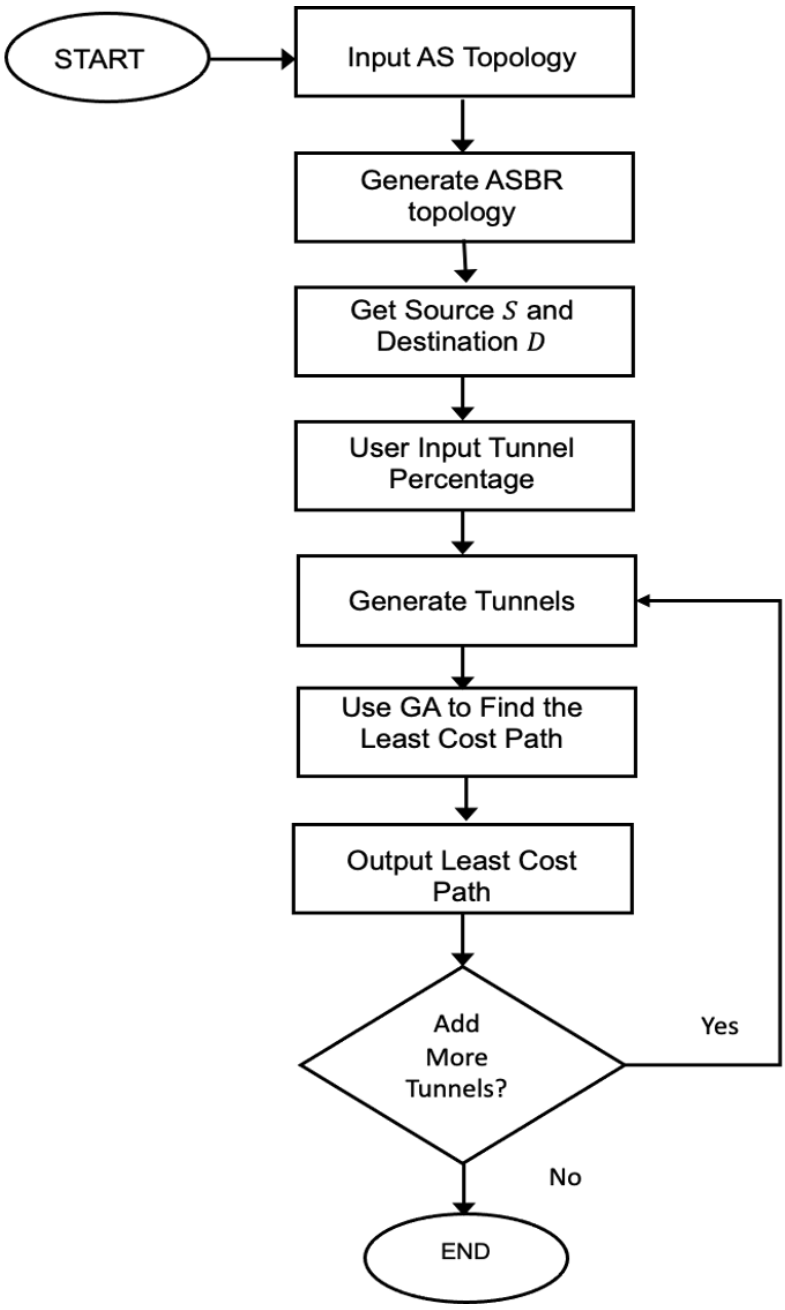


Figure 5.6: Flowchart of PCAT–I

For a better understanding of the tool, the steps can be explained briefly:

- A text file containing an AS topology (which can be generated using PFP topology generator tool [83]) is given as an input to the tool.
- From the data in the text file, the tool finds the connecting AS(es) from each AS.
- The tool creates a csv file containing the AS topology clearly indicating the source AS and the destination ASes from it.
- Then the sequence of ASBRs is worked out by the tool and it is also printed out on the terminal showing the source ASBR and the destination ASBRs (which have a direct link to the source ASBR) from it.
- Similar to the AS topology, the tool creates a csv file with the ASBR topology.
- Next, the tool takes the expected percentage of tunnels (0%-99%) as input, to be present in the topology.
- It generates the exact number of tunnels randomly placed in different ASes and creates a text file containing this information.
- Then the tool implements Genetic Algorithm to find the best suitable path.
- It repeats the process of taking user input for generating more tunnels until the user does not want any more of the tunnels. New tunnels are added to the existing ones.
- The best path(s) for the expected percentage of tunnel(s) is(are) then presented to the user.

Flowchart of GA implementation

The implementation of the GA itself can be represented with a flowchart presented in Figure 5.7.

For a better understanding of this part, the steps can be explained briefly:

- A number of possible paths are generated for sending data from source AS S to Destination AS D. This is the set of initial population.

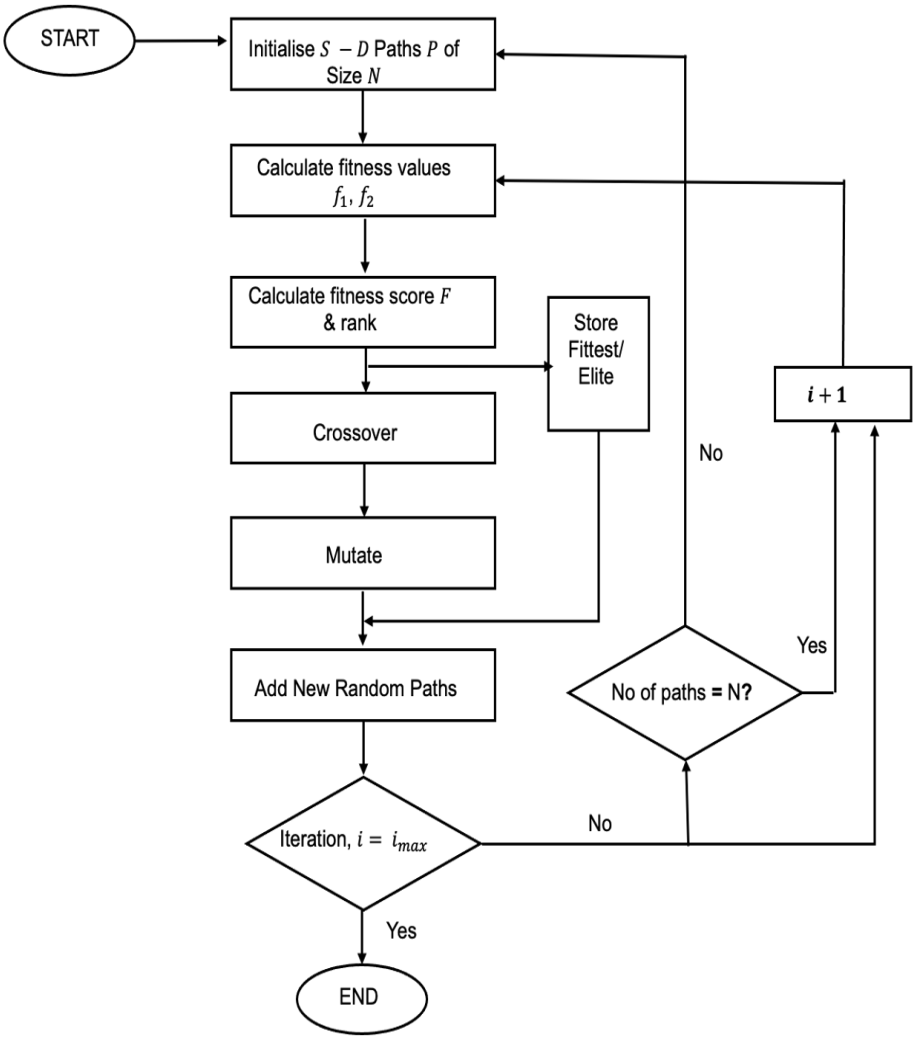


Figure 5.7: Flowchart showing the steps in GA implemented for path computation

- The total end-to-end delay and financial cost are obtained for each path by adding up the assigned delay and cost to the tunnel and no-tunnel links used in the path. From these, the delay fitness and cost fitness are calculated using Equations 5.4 and 5.6.
- The algorithm then uses another equation developed for finding the final fitness score of each path.
- Then the paths are ranked and sorted according to the ascending order of the fitness score since the smaller fitness score represents fitter path.

- The path with the smalleast fitness score is the fittest path, which is known as the “elite” chromosome.
- Next, the PCAT–I selects a number of top-ranked paths from the initial ones which are fit for going through the mating pool to produce new paths from themselves.
- For a certain crossover probability, the selected paths undergo the process of crossover where a pair of paths exchange information at a single point.
- For a certain mutation probability, each the selected paths undergo the process of mutation.
- New paths are generated from the source to destination, which represent a set of secondary population.
- The elite path, mutated paths and the newly generated paths are combined together to form a new set of paths. This is the initial population for the next iteration.
- If the total number of paths in the new population is not the same as the initial population size, then the algorithm starts over.

  To maintain the population size, the size of secondary population is defined in such a way that after combining them with the elite and mutated population, the total number of paths is same as that in the initial population.
- The elite path is the best suitable path from the current iteration.
- The final set of a previous iteration is the population for the next one; i.e., there is no need to generate possible paths anymore.
- The algorithm stops when it reaches the maximum number of iterations.
- The elite path of the final iteration is the output of the PCAT–I which represents the best suitable path from S to D.

5.1.6 Pseudo Code

This section presents a pseudo code of the PCAT–I, programmed implementing GA (in Python).

Algorithm 2 PCAT–I

INPUT: *Graph,* $G(V, E)$; *Source,* S; *Destination,* D;
Iteration limit, i_{max}; *Tunnel Percentage,* β
Delay for tunnel link, d_t; *Cost for tunnel link,* c_t
Delay for no-tunnel (intra-AS) link, d_{nt}
Cost for no-tunnel (intra-AS) link, c_{nt}
Delay for Inter-AS link, d_{as}; *Cost for Inter-AS link,* c_{as}
Crossover probability, ρ_c; *Mutation Probability,* ρ_m
OUTPUT: *Least Cost Path,* P'

```
while itr ≤ i_max do
    T ← Tunnels to generate in random ASes
    N_T ← (β*N)/100 Number of tunnelled ASes
    if β > 0 then
        Generate tunnels in N_t ASes
    else
        Do not generate tunnels
    end if
    P ← Randomly generated Initial paths //Algorithm 3
    N ← Size of P
    F ← Fitness Score //Algorithm 4
    Sort P in ascending order of F
    P' ← P with minimum F
    N_mp ← Mating pool
    N_mp ← (80%*N)/100, Size of mating pool
    C ← Paths after Crossover(P, N_mp, ρ_c) //Algorithm 5
    M ← Paths after Mutation (C, N_mp, ρ_m) //Algorithm 6
    P_s ← Randomly generated secondary paths // using Algorithm 3
    N_Ps ← ((20%*N)/100 − 1), Size of Secondary population
    P ← P' ∪ M ∪ P_s, Final set of Population for next iteration
    itr ← (itr + 1)
end while
return P'
```

Algorithm 3 Initialisation of Paths

INPUT: $G(V,E), S, D, T, d_{nt}$, d_{as}, c_{nt}, c_{as}
OUTPUT: P and *Maximum delay,* d_{max}

$P_{all} \leftarrow$ All Possible $S-D$ paths
$N_p \leftarrow$ Number of possible $S-D$ paths
$L \leftarrow$ Links in N_p
$P \leftarrow$ Initial population paths
$N \leftarrow$ Initial population size
for $i \leftarrow 1$ to N_p **do**
  $d_\pi \leftarrow 0$
  if L is inter AS **then**
    $d_\pi \leftarrow d_\pi + d_{as}$
  else if L is intra-AS **then**
    $d_\pi \leftarrow d_\pi + d_{nt}$
  end if
  sort d_π
  $d_{max} \leftarrow$ largest d_π
end for
return d_{max}, P

Algorithm 4 Evaluation

INPUT: P, T, d_t, d_{nt}, $d_{as}, c_t, c_{nt}, c_{as}$
OUTPUT: f_1, f_2, F

for $i \leftarrow 1$ to N_p **do**
  $d_\pi \leftarrow 0$
  $c_\pi \leftarrow 0$
  if $L_{i,j}$ is in tunnelled AS **then**
    Generate random number rnd in range $(1,2)$
    if $rnd \leftarrow 1$ **then**
      $d_\pi \leftarrow d_\pi + d_t$
      $c_\pi \leftarrow c_\pi + c_t$
    else
      $d_\pi \leftarrow d_\pi + d_{nt}$
      $c_\pi \leftarrow c_\pi + c_{nt}$
    end if
  else
    $d_\pi \leftarrow d_\pi + d_{as}$
    $c_\pi \leftarrow c_\pi + c_{as}$
  end if
  $f_1 \leftarrow \frac{d_\pi}{1+d_{max}}$
  $f_2 \leftarrow \frac{c_\pi}{c_t+c_\pi}$
  $F \leftarrow \alpha \times f_1 + (1-\alpha) \times f_2$
end for
return f_1, f_2, F

Algorithm 5 Crossover

INPUT: P, N_{mp}, ρ_c
OUTPUT: Children Paths after crossover, C

while $i \leftarrow \frac{N_{mp}}{2}$ **do**
  $l \leftarrow length(N_{mp})$
  Generate 2 random numbers, rnd_1 & rnd_2 within the range $(1 - l)$
  $P1 \leftarrow P$ at rnd_1
  $P2 \leftarrow P$ at rnd_2
  Generate random number, rnd
  if $rnd \leq \rho_c$ **then**
    Read node in P_1, g_i
    Read nodes in P_2
    if g_i in $P_1 == g_i$ in P_2 **then**
      site of cross in $P1 \leftarrow g_1$
      site of cross in $P2 \leftarrow g_2$
      $C_1 \leftarrow$ first node to g_1 in P_1+ next to g_2 to last node in P_2
    else
      $C_1 \leftarrow P_1$
    end if
    Read node in P_2, h_i
    Read nodes in P_1
    if h_i in $P_2 == h_i$ in P_1 **then**
      site of cross in $P_2 \leftarrow h_2$
      site of cross in $P_1 \leftarrow h_1$
      $C_2 \leftarrow$ first node to h_2 in P_2 + next to h_1 to last node in P_1
    else
      $C_2 \leftarrow P_2$
    end if
  else
    $C_1 \leftarrow P_1$
    $C_2 \leftarrow P_2$
  end if
end while
return C

Algorithm 6 Mutation

INPUT: C, N_{mp}, ρ_m, D
OUTPUT: Children paths after mutation, M

```
for i ← 0 to length(N_mp) do
    Select a path, C_i
    Generate random numbers, rnd
    if rnd ≤ ρ_m then
        l ← number of nodes in (C_i)
        Generate random number, r within the range (1 to l)
        g_i ← node at r
        Find a feasible path, m from the g_i to D
        //Algorithm 3, where S ← node at g_i
        M_i ← first node to the node before g_i in C_i + m
    else
        M_i ← C_i
    end if
end for
return M
```

5.2 Validation

The PCAT–I has been tested and validated step-by-step to make sure that it works as specified.

5.2.1 Possible Paths and Initial Population

In order to make sure that all the possible paths are obtained, a small topology of 7 ASes, as presented in Table 5.3, is used as an input to produce all possible paths from each node to the other nodes. A table showing all the possible paths from AS1 to every other ASes is included in Appendix B.

Table 5.4 represents all possible paths from AS2 to AS7.

Table 5.4: All possible Paths from AS2 to AS7

Path Index	Possible Paths
Path1	2_1>1_2>1_5>5_1>5_4>4_5>4_7>7_4
Path2	2_1>1_2>1_4>4_1>4_7>7_4
Path3	2_3>3_2>3_5>5_3>5_1>1_5>1_4>4_1>4_7>7_4
Path4	2_3>3_2>3_5>5_3>5_4>4_5>4_7>7_4
Path5	2_5>5_2>5_1>1_5>1_4>4_1>4_7>7_4
Path6	2_5>5_2>5_4>4_5>4_7>7_4
Path7	2_6>6_2>6_3>3_6>3_5>5_3>5_1>1_5>1_4>4_1>4_7>7_4
Path8	2_6>6_2>6_3>3_6>3_5>5_3>5_4>4_5>4_7>7_4

A 30-AS topology is then used as an input topology and a pair of source and destination ASes are selected for validating the tool with a larger topology. All possible paths are calculated between AS12 and AS16 and the PCAT–I outputs 751 possible paths. The topology is included in Appendix B.

Considering Tunnels

The tool is also tested to see if it can find all paths for having tunnels implemented in the topology. It is provided with a list of ASes having tunnels implemented in them.

Three tunnels are considered in AS1, AS4 and AS5 and the tool is asked to find all the possible paths with using 0 to 3 tunnels.

Table 5.5 presents all the possibilities of using either or all the 3 tunnels, for the path, Path1:

`2_1>1_2>1_5>5_1>5_4>4_5>4_7>7_4`

Note that first column is the Path1.

Table 5.5: Possible paths using tunnels for Path1

Possibility	No. of Tunnels Used	Tunnels Used
p1	3	1;5;4
p2	2	1;5
p3	2	1;4
p4	2	4;5
p5	1	1
p6	1	5
p7	1	4
p8	0	0

Hence, for a much larger topology, there can be thousands of possible paths for a pair of Source-Destination ASes. The possible paths mean the different combinations of whether a tunnel is being used or not along a given AS path. In order to make sure that the PCAT–I does not take long to calculate the fitter paths, the maximum number of generated possible paths is limited to 5000. The environmental selection at each iteration makes sure that no potential paths is lost. From this 5000, an initial population set is selected, for which the size is predefined as the number of nodes in the topology used. The works in [114, 196, 203] support the decision made.

5.2.2 Calculation of Fitness Value

The two fitness functions, delay fitness and cost fitness are normalised so that it is greater than 0 and smaller than 1. For cost metrics used while running the simulations for validation purpose are as follows:

- Average delay for no-tunnel link = 4
- Average delay for tunnel link = 1
- Financial cost for no-tunnel link = 1
- Financial cost for tunnel link = 5

While finding all the possible paths, the PCAT–I finds and stores the largest possible delay after sorting them. The total average end-to-end delay and financial cost for each path are calculated. Then using Equations 5.4 and 5.6 the delay fitness, f_1 and cost fitness, f_2 are calculated. Finally, using these values, Equation 5.7 calculates the overall fitness score, F of the path for any user-specific value of α.

For the $S - D$ pair $2 - 7$, the largest possible delay is 26. An output file is generated with the fitness values of the paths from Table 5.4 considering no tunnel present in the topology and using $\alpha = 0.5$. Table 5.6 includes the output data. The values are matched with the ones calculated by hand.

Table 5.6: Fitness scores for the paths shown in Table 5.4

ASBRpath	delay	cost	f_1	f_2	F
Path1	16	7	0.5926	0.5833	0.588
Path2	11	5	0.4074	0.5	0.4537
Path3	21	9	0.7778	0.6429	0.7103
Path4	16	7	0.5926	0.5833	0.588
Path5	16	7	0.5926	0.5833	0.588
Path6	11	5	0.4074	0.5	0.4537
Path7	26	11	0.9623	0.6875	0.8252
Path8	21	9	0.7778	0.6429	0.7103

The Table 5.7 has the values of delay fitness and cost fitness for Table 5.5.

Table 5.7: Fitness values for the paths shown in Table 5.5

Paths	Delay	Cost	f_1	f_2
p1	7	19	0.25925926	0.79166667
p2	10	15	0.37037037	0.75
p3	10	15	0.37037037	0.75
p4	10	15	0.37037037	0.75
p5	13	11	0.48148148	0.6875
p6	13	11	0.48148148	0.6875
p7	13	11	0.48148148	0.6875
p8	16	7	0.592592593	0.875

It is important to notice that, more than one path can have the same fitness values, resulting into the same final fitness score.

5.2.3 Selection

The paths are then sorted according to the ascending order of the fitness score and the top one is stored as the "elite" chromosome. If more than one paths in the top of the sorted list has the same value, the algorithm can pick any of the paths. From the Table 5.6, it can be seen that two paths have the same value for fitness score. In the experiment, the PCAT–I selects and stores `2_1>1_2>1_4>4_1>4_7>7_4` as the "elite" path.

5.2.4 Reproduction

All the paths are then assigned with unique parent ID starting from 0 to (Number of initial paths -1). The top 80% paths, including the elite one are selected for reproduction. This confirms that the best solution from every iteration takes part in the reproduction.

Crossover

Any pair of paths is picked for crossover and the crossover is performed if the pair has 60% crossover probability. To do so, a random number, rnd is generated ranging between 1 to 10. If $rnd \leq 6$,then crossover is done.

Figure 5.8 shows the output paths on the console for the two selected paths:

`2_5>5_2>5_1>1_5>1_4>4_1>4_7>7_4` and

`2_5>5_2>5_4>4_5>4_7>7_4`.

The rnd generated for this pair is 5.

```
Crossover:

('Nodes in P1:', ['2_5', '5_2', '5_4', '4_5', '4_7', '7_4'])
('Nodes in P2', ['2_5', '5_2', '5_1', '1_5', '1_4', '4_1', '4_7', '7_4'])
Crossed-over paths:
('C1:', '2_5>5_2>5_1>1_5>1_4>4_1>4_7>7_4')
('C2:', '2_5>5_2>5_4>4_5>4_7>7_4')
```

Figure 5.8: Two output Paths after crossover

Mutation

These paths are then sent for mutation. The probability of the path undergoing mutation process is 2%. Similar to the crossover method, a random number, rnd is generated ranging between 1 to 100. If $rnd \leq 2$, then mutation is done.

Figure 5.9 shows the information generated on the python console when rnd generated by the algorithm is 1 and hence the mutation operation is done on the path:

`2_6>6_2>6_3>3_6>3_5>5_3>5_4>4_5>4_7>7_4`

```
Mutation:
Node picked for the randomly generated mutation position is 2_6
Paths before loop 2_6>6_2>6_3>3_6>3_2>2_3>2_1>1_2>1_4>4_1>4_7>7_4
Paths after removing all loop present 2_6>6_2>6_3>3_6>3_2>2_3>2_1>1_2>1_4>4_1>4_7>7_4
Paths after removing intra AS loop 2_1>1_2>1_4>4_1>4_7>7_4
About to calculate distance and cost for 2_1>1_2>1_4>4_1>4_7>7_4
Calculated distance: 11 & cost: 5 from src path: 2_1 and dest path: 7_4
```

Figure 5.9: Console output for a path after mutation

5.2.5 Final Set of Paths

Some randomly generated paths between the $S - D$ pair are injected. This step ascertains the claimed statement that in the developed PCAT–I, the population diversity is not compromised. These are referred as the secondary population in this thesis and the secondary population size is set as:

`20% of the initial population - 1.`

This population size confirms that the total number of paths at the end of an iteration is same as that of the initial population.

The elite path, mutated paths and the secondary paths are then merged together to form the final set of population. The elite path is the output least cost path for the $S - D$ pair and the final set of becomes the population for the next iteration.

Finally, this section confirms the performance evaluation of the GA-based PCAT–I. Section 5.3 explains the results for different experimental setups.

5.3 Results and Evaluation

A number of simulations are run in order to find the least cost path using GA for sending data over in a regional network topology.

To start with, five different topologies with same network properties are generated using PFP and the probability of a node gaining a new link, which is a function of the node degree, is 0.048 [58]. Section 2.6.5 gives a detailed explanation about how PFP works and what characteristics are to be considered while using it. In each case, the same topology having 7 ASes is provided to the PFP tool, which generates a larger synthetic topology of 30 ASes as required. Taking the PFP-generated AS-level topology as input, the developed PCAT–I uses GA for calculat-

ing the least cost path from a given AS to a desired destination AS for no tunnels and different percentages of intra-domain tunnels present in the network.

This is done when there is no tunnel in any of the ASes and also for 10%, 20%, 30%, 40% and 50% of tunnels present in the topology. For the different percentages, the expected exact number of tunnels in a 30 ASes are shown in Table 5.8:

Table 5.8: Number of tunnels in the a 30-AS topology

Percentage	Calculated Number in 30 ASes	Number of Tunnels Generated
10%	3	3
20%	6	6
30%	9	9
40%	12	12
50%	15	15

For the experiments, unless otherwise stated, the ratio of the average end-to-end delay of an intra-domain tunnel to that of a tunnel is 1:4 and the ratio of financial cost for using a tunnel link to that of a no-tunnel link is set as 5:1.

Table 5.9 includes the parameters used for the PCAT–I.

Table 5.9: Parameters used in the PCAT–I

Parameter	Value
Delay cost for tunnel	1
Delay cost for no-tunnel link	4
Delay cost for tunnel during peak time	1
Delay cost for tunnel during peak time	20
Delay for inter-domain link	1
Financial cost for tunnel	5
Financial cost for no-tunnel link	1
Financial for inter-domain link	1

The parameters used for implementing GA are shown in Table 5.10.

Table 5.10: Parameters used in GA

Parameter	Value
Initial population size, N	30
Reproduction percentage	80
Size of mating pool, N_{mp}	$\frac{80\%*N}{100} = 24$
Number of elite population, P'	1
Cross-over rate, ρ_c	0.6
Mutation rate, ρ_m	0.02
Size of secondary population, N_{ps}	$(\frac{20\%*N}{100} - 1) = 5$
Maximum number of iterations, i_{max}	30

5.3.1 Results for Different Topologies

Five topologies with the same network properties are used for the experiment. These are included in Appendix B. The total number of nodes for all the topologies is 30 and the average node degree is 3.6. Then the PCAT–I has been used to calculate the least cost path from source AS 12 to the destination AS 16 in the topology. As specified, this is done when there are no tunnels as well as when different percentages of tunnels are present in the network topologies.

For the experiments, unless mentioned otherwise, the fitness score is calculated in such a way that both the constraints of delay and financial cost associated with the paths are given equal weight. Hence, the value of α in Equation 5.7 is set as 0.5. In short, the Equation becomes:

$$F = (0.5 \times f_1) + (0.5 \times f_2) \tag{5.8}$$

where f_1 = delay fitness of the path and f_2 = cost fitness of the path.

For the same $S - D$ pair, the PCAT–I is run 10 times. In every run, the number of iterations for the GA is set as 30. Then the average and standard deviation are calculated after 10 runs.

Table 5.11 summarises the results for the five topologies.

These results are plotted in graphs, where X-axis shows the different percentages of tunnels present in the network topology and Y-axis represents the value of the fitness score, F. On the Y-axis, the values on the left hand side are for the average of the fitness score and those on the right side represent the Standard deviation of the fitness scores.

Table 5.11: Average and standard deviation of the fitness score for different topologies

Tunnel Percentage	**Topology1** Avg/Std	**Topology2** Avg/Std	**Topology3** Avg/Std	**Topology4** Avg/Std	**Topology5** Avg/Std
0%	0.528770/ 0.005711	0.534154/ 0.004093	0.506592/ 0.004422	0.439583/ 0.012182	0.517170/ 0.012439
10%	0.518975/ 0.017973	0.512323/ 0.026444	0.485196/ 0.012845	0.415351/ 0.027349	0.481129/ 0.015703
20%	0.464915/ 0.023556	0.497117/ 0.035897	0.461979/ 0.013655	0.389119/ 0.032900	0.467794/ 0.017378
30%	0.441189/ 0.031529	0.477342/ 0.039035	0.447749/ 0.017786	0.366165/ 0.041827	0.459370/ 0.018698
40%	0.415282/ 0.034279	0.437504/ 0.043127	0.430917/ 0.022715	0.317137/ 0.050503	0.443506/ 0.027945
50%	0.386493/ 0.036679	0.370445/ 0.088695	0.388637/ 0.031322	0.286419/ 0.053501	0.371834/ 0.045327

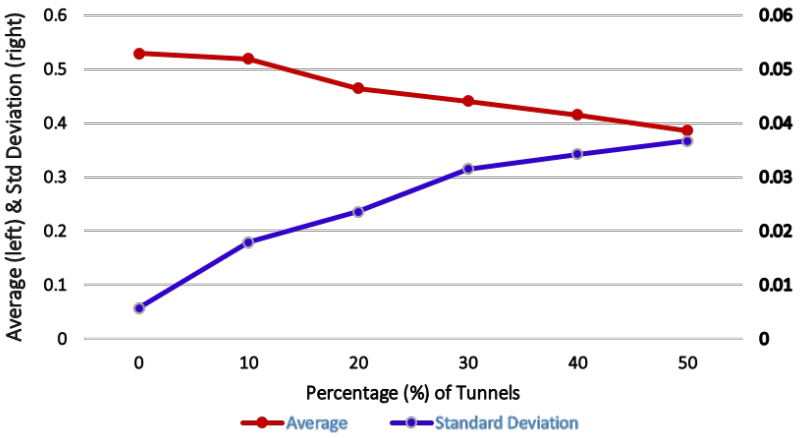


Figure 5.10: Average and standard deviation of the fitness score for topology 1

As shown in Figure 5.10, even for 10% tunnels present in the topology, the decrease in the fitness value represents the existence of a fitter path although the difference is not very high. This is because of the fact that the tunnels are generated randomly, which may not be suitable for inclusion in the path from source AS12 to destination AS16. Upon adding another 10% tunnels to the existing ones, the value of the fitness score experiences a further decrease of 0.05398. For 9 tunnels in the topology, the average fitness value is 0.087581 smaller than that for no tunnels present in the network. For every increment in the number of tunnels, the average of the fitness score keeps increasing and

finally, for 15 tunnels, the average fitness score is only 0.386493.

The standard deviation increases with more tunnels as for the $S-D$, the tunnels provide a better path. The chances of this depend on their proximity to S and D.

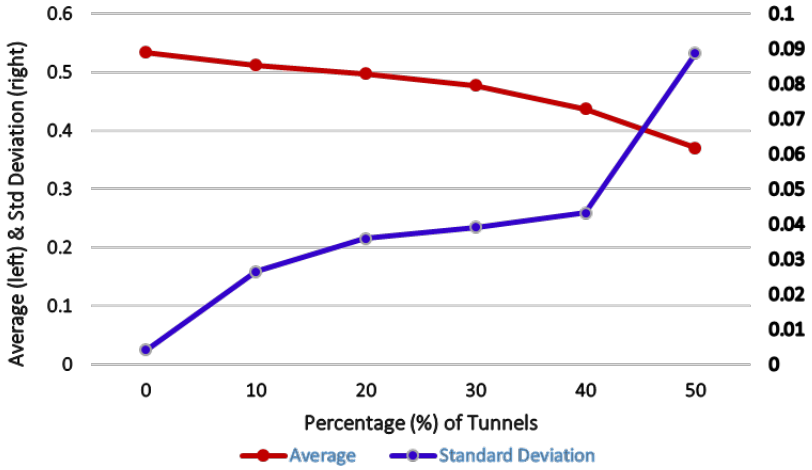


Figure 5.11: Average and standard deviation of the fitness score for topology 2

For topology 2, Figure 5.11 shows that the average fitness score keeps decreasing representing an improvement in the fitness, with the increase in the number of tunnels present. For 10% tunnels present in the topology, the fitness value is approximately 0.51. Then the graph line shows slow decrease in average delay fitness for 20% and 30% tunnels. This behaviour is because of the fact that although new tunnel(s) is/ are being added to the topology, they will not necessarily be used by the end user. It is noteworthy that the fitness value of the least cost path for 50% tunnels is much smaller which means the tunnels provide much benefit in terms of average end-to-end delay.

Figure 5.12 shows that for topology 3, there is fitter path for the continuous increase in the number of tunnels present. The slope in the graph shows that the fitness score experiences slow decrease in this topology, i.e., the random tunnels are not always chosen. One reason is that the tunnel may not always be in the ASes which are traversed through by the data. So, for a specific occurrence of sending a traffic over the network, the use of tunnels will not always yield noticeable benefits.

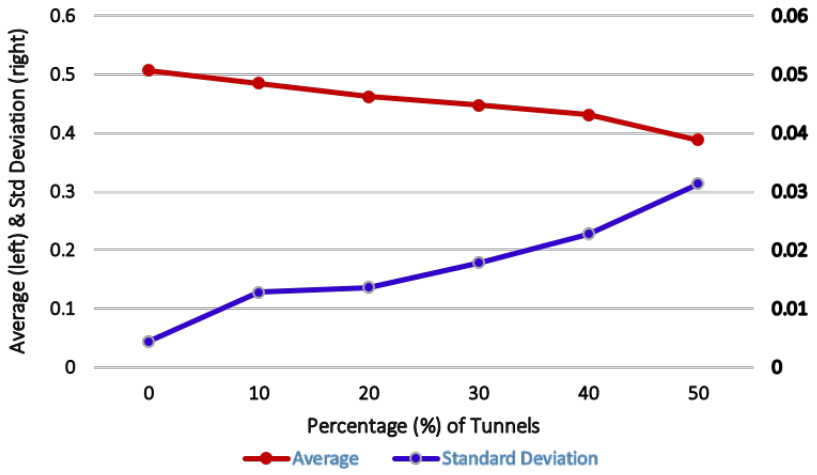


Figure 5.12: Average and standard deviation of the fitness score for topology 3

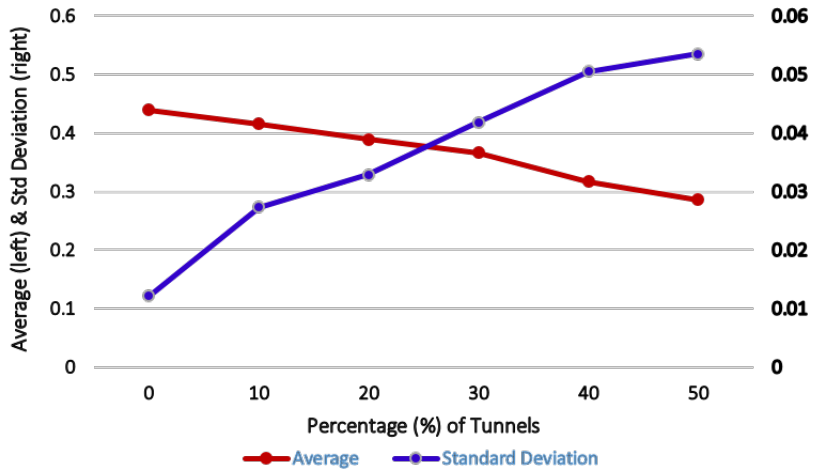


Figure 5.13: Average and standard deviation of the fitness score for topology 4

In Figure 5.13, the red line showing the average fitness becomes approximately 0.29 starting from 0.44. This confirms that the PCAT–I finds fitter paths at the end of the iterations for more tunnels being introduced in the network topology 4. Even for only 3 tunnels in the topology, the improvement is noteworthy, which says that the end user can experience significantly less average end-to-end delay by choosing to use one or more tunnel(s).

For topology 5, Figure 5.14, for the first case of 10% tunnels, the average fitness score decreases by a notable degree. Then it becomes a little slower for the next three sets of increments showing slower improvement in the fitness of the least cost paths. Again adding 10% more tunnels to the existing 40% tunnels shows that there is a much fitter path in terms of end-to-end delay while paying financial cost for tunnel usage.

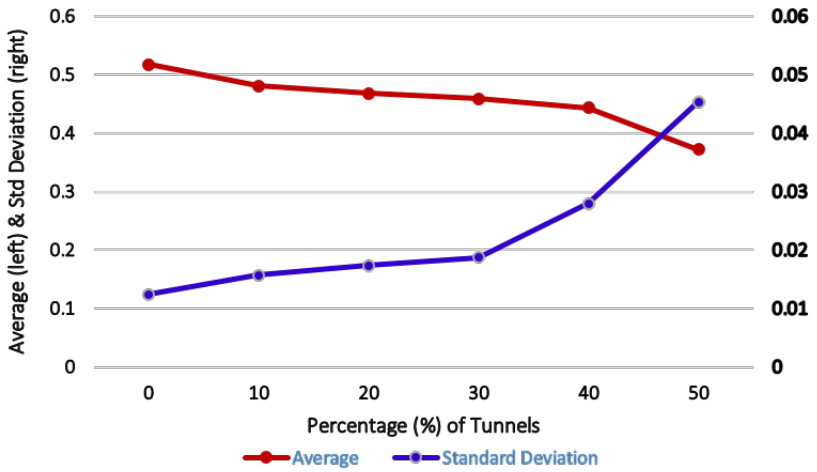


Figure 5.14: Average and standard deviation of the fitness score for topology 5

Finally, it can be observed from the graphs that the presence of tunnels introduces fitter paths for a source-destination pair while the path chosen does not cost a lot of money for the end user. However, relatively small improvements are not surprising, as some existing tunnels are avoided deliberately to keep the financial cost associated with the total path low.

5.3.2 Results for Different Node Degree

Using the same initial topology of 7 ASes that is used for the topologies used above, another 30-AS topology is generated. Keeping other properties as they were, only the average node degree has been changed, which is 2 now. The topology is included in Appendix B.

The ratio of delay for tunnel links and no-tunnel links is 4:1 and that of the financial cost for tunnel and no-tunnel links are 5:1. With the same set of experiment as before, the PCAT–I is run 10 times and then

the average and standard deviation of the fitness score are plotted in the graph of Figure 5.15.

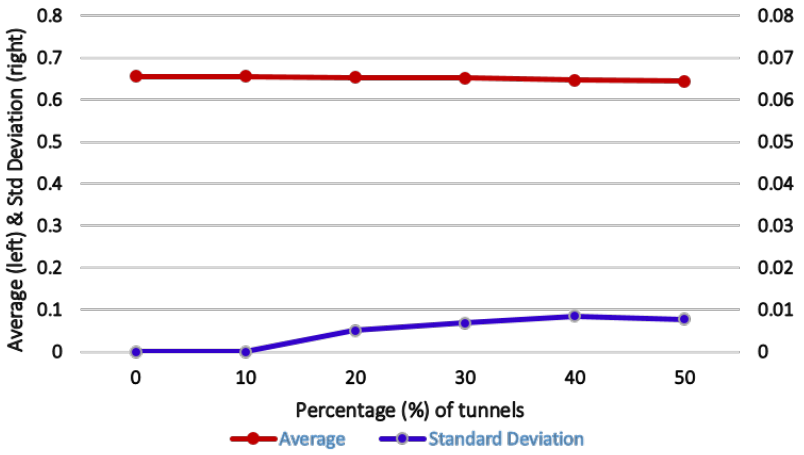


Figure 5.15: Average and standard deviation of the fitness score for topology of average node degree 2

From Figure 5.15 it is observed that for such low node degree, there is very less chance of finding noteworthy improvement in the least cost path. For 3 tunnels present in the 30-AS topology, the average fitness value is same as the one for no tunnels present. The standard deviation is also 0, which implies that in a less connected graph, with a small number of tunnels introduced, there is no variation in the least cost paths since there are very few possible paths in a less connected graph. However, the presence of 50% tunnels shows some decrease, though this is marginal.

Table 5.12 make it the easier to compare the benefits of tunnels in internet topologies of different node degrees. The observations from topology 1 is used here to compare the data with.

Table 5.12: Average and standard Deviation of the fitness score for two different topologies of different average node degree

Tunnel Percentage	**Avg Node Degree = 2** Avg/Std	**Avg Node Degree = 3.6** Avg/Std
0%	0.655303/ 0	0.528770/ 0.005711
10%	0.655303/ 0	0.518975/ 0.017973
20%	0.653693/ 0.005098	0.464915/ 0.023557
30%	0.652083/ 0.006788	0.441186/ 0.031529
40%	0.647254/ 0.008485	0.415282/ 0.034279
50%	0.644034/ 0.007776	0.386493/ 0.036679

It is clear that the tunnels add more benefits for more connected graphs, and for a less connected graph, the number of possible route to send data over the network is generally less.

5.3.3 Results for Different Weights of the Constraints

This section explains the impact of tunnels on the least cost paths in the network topology observed varying the weights of the objective functions, delay fitness and cost fitness. For this experimental setup, the $S-D$ pair AS12-AS16 is used to send data in a 30-AS topology, Topology 1 (APPENDIX B). This is done by changing the value of α in Equation 5.7 . For each value, the PCAT–I is run for 10 times and the average and standard deviation of the final 10 least cost paths are calculated.

Table 5.13 includes the data of this experiment.

Table 5.13: Average and standard deviation of the fitness score for different values of α

Tunnel Percentage	**α = 0** Avg/Std	**α = 0.25** Avg/Std	**α = 0.50** Avg/Std	**α = 0.75** Avg/Std	**α = 1** Avg/Std
0%	0.71528 0.01464	0.63993 0.01236	0.52877 0.00571	0.45423 0.01210	0.39231 0.02433
10%	0.71528 0.01464	0.61837 0.02825	0.51898 0.01797	0.43945 0.01284	0.31154 0.02979
20%	0.711806 0.014640	0.584890 0.040465	0.464915 0.023557	0.405779 0.023546	0.286538 0.038939
30%	0.71181 0.01677	0.57738 0.04352	0.44119 0.03153	0.38610 0.02561	0.25962 0.05148
40%	0.71181 0.05917	0.53684 0.05671	0.41528 0.03428	0.36873 0.03508	0.225 0.060168
50%	0.71181 0.06399	0.49651 0.07843	0.38649 0.03667	0.32151 0.03991	0.14808 0.05948

For path calculation based solely on the fitness cost, the delay fitness, f_1 is given 0% and cost fitness, f_2 100% weight, making the fitness equation as follows:

$$F = (0 \times f_1) + (1 - 0) \times f_2 = f_2 \tag{5.9}$$

The results using Equation 5.9 are plotted in Figure 5.16. It is observed that when only financial cost is considered to find the fitness score, the change in the average value is almost none. This is due to the fact that even after tunnels being introduced, the paths using the tunnels will have higher financial cost which increases the cost fitness causing an increase in the final fitness score. Hence, the algorithm efficiently avoids such paths and chooses the one with least financial cost involved, i.e., the least cost path contains least tunnels possible. The slight change in the red line showing the average fitness indicates use of tunnel for 20% tunnels. This is done only because it does not contribute to an overall increase in the end-to-end path's financial cost. Further flatness of the line supports the statement. The change in the standard deviation confirms little variance in the $S - D$ paths.

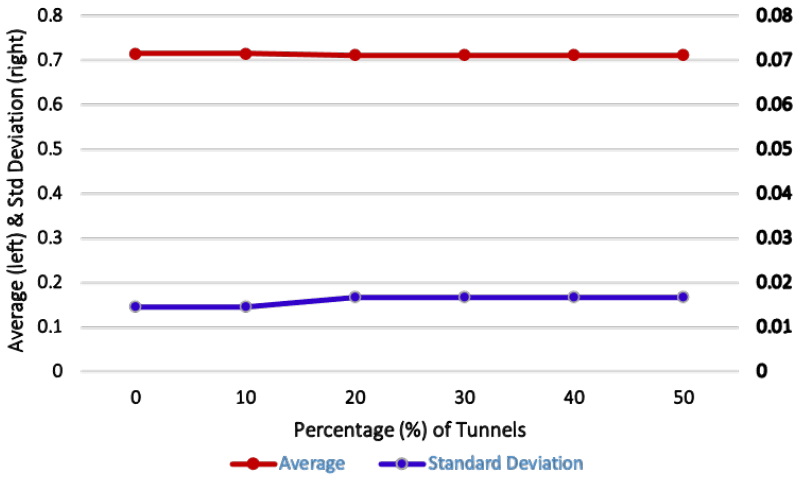


Figure 5.16: Average and standard deviation of the fitness score for $\alpha = 0$

$\alpha = 0.25$ gives 25% weight to the delay fitness and 75% weight to the cost fitness. Equation 5.10 shows the brief calculation method of fitness score, F.

$$F = (0.25 \times f_1) + (0.75 \times f_2) \tag{5.10}$$

Figure 5.17 plots the average and standard deviation of the fitness scores of the least cost paths calculated under this scenario. The average fitness decreases approximately by 0.02 for 3 tunnels present in the 30-AS topology. Implementation of another 3 tunnels adds further improvement in the fitness of the least cost path. For 40% and 50% tunnels, the line indicates a noticeable drop in the average value.

The results plotted in Figure 5.10 already shows the change in the average and standard deviation of the least cost path's fitness value for the 10 runs of each tunnel percentage in the topology used here.

For a ratio of the weight of delay fitness to the weight of cost fitness 3:1, Equation 5.11 represents how the fitness score of the best possible path is calculated.

$$F = (0.75 \times f_1) + (0.25 \times f_2) \tag{5.11}$$

From the graph 5.18, it can be said that there is a consistent decrease in the average fitness value for each implementation of 10% tun-

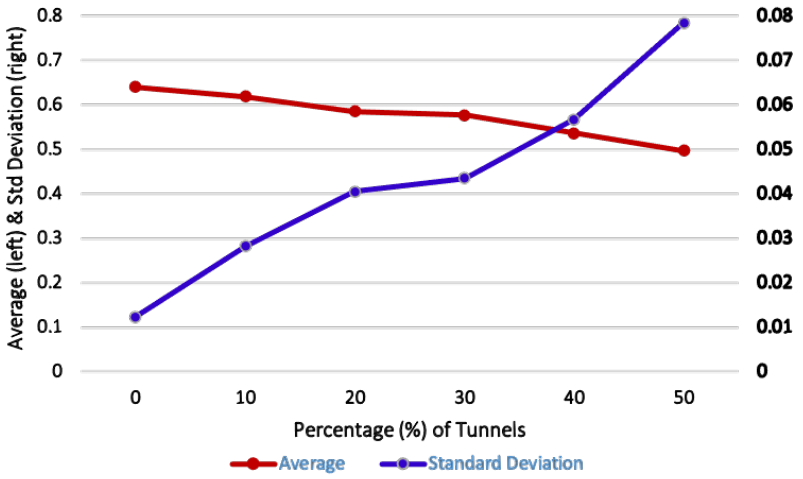


Figure 5.17: Average and standard deviation of the fitness score for $\alpha = 0.25$

nels in the topology. For 9 tunnels present in the network, the value falls to almost 0.4 from 0.45. This is because the use of more tunnels contributes to the reduction of average end-to-end delay. Since the financial cost has only 25% weight, the PCAT–I finds paths with the tunnels giving more priority to the requirement of delay reduction.

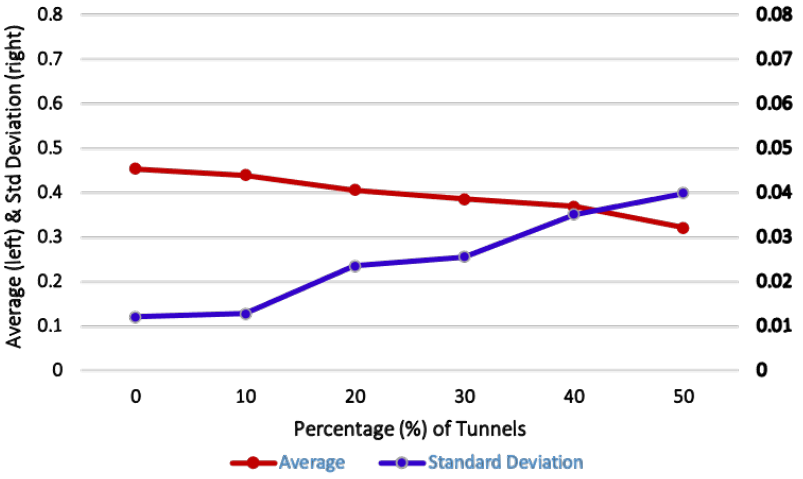


Figure 5.18: Average and standard deviation of the fitness score for $\alpha = 0.75$

Finally, Figure 5.19 plots the values which are calculated considering the delay fitness only. The fitness equation is now:

$$F = (1 \times f_1) + (0 \times f_2) = f_1 \tag{5.12}$$

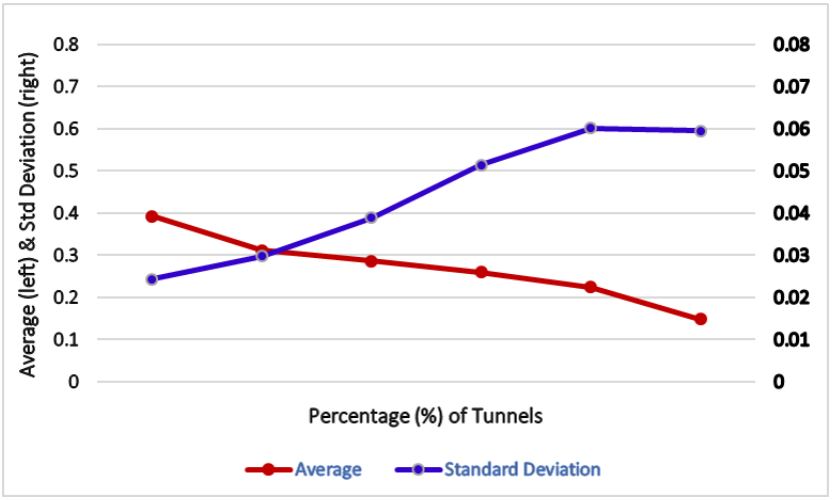


Figure 5.19: Average and standard deviation of the fitness score for $\alpha = 1$

The average fitness score becomes approximately 0.3 for only 10% of tunnels whereas the fitness score of the least cost path is approximately 0.4 for no tunnel present in the topology. For every increase in the number of tunnels, the decrease in value is noticeable. For 50% tunnels, the reduction is more than 2.5 times, which indicates that the PCAT–I finds a 2.5 times fitter path for 50% tunnels present in the network when the financial cost for the tunnel usage is ignored. This graph also shows the continuing increase in the standard deviation. This behaviour is due to the fact that if the financial cost is not taken into account, there are more possible paths for sending data from AS12 to AS16.

Finally, the results described in this section confirm the benefit of tunnel usage in the topology.

5.3.4 Results Considering Peak Time

The experiments in the above sections have been performed using a very conservative estimate of the average delay introduced to each no-tunnel link. The Internet can become very busy during certain times of the

day. This time is usually referred as the peak time, when traffic experiences queuing delay which contributes a lot to the overall degradation of QoE for the end user [188]. In such cases, the tunnels are expected to add more benefit. To prove this, the least cost paths from source AS12 to destination AS16 in the 30-AS topology (Topology 1) is calculated keeping the ratio of financial cost for tunnel and no-tunnel links same as the previous experiments, which is 5:1 and altering the ratio of the average delay for tunnel and no-tunnel links to 1:20 from 1:4. As stated in Section 4.1.4, a data packet typically traverses through 4 to 6 hops within an AS while reaching its destination AS [187]. During peak time, if no-tunnel link is used by the data packet, it will experience 4 to 6 times more delay than the usual time. Hence, the ratio of delay for tunnel usage and no-tunnel link usage is set as 1:20.

This is done for no tunnels present in the network topology as well as 10%, 20%, 30%, 40% and 50% tunnels implemented there. Similar to the previous experiments, the average and standard deviation of the overall fitness score for the best possible path selected by the PCAT–I is calculated after running the simulation for 10 times. Table 5.14 includes the results.

Table 5.14: Average and standard deviation of the fitness score for peak time

Tunnel Percentage	Avg	Std Deviation
0%	0.518406	0.007746
10%	0.509974	0.011159
20%	0.478281	0.024441
30%	0.447430	0.028562
40%	0.413957	0.037868
50%	0.381755	0.038035

The results are plotted in Graph 5.20.

The average fitness value decreases from more approximately 0.52 to 0.48 for 6 tunnels (20%) present in the network topology. This is a large decrease confirming that the additional tunnels introduce better paths. For 30% tunnels, the best possible path's fitness score reduces by almost 40% from that for the 20% tunnels. With the increase in the number of tunnels, the fitness value keeps decreasing indicating the better least cost path. The blue line showing the standard deviation

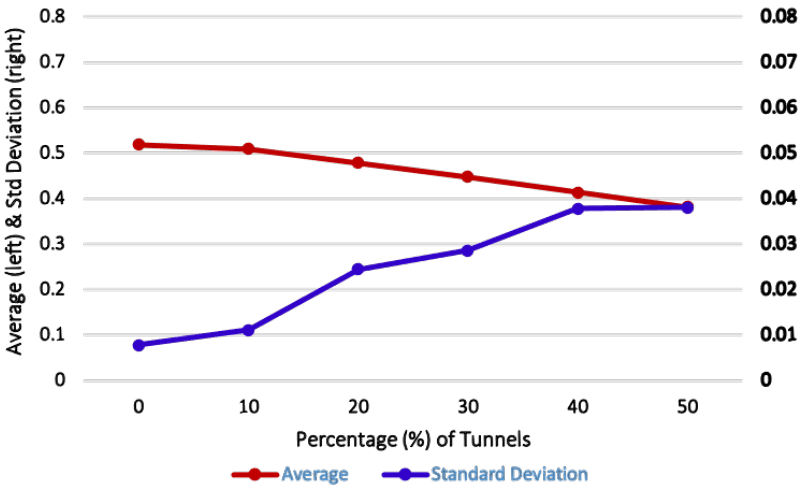


Figure 5.20: Average and standard deviation of the fitness score considering peak time

means that there are more choices of paths when using more tunnels. It increases slowly from 40% to 50%. This behaviour is because of the less variation in the 10 fitness values obtained from the 10 runs of the simulation. However, it is mention-worthy that more than one path can have the same fitness value.

5.4 Summary

In this chapter we confirm that with more tunnels being introduced to a topology, there are fitter least cost paths for a specific pair of source and destination ASes. According to the requirement of the end users, the best path can be calculated by altering the equation of the fitness evaluation for the paths by selecting a suitable alpha value as a compromise between the importance of delay relative to financial cost. The tunnels add more benefit particularly during the peak hours. Instead of random tunnel placements, other methods of can also be explored using PCAT–I.

Chapter 6

Path Computation Algorithm for Tunnels using SPEA (PCAT–II)

The final contribution of this research is the SPEA-based PCA, named as "Path Computation Algorithm for Tunnels–II" (PCAT–II) which efficiently solves a MOOP. Researchers have already proposed the application of SPEA to solve the shortest path routing problem in network topology [209–212]. This supports the choice of algorithm for the PCAT–II. The novelty in our work is the efficient implementation of SPEA to output the paths which use any tunnel only if the usage adds any benefit to the end users in terms of the end-to-end delay experienced along the path and the amount of financial cost to be paid by them.

The next section describes the design and implementation of this path computation tool.

6.1 Design and Implementation

The primary steps of developing the PCAT–II are the same as those of the GA-based PCAT–I. As explained in Sections 5.1.1 and 5.1.2), this also takes an AS topology and generates certain percentage of tunnels in that. The main difference is that after the successful implementation of SPEA, this PCAT–II outputs one or more paths without converting the problem into a SOOP. In other words, a range of suitable paths are identified between a given source-destination pair based on the trade-off

between the optimisation parameters being considered. These solutions might minimise the end-to-end delay whilst minimising the financial cost to be paid by the user, or indicating the delay benefit, if a certain amount of financial expenditure is permitted.

Section 6.1.1 explains how the SPEA is implemented in this research work for path computation.

6.1.1 Implementation of SPEA

The implementation of the SPEA can be explained step by step as follows:

- Step 1:
  - Create initial population, P, based on random generation of chromosomes.
  - Create external archive or empty list, referred to as the external population P'. This population is used to hold the non-dominated solutions found as the algorithm evolves.
- Step 2:
  - Combine P with P' to create super population, PP.
  - Purge P' (empty list).
- Step 3: For each chromosome, calculate:
  1. Delay fitness, f_1 and
  2. Cost fitness, f_2.
- Step 4: Perform dominance ranking.
- Step 5:
  - Copy all non-dominated solutions in PP to P'.
  - Prune bottom ranked x chromosomes in PP (kill).
  - If the maximum number of iterations has been reached, then go to Step 9 or continue to step 6.
- Step 6:
  - Apply genetic operators on Pruned PP. Pruned PP acts as the mating pool.

  - Enter the children chromosomes as the new P.

- Step 7: Add $x\%$ random chromosomes to new P. This ensures "genetic diversity" is maintained.
- Step 8:
  - Increment the iteration count.
  - Go to step 1.
- Step 9:
  - Print the final chromosomes (paths) in P'.
  - Terminate the algorithm.

Here, the steps are explained explained further.

Initialise Paths

A number of possible paths between a source AS and a destination AS form the initial population for the algorithm. This process is same as the one for GA, as explained in Section 5.1.4.

During the same step, a list is created to store the non-dominated paths, which is empty at the beginning.

Evaluate Paths

As described previously, the two objective functions are: delay fitness, f_1 and cost fitness, f_2. The calculation of f_1 and f_2 are done using 5.4 and 5.6, as explained in Section 5.1.4. Smaller values of f_1 and f_2 mean higher delay fitness and higher cost fitness.

Unlike the GA, here the paths are evaluated based on dominance score, δ. To do so, for each path, P_i, the f_1 and f_2 values are compared with the f_1 and f_2 values of other paths, P_j in the list of the super population, PP. P_i dominates P_j if one of the fitness values is better and the other is definitely NOT worse, i.e., if at least one of f_1 and f_2 associated with P_i is smaller than f_1 and f_2 of other P_j AND at the same time the another of f_1 and f_2 associated with P_i is definitely not bigger than f_1 and f_2 then P_i gets a dominance count, Δ of 1. The equation 6.1 represents this process.

$$\Delta_{P_i} = \begin{cases} 1 & \text{if } f_1(P_i) < f_1(P_j) \text{ AND } f_2(P_i) \leq f_2(P_j)) \\ 1 & \text{if } f_1(P_i) \leq f_1(P_j) \text{ AND } f_2(P_i) < f_2(P_j)) \\ 0 & \text{otherwise} \end{cases} \tag{6.1}$$

Every time the value of Δ is 1, the dominance count of P_i gets incremented. Hence,

$$\Delta(P_i) = P_j^n \tag{6.2}$$

where P_j^n is the total number P_j dominated by P_i.
Then dominance score of P_i, δ is calculated using Equation 6.3.

$$\delta = \frac{\Delta}{1 + PP_n} \tag{6.3}$$

Get Non-dominated Paths

The next step is to find the non-dominated paths. There are two cases when a path can be non-dominated.

- Dominating condition: If at least one of the fitness values is better and the other is definitely NOT worse.
- Indifferent condition:
  - If at least one of the fitness values is better and the other is worse.
  - If both fitness values are equal

Assuming that there is a flag value, γ for a path P_i to be non-dominated, the non dominance is calculated as follows:

$$\gamma_{P_i} = \begin{cases} 1 & \text{if } f_1(P_i) < f_1(P_j) \text{ AND } f_2(P_i) \leq f_2(P_j)) \\ 1 & \text{if } f_1(P_i) \leq f_1(P_j) \text{ AND } f_2(P_i) < f_2(P_j)) \\ 1 & \text{if } f_1(P_i) < f_1(P_j) \text{ AND } f_2(P_i) > f_2(P_j)) \\ 1 & \text{if } f_1(P_i) > f_1(P_j) \text{ AND } f_2(P_i) < f_2(P_j)) \\ 1 & \text{if } f_1(P_i) == f_1(P_j) \text{ AND } f_2(P_i) == f_2(P_j)) \\ 0 & \text{otherwise} \end{cases} \tag{6.4}$$

The non-dominated paths are then copied to the P'.

Reproduction

The paths are ranked according to the descending dominance score. The top 80% paths are selected for mating and the remaining 20% at

the bottom of the list are killed off. This ensures that the fitter paths are chosen for further genetic operations and at the same time the non-dominated paths definitely take part in the further reproduction steps.

The selected paths then potentially undergo crossover and mutation. The processes are same as described in Section 5.1.4 for the PCAT–I.

Inject Random Paths

To ensure the diversity of solutions, a number of random possible paths for the same pair of source and destination ASes are generated. These paths are obtained following the same steps as for initialisation. These paths are addressed as "secondary paths".

Finally, a set of paths is sent it to the next generation. This includes the external path, resultant potentially crossovered and mutated paths and the secondary paths are merged to create the new set.

External Population

The non-dominated solutions in P' are never deleted. But in every iteration new non-dominated ones are copied to P'. After doing so, it is made sure that no duplicate paths exist and the dominated paths from P' are removed. The final paths in P' are the Pareto optimal solutions that take part in forming the Pareto front.

Terminating Condition

The terminating condition for this algorithm is the maximum number of iterations for the SPEA. Once it reaches the maximum iteration, i_{max}, the final set of paths present in P' becomes the final output. The two fitness values of these paths f_1 and f_2 make the desired Pareto front.

6.1.2 Flowchart

Figure 6.1 represents the high level flowchart of the SPEA-based Path Computation Tool.

For a better understanding of the tool, the steps can be explained briefly:

- A PFP-generated AS-level topology is provided as an input to the PCAT–II.
- From the data in the text file, the tool finds every pair of source and destination AS.

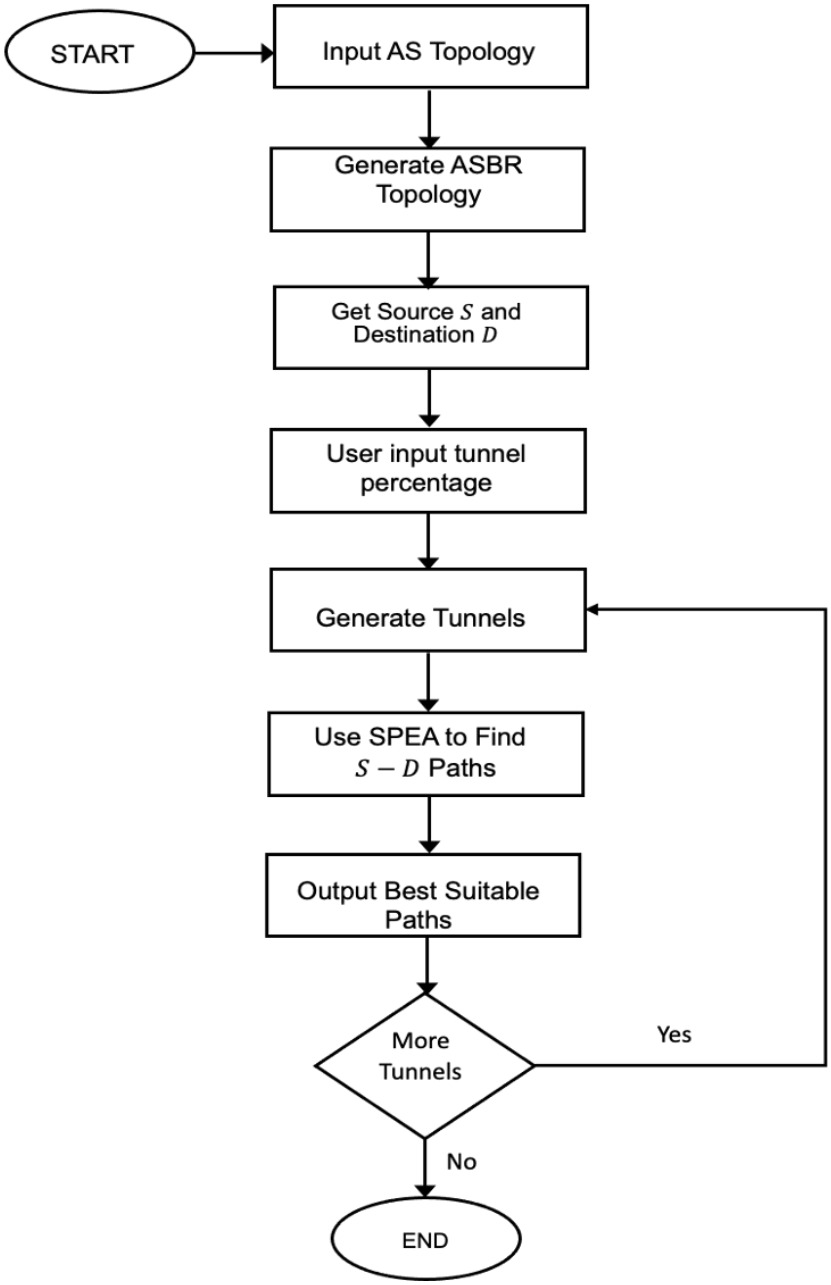


Figure 6.1: Flowchart of PCAT–II

- From the source-destination pairs, ASBR connections are gathered.
- Next, the tool takes the expected percentage of tunnels (0%-99%) as input, to be present in the topology.

- It generates the exact number of tunnels randomly placed in different ASes and creates a text file containing this information.
- Then the tool implements SPEA to find the best suitable path(s).
- It repeats the process of taking user input for generating more tunnels until the user does not want any more of the tunnels. New tunnels are added to the existing ones.
- The best path(s) for the expected percentage of tunnel(s) is(are) then presented to the user.

The implementation of the SPEA itself can be represented with a flowchart as presented in Figure 6.2.

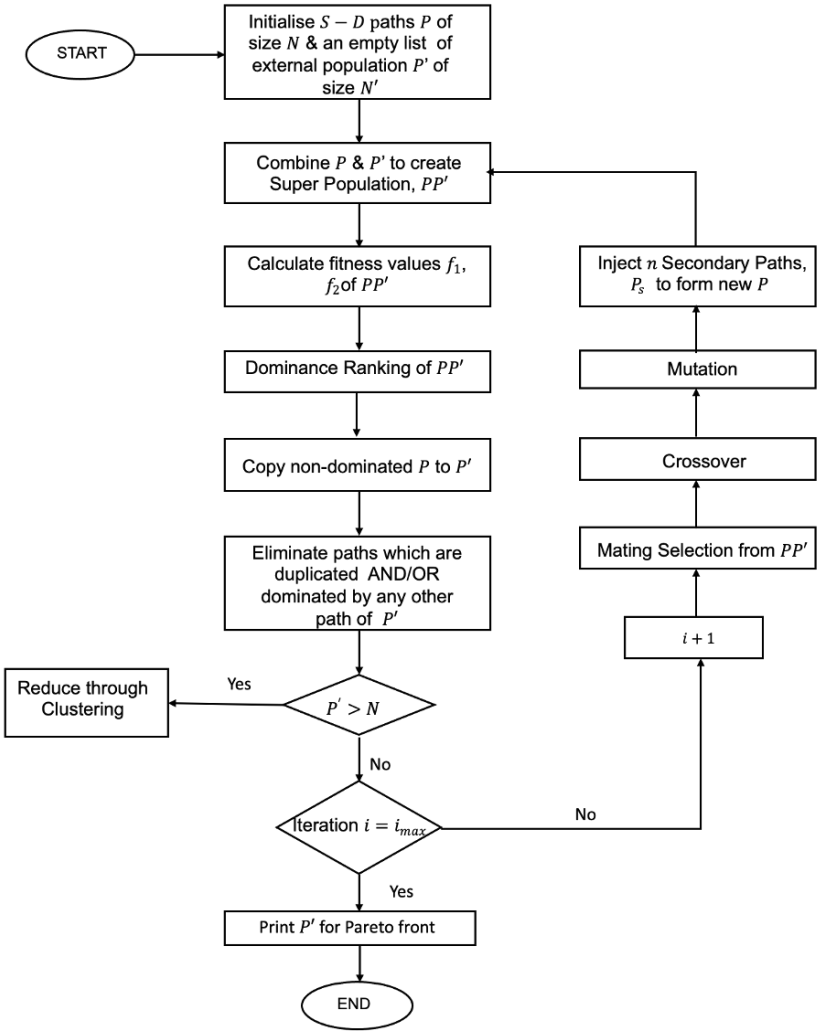


Figure 6.2: Flowchart showing the steps of SPEA implemented for path computation

6.1.3 Pseudo Code

This section presents a pseudo code of the PCAT–II programmed implementing SPEA (in Python).

Algorithm 7 PCAT-II

INPUT: *Graph,* $G(V, E)$; *Source,* S; *Destination,* D;
Iteration limit, i_{max}; *Tunnel Percentage,* β
Delay for tunnel link, d_t; *Cost for tunnel link,* c_t
Delay for no-tunnel (intra-AS) link, d_{nt}
Cost for no-tunnel (intra-AS) link, c_{nt}
Delay for Inter-AS link, d_{as}; *Cost for Inter-AS link,* c_{as}
Crossover probability, ρ_c; *Mutation Probability,* ρ_m
OUTPUT: *Pareto Optimal Paths,* P'

while $itr \leq i_{max}$ **do**
  $T \leftarrow$ Tunnels to generate in random ASes
  $N_T \leftarrow \frac{\beta * N}{100}$ Number of tunnelled ASes
  if $\beta > 0$ **then**
    Generate tunnels in N_t ASes
  else
    Do not generate tunnels
  end if
  $P \leftarrow$ Randomly generated Initial paths //Algorithm 3
  $N \leftarrow$ Size of P
  $P' \leftarrow$ External archive for Pareto optimal paths
  $N' \leftarrow$ Size of P'
  $PP \leftarrow$ Super Population $(P + P')$
  $PP_n \leftarrow$ Size of PP
  $f_1, f_2 \leftarrow$ Delay fitness, Cost Fitness //Algorithm 4
  $\delta \leftarrow$ Dominance Score //Algorithm 8
  Sort PP in descending order of δ
  Find non-dominated solutions //Algorithm 9
  Copy non-dominated solutions to P'
  if $itr = 1$ **then**
    $P \leftarrow P' \cup M \cup P_s$, Final set of Population for next iteration
    $itr \leftarrow (itr + 1)$
  else
    $N_{mp} \leftarrow$ Mating pool
    $N_{mp} \leftarrow \frac{80\% * N}{100}$, Size of mating pool
    $C \leftarrow$ Paths after Crossover(P, N_{mp}, ρ_c) //Algorithm 5
    $M \leftarrow$ Paths after Mutation (C, N_{mp}, ρ_m) //Algorithm 6
    $P_s \leftarrow$ Secondary paths // Using Algorithm 3
    $N_{P_s} \leftarrow (\frac{20\% * N}{100} - 1)$, Size of Secondary population
    $P \leftarrow P' \cup M \cup P_s$, b Final set of Population for next iteration
    $itr \leftarrow (itr + 1)$
  end if
end while
return P'

Algorithm 8 Evaluate with Dominance

INPUT: PP, f_1, f_2
OUTPUT: δ

```
Δ ← 0, Dominance count of a path
while i < length(PP) do
  Compare P_i with the rest in PP, P_j
  for j ← 1 to length(P_j) do
    if f_1(P_i) < f_1(P_j) AND f_2(P_i) ≤ f_2(P_j)) then
      Δ ← Δ + 1
    else if f_1(P_i) ≤ f_1(P_j) AND f_2(P_i) < f_2(P_j)) then
      Δ ← Δ + 1
    else
      Δ ← Δ + 0
    end if
    δ = Δ / (1 + PP_n)
  end for
end while
return δ
```

Algorithm 9 Pareto Optimality

INPUT: PP, f_1, f_2
OUTPUT: P'

```
γ ← 0, Flag for non-dominated path
while i < length(PP) do
  Compare P_i with the rest in PP, P_j
  for j ← 1 to length(P_j) do
    if f_1(P_i) < f_1(P_j) AND f_2(P_i) ≤ f_2(P_j) then
      γ ← 1
    else if f_1(P_i) ≤ f_1(P_j) AND f_2(P_i) < f_2(P_j) then
      γ ← 1
    else if f_1(P_i) < f_1(P_j) AND f_2(P_i) > f_2(P_j) then
      γ ← 1
    else if f_1(P_i) > f_1(P_j) AND f_2(P_i) < f_2(P_j) then
      γ ← 1
    else if f_1(P_i) == f_1(P_j) AND f_2(P_i) == f_2(P_j) then
      γ ← 1
    else
      γ ← 0
    end if
    For γ == 1, P_i is non-dominated
    Send P_i to P'
  end for
end while
return P'
```

6.2 Validation

The main genetic operations involved in the SPEA are not different from the GA. Hence the PCAT–I designed with GA is further modified to implement SPEA.

Since the main difference is in evaluation of the paths, this section initially provides the validation of the process.

Fitness Evaluation

To start with, the tool is provided with the paths presented in Table 5.4, with the fitness values shown in Table 5.6. Figure 6.3 shows the input file.

```
src,dest,ASBRpath,delay,cost,delay_fitness,cost_fitness
2,7,2_1>1_2>1_5>5_1>5_4>4_5>4_7>7_4,16,7,0.592592593,0.583333333
2,7,2_1>1_2>1_4>4_1>4_7>7_4,11,5,0.407407407,0.5
2,7,2_3>3_2>3_5>5_3>5_1>1_5>1_4>4_1>4_7>7_4,21,9,0.777777778,0.642857143
2,7,2_3>3_2>3_5>5_3>5_4>4_5>4_7>7_4,16,7,0.592592593,0.583333333
2,7,2_5>5_2>5_1>1_5>1_4>4_1>4_7>7_4,16,7,0.592592593,0.583333333
2,7,2_5>5_2>5_4>4_5>4_7>7_4,11,5,0.407407407,0.5
2,7,2_6>6_2>6_3>3_6>3_5>5_3>5_1>1_5>1_4>4_1>4_7>7_4,26,11,0.962962963,0.6875
2,7,2_6>6_2>6_3>3_6>3_5>5_3>5_4>4_5>4_7>7_4,21,9,0.777777778,0.642857143
```

Figure 6.3: Input paths for fitness validation using SPEA

Unlike the previous tool developed for the GA, this one outputs all the paths with non-dominated fitness values, which are the set of least cost paths. In Figure 6.3, it can be easily observed from the fitness values that two paths are non-dominated. AS presented in Figure 6.4, the output of the PCAT–II matches this expectation.

```
src,dest,ASBRpath,delay,cost,delay_fitness,cost_fitness,fitness_score,dominance_count,dominance_score,is_dominating
2,7,2_1>1_2>1_4>4_1>4_7>7_4,11,5,0.40740740700000005,0.5,0.453703704,6,0.6666666666666666,1
2,7,2_5>5_2>5_4>4_5>4_7>7_4,11,5,0.40740740700000005,0.5,0.453703704,6,0.6666666666666666,1
```

Figure 6.4: Output non-dominated paths for Figure 6.3

For a Larger Topology

To provide input data for a set of possible paths a 30-AS topology is used for the further validation of this tool.

Table 6.1: Fitness values of 30 paths

Path	f_1	f_2	Path	f_1	f_2
P1	0.7692	0.8438	P16	0.6346	0.7917
P2	0.4615	0.8611	P17	0.8654	0.8529
P3	0.7692	0.8438	P18	0.7308	0.8077
P4	0.6538	0.8750	P19	0.6731	0.8333
P5	0.5962	0.7222	P20	0.4423	0.7500
P6	0.5385	0.7727	P21	0.6731	0.8333
P7	0.5577	0.8684	P22	0.6538	0.8750
P8	0.7115	0.8611	P23	0.7308	0.8077
P9	0.6731	0.8333	P24	0.6346	0.7917
P10	0.5192	0.8438	P25	0.4423	0.7500
P11	0.6346	0.7917	P26	0.6346	0.7917
P12	0.6154	0.8529	P27	0.7692	0.8438
P13	0.5769	0.8214	P28	0.4615	0.8611
P14	0.8846	0.7917	P29	0.7308	0.8077
P15	0.6346	0.7917	P30	0.7692	0.8438

Table 6.1 has the values of the two objective functions, f_1 and f_2 of 30 initial paths.

From the fitness values, using Equation 6.2, for each chromosome, the PCAT–II finds how many other paths it dominates. Then the dominance score is calculated using Equation 6.3. A flag value γ is assigned to see whether the path is dominated by any other path. It is 1 as long as the path is not dominated and is changed to 0 if another path dominates it using the logic explained in Equation 6.4. Then the paths are sorted according to the dominance score.

Table 6.2 shows the output data.

The data set matches with the values calculated manually using the referred equations, confirming the process is operating satisfactorily.

Table 6.2: Paths sorted from Table 6.1

f_1	f_2	Δ	δ	γ	$f1$	f_2	Δ	δ	γ
0.4423	0.7500	27	0.8710	1	0.7308	0.8077	5	0.1613	0
0.4423	0.7500	27	0.8710	1	0.7308	0.8077	5	0.1613	0
0.5385	0.7727	23	0.7419	0	0.4615	0.8611	4	0.1290	0
0.5962	0.7222	21	0.6774	1	0.4615	0.8611	4	0.1290	0
0.6346	0.7917	15	0.4839	0	0.6154	0.8529	4	0.1290	0
0.6346	0.7917	15	0.4839	0	0.5577	0.8684	2	0.0645	0
0.6346	0.7917	15	0.4839	0	0.7692	0.8438	1	0.0323	0
0.6346	0.7917	15	0.4839	0	0.7692	0.8438	1	0.0323	0
0.6346	0.7917	15	0.4839	0	0.7692	0.8438	1	0.0323	0
0.5769	0.8214	12	0.3871	0	0.7692	0.8438	1	0.0323	0
0.5192	0.8438	10	0.3226	0	0.6538	0.8750	0	0	0
0.6731	0.8333	6	0.1935	0	0.8654	0.8529	0	0	0
0.6731	0.8333	6	0.1935	0	0.8846	0.7917	0	0	0
0.6731	0.8333	6	0.1935	0	0.7115	0.8611	0	0	0
0.7308	0.8077	5	0.1613	0	0.6538	0.8750	0	0	0

It is clearly seen from Table 6.2 that there are three paths with $\gamma = 1$, which are paths are not dominated by any other path.

Figure 6.5 shows the output non-dominated paths from the tool.

```
src,dest,ASBRpath,delay,cost,delay_fitness,cost_fitness,number_of_tunnels_used,tunnels_used ,dominance_count,dominance_score,is_dominating
12,16,12_18>18_12>18_3>3_18>3_15>15_3>15_4>4_15>4_5>5_4>5_16>16_5,23,15,0.44230769230800004,0.75,1,5,27,0.87096774193548 39,1
12,16,12_3>3_12>3_15>15_3>15_4>4_15>4_7>7_4>7_2>2_7>2_5>5_2>5_16>16_5,23,15,0.44230769230800004,0.75,1,3,27,0.87096774193548 39,1
12,16,12_2>2_12>2_7>7_2>7_4>4_7>4_1>1_4>1_3>3_1>3_5>5_3>5_16>16_5,31,13,0.596153846154,0.7222222222222,0,0,21,0.67741935483870 96,1
```

Figure 6.5: Output non-dominated paths

As none of the non-dominated paths are duplicated, these are the final Pareto optimal solutions. The objective functions i.e., the fitness values of all the are paths are plotted in Figure 6.6.

The points marked with red colour confirm that no other points plotted in the graph can be fitter than these two.

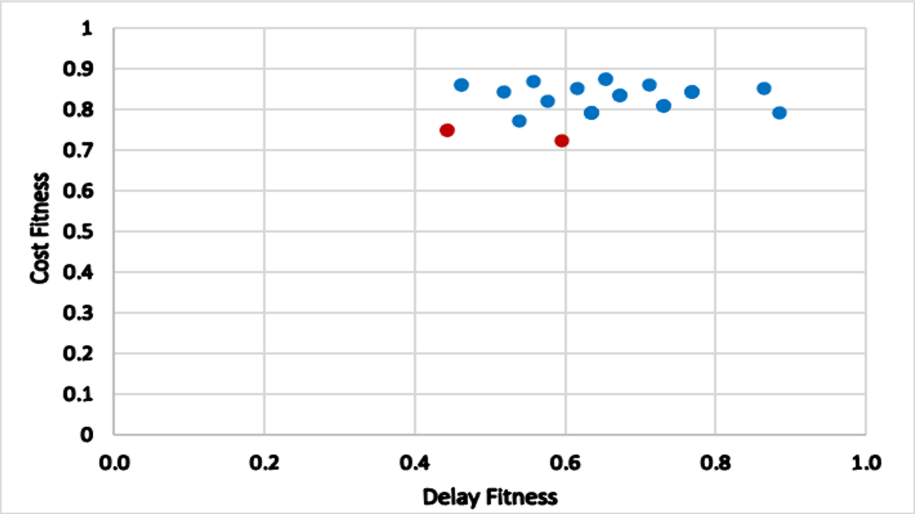


Figure 6.6: Graph showing Pareto optimal solutions

6.3 Results and Evaluation

This section includes the observation and comments on the results obtained from the PCAT–II. For the experiments performed to obtain the results described in this section, unless and otherwise specified, the followings are considered:

- Initial population size, N is 30. The reason to choose this size is explained in Section 5.1.4.
- Size of external population, P' is limited to 30.
- The ratio of the average end-to-end delay of an intra-domain tunnel to that of a no-tunnel link is used as 1:4 and the ratio of financial cost for using a tunnel along the path to that of a no-tunnel path is set as 5:1, except for the peak time.

Table 6.3 includes the parameters used for the PCAT–II.
The parameters used for implementing SPEA are shown in Table 6.4.

Table 6.3: Parameters used in the PCAT–II

Parameter	Value
Delay cost for tunnel	1
Delay cost for no-tunnel link	4
Delay cost for tunnel during peak time	1
Delay cost for tunnel during peak time	20
Delay for inter-domain link	1
Financial cost for tunnel	5
Financial cost for no-tunnel link	1
Financial for inter-domain link	1

Table 6.4: Parameters used in SPEA

Parameter	Value
Initial population size, N	30
Reproduction percentage	80
Size of mating pool, N_{mp}	$\frac{80\%*N}{100} = 24$
Maximum size of external population, N'	30
Cross-over rate, ρ_c	0.6
Mutation rate, ρ_m	0.02
Size of secondary population, N_{ps}	$(\frac{20\%*N}{100} - 1) = 5$
Maximum number of iterations, i_{max}	30

However, before fixing the crossover probability, ρ_c and mutation probability, ρ_m, for further experiments, a large number of simulations were run varying them. Section 6.3.1 discusses the results for two selected scenarios.

6.3.1 Results for Different Crossover Probability (ρ_c) and Mutation Probability (ρ_m)

A 30-AS topology, referred in the thesis as Topology 1, is used for the experiments here. The maximum number of iterations, i_{max} is set to 30. Then the PCAT–II is run for 20 times to find efficient paths from a provided source(S)-destination pair (D), $12-16$. The random seed is different in each run, leaving the choice to select tunnels to the algorithm.

Each run outputs one or more fit path(s). After every 5 runs, the number of unique paths are noted. This is to see if there is an increase in the number and if so, then how it behaves with the change of ρ_c and ρ_m. Then, from these solutions, all the non-dominated solutions are obtained and used for plotting the Pareto fronts.

This is done for 20% and 40% tunnels.

Figure 6.7 shows the number of unique output paths for 20% tunnels. The red line shows the results obtained using $\rho_c = 0.6$ and $\rho_m = 0.02$. In the first run, there is only one unique path. The number increases to 7 after 5 runs and 10 after 10 runs. The next 5 runs add another 3 paths. After 20 runs, the total number becomes almost double.

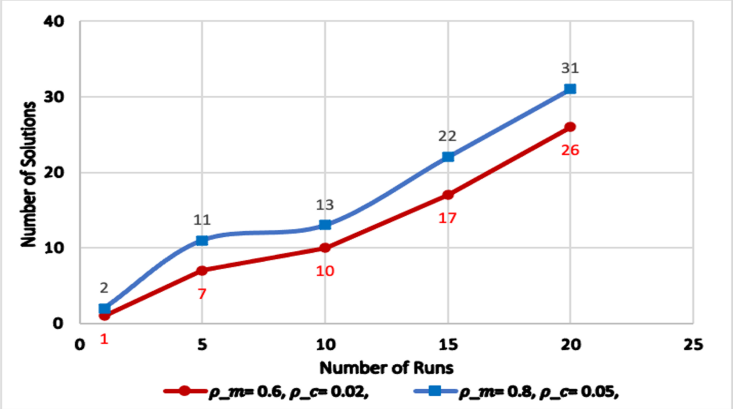


Figure 6.7: Number of total solutions for 20% tunnels using different ρ_c and ρ_m

For $\rho_c = 0.8$ and $\rho_m = 0.05$, the number of solutions after the first run is 2, which increases to 11 after adding the solution from the next 4 runs. Only after another 10 runs, the number of paths become twice as before. At the end of 20 runs, the total number of solutions is 31, which is almost 19% more than the number obtained using $\rho_c = 0.6$ and $\rho_m = 0.02$.

The number of output paths are again counted and plotted in Figure 6.8 for 40% tunnels. The blue line shows a consistent increase in the total number of paths. After every 5 runs, the total number of solution paths shown in the graph is higher than that for using the same number of tunnels just altering the value of ρ_c and ρ_m.

Therefore, it is clear that the when the values of ρ_c and ρ_m are preset as 0.6 and 0.02, the number of total solutions are more, indicating the PCAT–II to be more efficient.

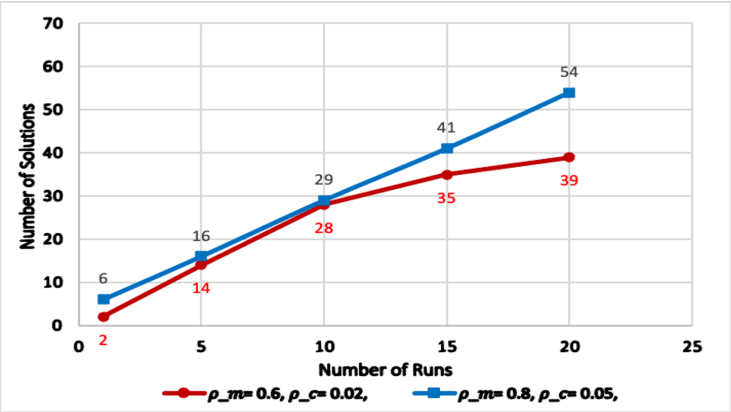


Figure 6.8: Number of total solutions for 40% tunnels using different ρ_c and ρ_m

Pareto Fronts of the Solutions

Next, the Pareto dominance for all the paths are evaluated and only the non-dominated ones are stored to form the Pareto front. This is done after the 1^{st}, 10^{th}, 20^{th} runs. Note that, the number of points can be less than the number of the paths having the set of values presented by it, since more than one paths can have the same fitness values. This is due to the fact that each cost of each tunnel is same. Also, all intra-AS no-tunnel link has the same cost metrics associated with them and the case for the inter-AS links are not any different. Hence, based on the number of links, the fitness values are calculated, which are not varied depending on which links are being used. Figure 6.4 has an example of two different paths with the same fitness values.

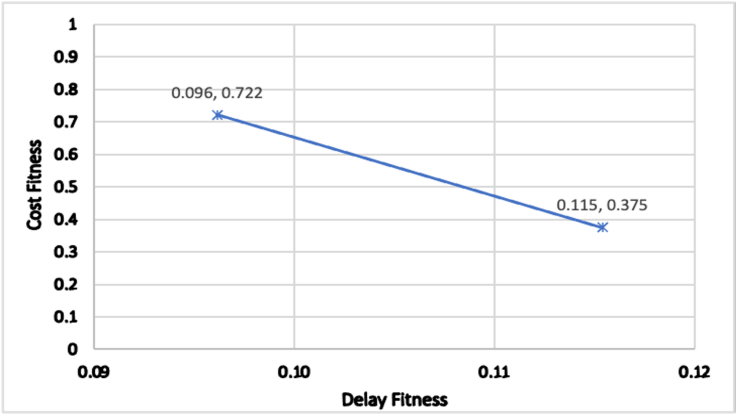
For 20% Tunnels

Figures 6.9 and 6.10 show Pareto fronts for 20% tunnels for the two different set of crossover and mutation probabilities.

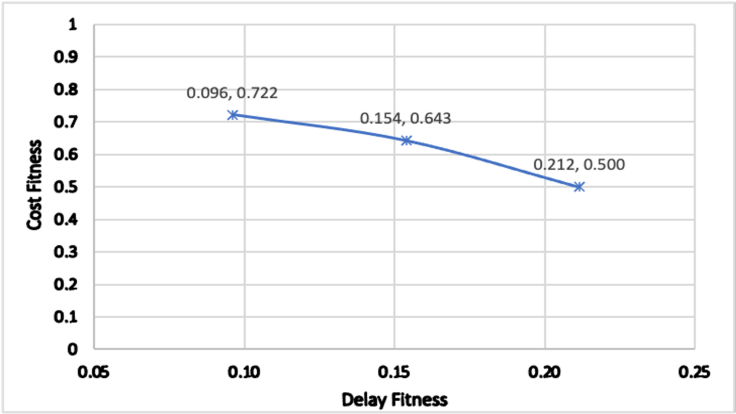
The observation from Figure 6.10a is important as there is only one point in the Pareto Front, but it represents 2 different optimal solutions with same fitness value.

For 40% Tunnels

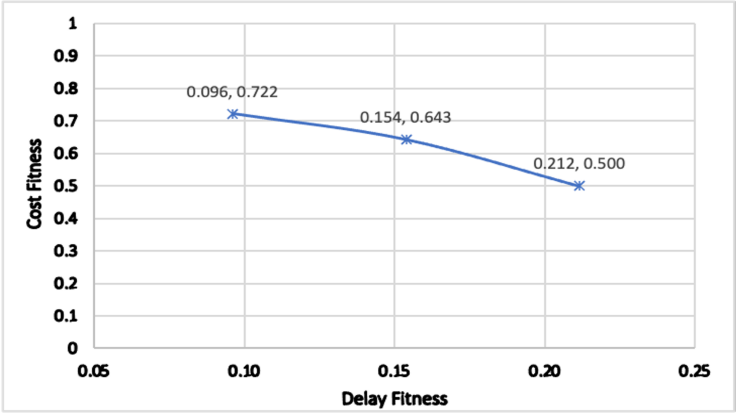
Similarly, the Pareto fronts for 40% tunnels are shown in Figures 6.11 and 6.12.



(a)

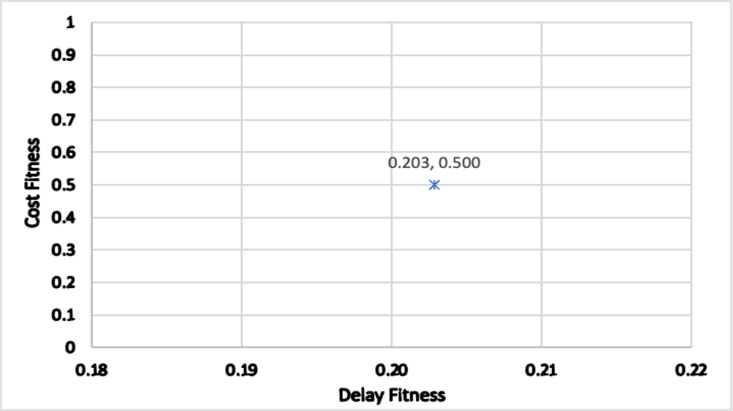


(b)

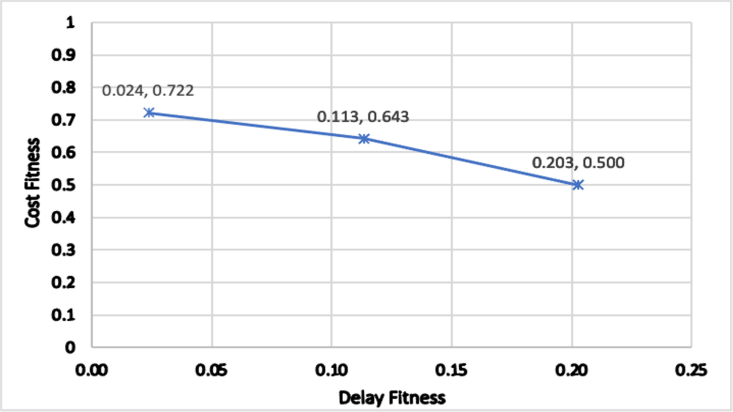


(c)

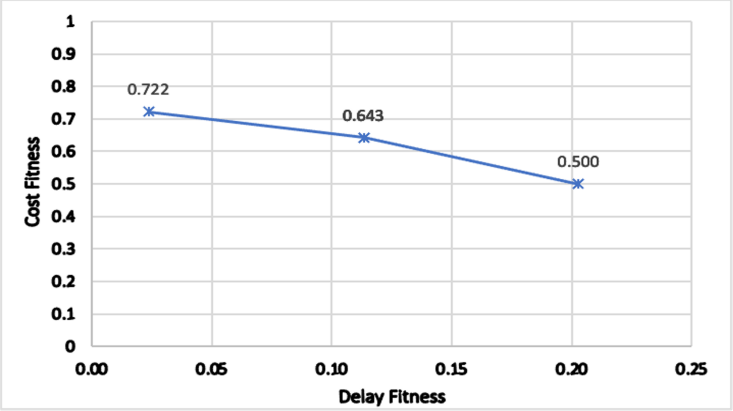
Figure 6.9: Pareto fronts after: (a) 1 run; (b) 10 runs; (c) 20 runs (for 20% tunnels when $\rho_c = 0.6$ and $\rho_m = 0.02$)



(a)

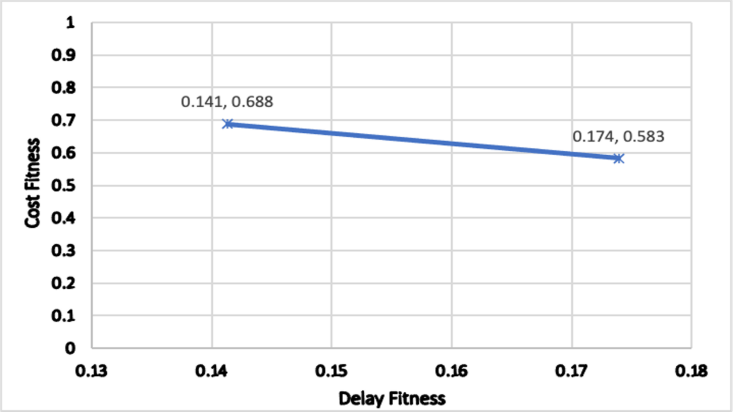


(b)

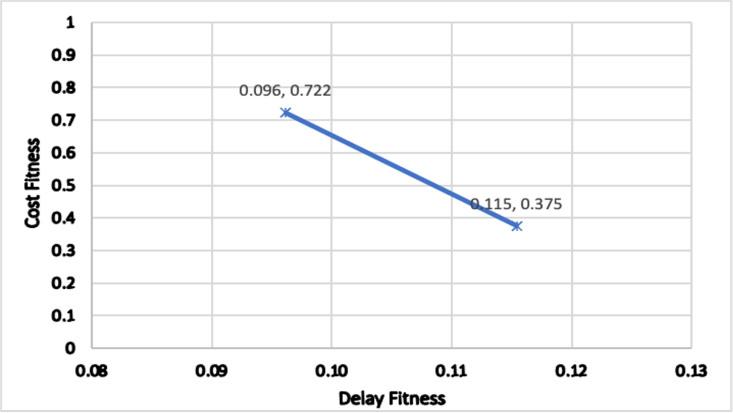


(c)

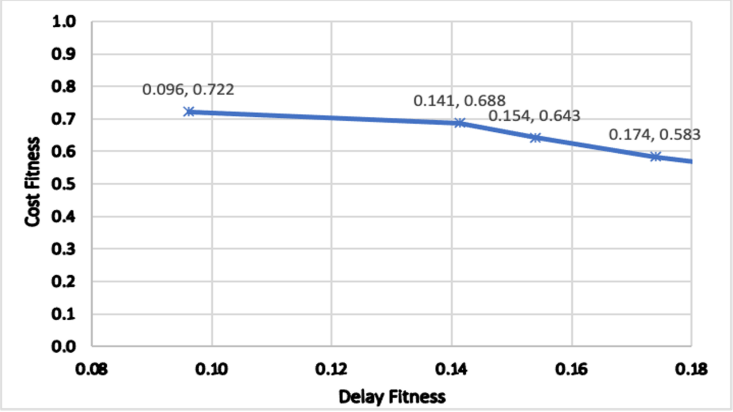
Figure 6.10: Pareto fronts after: (a) 1 run; (b) 10 runs; (c) 20 runs (for 20% tunnels when $\rho_c = 0.8$ and $\rho_m = 0.05$)



(a)

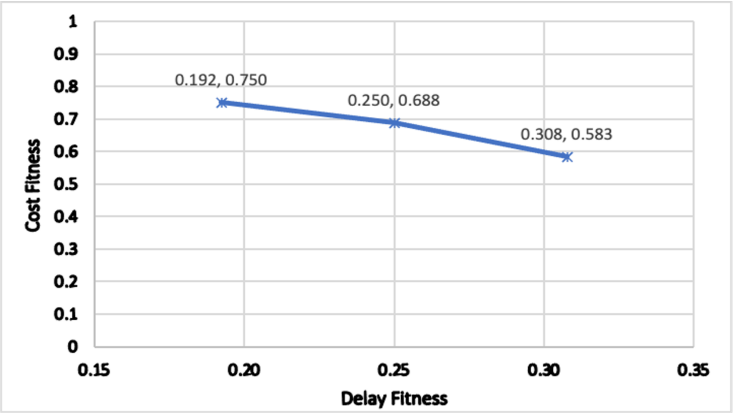


(b)

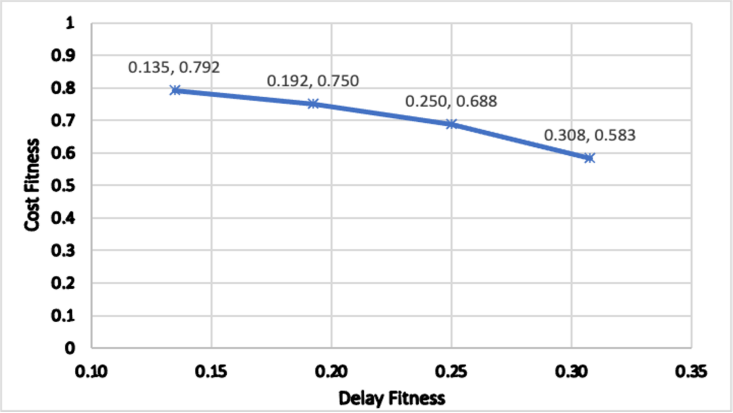


(c)

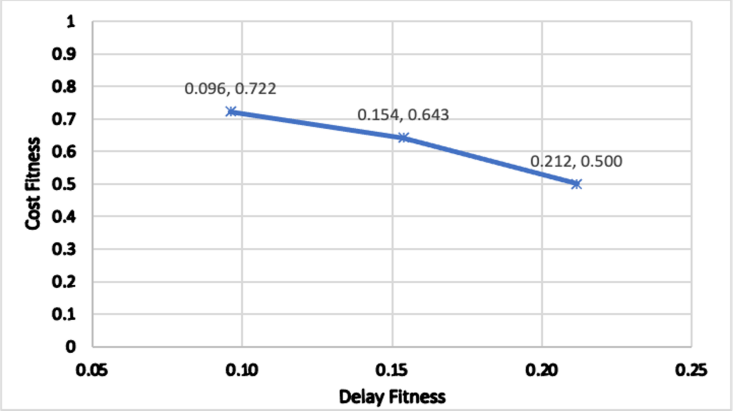
Figure 6.11: Pareto fronts after: (a) 1 run; (b) 10 runs; (c) 20 runs (For 40% tunnels when $\rho_c = 0.6$ and $\rho_m = 0.02$)



(a)



(b)



(c)

Figure 6.12: Pareto fronts after: (a) 1 run; (b) 10 runs; (c) 20 runs (For 40% tunnels when $\rho_c = 0.8$ and $\rho_m = 0.05$)

The number of Pareto optimal solutions in different figures are included in Table 6.5.

Table 6.5: Total number of Pareto optimal paths for different values of crossover probability ρ_c and mutation probability ρ_m

20%				40%			
ρ_c=0.6, ρ_m=0.02		ρ_c=0.8, ρ_m=0.05		ρ_c=0.6, ρ_m=0.02		ρ_c=0.8, ρ_m=0.05	
Fig.	#Sol.	Fig.	#Sol.	Fig.	#Sol.	Fig.	#Sol.
6.9a	2	6.10a	2	6.11a	2	6.12a	3
6.9b	2	6.10b	6	6.11b	4	6.12b	8
6.9c	4	6.10c	10	6.11c	9	6.12c	11

After observing the results from this section, the rest of the experiments have been performed considering the crossover probability, $\rho_c = 0.8$ and mutation probability, $\rho_m = 0.05$.

6.3.2 Results for Different Topologies

Five different topologies (referred as Topology 1 to 5, included in Appendix B) with the same network properties are considered for this experiment. All the topologies have same number of nodes, which is 30 and the same average node degree, 3.6. These are the same topologies as used for the results and discussion using PCAT–I in Chapter 5.

All the topologies are assumed to have 20% tunnels implemented. A pair of nodes, $S - D$, AS12-AS16 is provided as the desired nodes to send data over. The assumption made before the experiment was that the SPEA would obtain the the PCAT–II output Pareto optimal paths with different fitness values since the network connectivity is not same for all the topologies and hence, the effectiveness of tunnels was not expected to be the same every time.

Simulations are run for 10 times and the Pareto fronts showing the Pareto optimal solutions are collected.

Figure 6.10a holds the Pareto front for Topology 1, representing 2 Pareto optimal solutions.

Figure 6.13 shows the Pareto front for the 2nd topology. 4 paths are plotted in the front, each of which can be chosen by the end users without having to make a compromise for any of average end-to-end delay and financial cost.

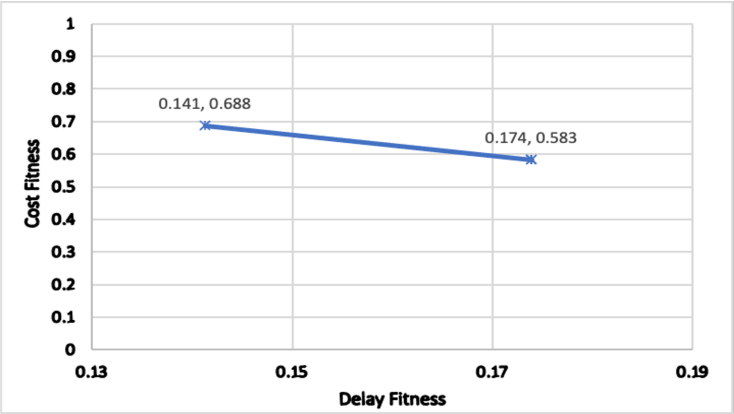


Figure 6.13: Pareto front for 20% tunnels in Topology 2

The number of most suitable $12 - 16$ paths existing in Topology 3 is also 4. The Pareto front 6.14 has their fitness values. Three of the paths have the same set of fitness values and only 1 of them has a different set.

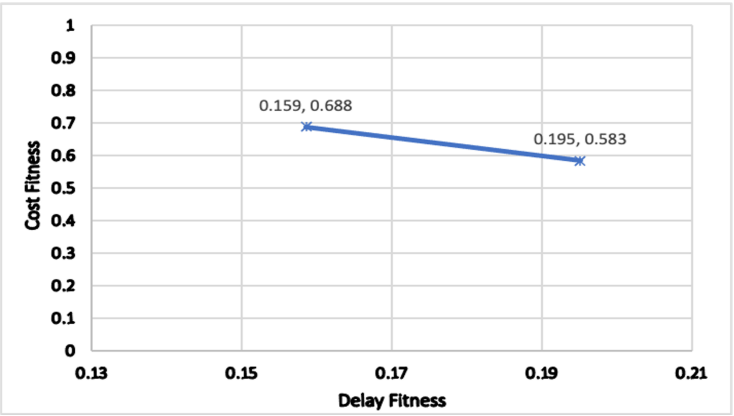


Figure 6.14: Pareto front for 20% tunnels in Topology 3

Both the Pareto fronts for the 4^{th} and 5^{th} toplogies have only one point. Figure 6.15 has 2 possible "fit" paths and Figure 6.16 has 3.

In topology 4, ASes 12 and 16 are only 3 hops away. So, the tunnels did not add much variation in the paths, hence the number of Pareto optimal path is low.

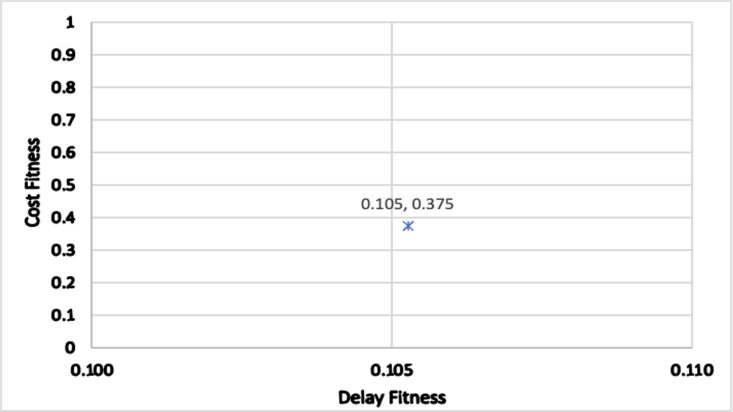


Figure 6.15: Pareto front for 20% tunnels in Topology 4

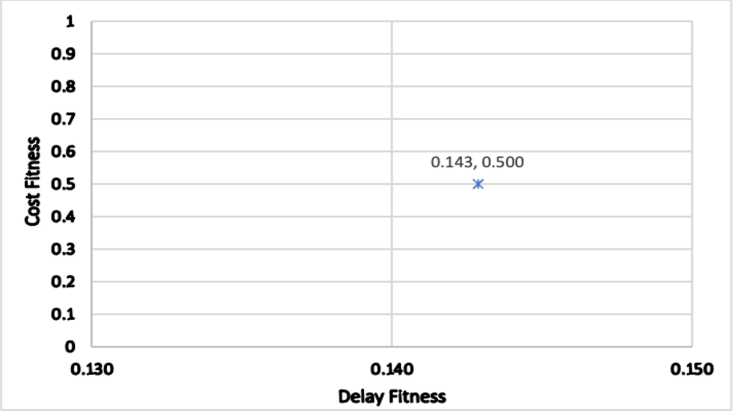


Figure 6.16: Pareto front for 20% tunnels in Topology 5

It can be clearly observed that, for each topology, the PCAT–II outputs a number of possible paths. None of the path(s) presented in the Pareto fronts can be covered by any other existing path between the $S-D$ pair, $12-16$.

6.3.3 Results for Different Percentages

Least cost Pareto optimal paths are generated for AS12-AS16 in Topology 1, considering different percentages of tunnels implemented.

Figure 6.17 shows the Pareto front for 10% tunnels.

Then keeping the existing 10% tunnels, more tunnels are added and the PCAT–II is asked to output the least cost paths. The Pareto front for 20%, 30% and 40% tunnels look alike, as the paths have the same values for their delay and cost fitness. It is shown in Figure 6.18.

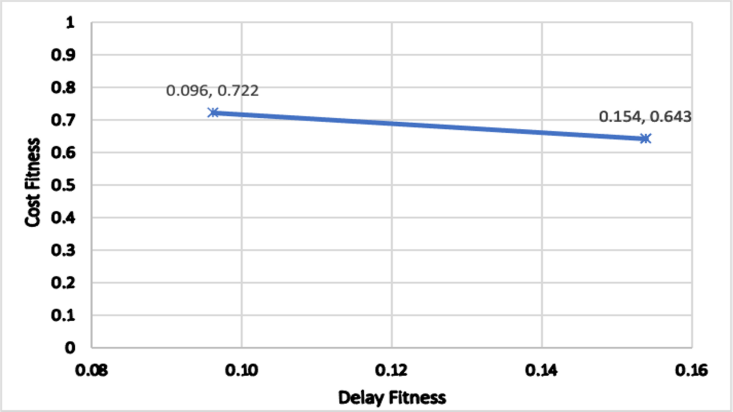


Figure 6.17: Pareto front for 10% tunnels in Topology 1

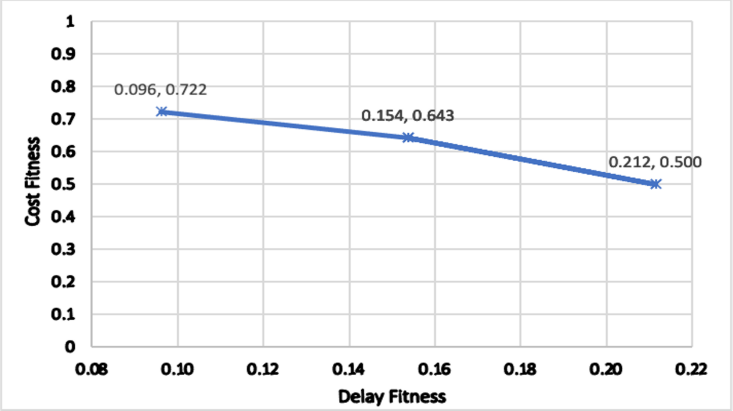


Figure 6.18: Pareto front for 20%, 30% and 40% tunnels in Topology 1

The similarity in the Pareto front is due to the fact that, more tunnels are added to the existing ones as the percentage increases.

The number of Pareto optimal paths that form the Pareto fronts, are included in Table 6.6.

Table 6.6: Number of Pareto optimal paths for different tunnel percentages

Tunnel Percentage	No of Paths
10	3
20	6
30	6
40	6

6.3.4 Results Considering Peak Time

As experimented with the GA-based PCAT, the peak time of the Internet remaining busy is now considered for Topology 1 to observe how the tunnels benefit in such cases. This is an important experiment, since reducing queueing delay is one of the main aims the research aims for. To do the experiment, the PCAT–II is run 10 times, with a ratio of the average end-to-end delay of tunnel and no-tunnel link, 1:20.

Starting from no tunnels present in the topology, the PCAT-II finds Pareto optimal paths for up to 40% tunnels being implemented in the topology.

The intriguing observation from this is that the Pareto front for all the different percentages look same since each has the same values of objective functions plotted there.

The Pareto front is presented in Figure 6.19.

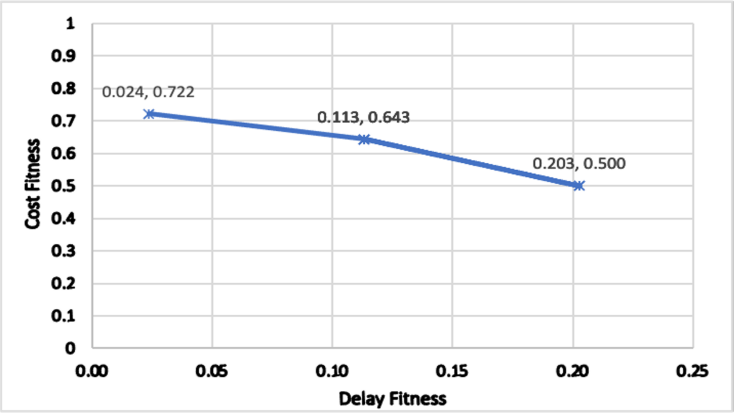


Figure 6.19: Pareto front for different tunnel percentages during peak time

Although there are only 2 points, the number of Pareto optimal paths for different percentages is not the same. This is again due to the way of generating tunnels for the increasing percentages, where new tunnels are added keeping the already-existing ones same as they are.

Table 6.7 includes the numbers.

Table 6.7: Number of Pareto optimal paths for different tunnel percentages

Tunnel Percentage	No of Paths
10	5
20	6
30	7
40	9

This table confirms that during peak time, the presence of more tunnels in the topology adds more benefits to the users as more paths become available to the users.

6.4 Summary

The results discussed in this chapter confirm that the PCAT–II efficiently performs the fitter path calculation based on the user requirements. As desired, the tool can output more than one least cost paths if there is any, without converting the problem into a Single Optimisation.

The important observations here include the fact that as the number of tunnels increase, the number of possible paths also increases. As the requirement is set as minimum possible delay to be experienced while paying the least, the PCAT–II only finds the path(s) confirming:

- Keeping the financial amount to be paid the same as it is, no other path can offer less average end-to-end delay.
- There is no other possible path even with the same average end-to-end delay for a lower financial cost.

Chapter 7

Discussion and Conclusion

This thesis explores the question whether adding user-selectable tunnels to portions of the Internet would provide a notable benefit in terms of performance and flexibility for at least some users. To this end we introduce a new Autonomous System (AS)-Domain tunnelling framework. The main motivation of developing this framework is to overcome the limitations associated with the traditional next-hop forwarding mechanism: strict routing and lack of traffic differentiation, leading to possible traffic imbalance.

7.1 Overview

In this chapter, the research findings are discussed in context, referring back to the novel contributions claimed in Chapter 1. We provide a review of the state-of-the-art in Chapter 2, showing that the end-user has little opportunity to determine how their traffic is transferred across the Internet to its destination. We also introduce various heuristic optimisation algorithms that can be used for path planning given various constraints and goals. Then, in Chapter 4, we introduce a possible tunnelling framework, that is technology agnostic and protects the operators from having to divulge sensitive architectural details. The crucial element is the introduction of a Broker that can advertise tunnels at an Autonomous System level and enable end-users to be granted access to them, for a price.

In Chapter 4, a simple least cost path computation tool is created

based on Dijkstra's algorithm, that is combined with a topology generator and tunnel placement mechanism. This allows various basic experiments to be examined, such as the impact of node degree and tunnel density. This is refined in Chapter 5, where using a Genetic Algorithm, a path computation tool, called Path Computation Algorithm with Tunnels –I (PCAT–I), is created to determine suitable paths between source-destination pairs, taking into account end-to-end delay and financial cost. These two parameters effectively conflict with each other in terms of optimisation. We use a weighting factor to set the balance between them, converting this multi-objective problem into a single objective one. The downside of this approach is that the end-user does not get a clear view of the spectrum of alternatives. Thus, in Chapter 6, a multi-objective path selection tool, PCAT–II is developed, based on SPEA, which evolves a Pareto front of alternative, viable solutions. This can be used by users to see how the various performance objectives inter-play with each other and allow a suitable compromise to be selected.

7.2 Novel Contributions Revisited

7.2.1 Feasibility of the Tunnelling Framework

The proposed framework allows cooperation between end-users and network operators via a simple brokerage mechanism. This mechanism does not require the inter-operator signalling or any information which network operators will not be willing to share due to privacy or trust issue. We avoid inter-tunnels spanning ASes as this would typically require cooperation between Service Providers. Instead, we focus on intra-AS tunnels that are added in a relatively low concentration.

The Directory Service Broker (DSB) operates with the cooperation of a number of network operators providing end-users selectable usage of tunnels to send their data over. It doesn't need to retrieve any router-level Internet topology map rather it only requires an AS-level map view provided by operators or obtained through traceroute or BGP messages.

7.2.2 Exploring Benefits of Tunnels: Are Internet Tunnels Worthwhile?

Initially, a tool has been designed and evaluated using a standard implementation of Dijkstra's algorithm. This tool reads an AS topology

and automatically generates an ASBR topology from which least cost paths are obtained. Tunnels are incorporated, potentially providing a benefit in terms of reduced end-to-end delay, which can be ascertained under various conditions. We show that although the delay-cost benefits would be marginal for many users, those users whose source-destination path is in relatively close proximity to one or more tunnels can experience a worthwhile gain. This is more apparent as the cost differential increases.

We believe that end-user selectable access to tunnels provides a worthwhile degree of choice whilst avoiding the issues of network security and would allow more flexible use of the Internet as demands on its resources continue to grow and traffic of dissimilar types are supported.

7.2.3 Path Selection Tool for End User Software

The work introduces financial cost besides the average end-to-end delay and modifies the path selection algorithm to incorporate multi-objective goals. Two versions of path computation tool have been developed for the end-user software, building upon the basic tool in Chapter 4, and described in detail in Chapters 5 and 6, respectively: Path Computation Algorithm with Tunnels, PCAT–I and PCAT–II.

PCAT–I

GA is implemented to develop the PCAT – I, which successfully calculates least cost path against the total average end-to-end delay and financial cost for each possible route for sending data over. This is done by finding the fittest path using an Equation developed to give certain weight to each of the constraints, which is:

$$F = \alpha \times f_1 + (1 - \alpha) \times f_2 \tag{7.1}$$

where, f_1 = delay fitness and
f_2 = cost fitness. These are calculated using the equations:

$$f_1 = \frac{d_\pi}{1 + d_{max}} \tag{7.2}$$

where d_π = average end-to-end delay of the path and
d_{max} is the maximum possible average end-to-end delay for that path.

$$f_2 = \frac{c_\pi}{c_t + c_\pi} \tag{7.3}$$

wherec_π = total financial cost of a path and
c_t = cost of using one intra-AS tunnel in the specific network topology.

The equations developed in this research, for adapting the GA, are also novel.

In addition, the results from the experiments using PCAT–I, as described in Chapter 5, proves that the presence of tunnels helps reduce the overall end-to-end delay.

PCAT–II

Since the GA finds one least cost path based on a combined value of the two provided cost metrics, another version of the path computation tool is developed using SPEA, which is efficient to find all the best suitable paths based on the multiple constraints, instead of a combined one. This is done using the same Equations for f_1 and f_2 and implementing the theory of Pareto dominance. Unlike PCAT–I, PCAT–II shows all the least cost paths even if they have the same cost associated with them.

7.3 Publications

- The results from Chapter 4 have been published in two papers:
  - Are internet tunnels worth-while? 28$^{\text{th}}$ International Telecommunication Networks and Applications Conference (ITNAC), pages 1–6. IEEE, 2018 [192].
  - Tunnelling the internet. Journal of Telecommunications and the Digital Economy, 7(1), 2019 [193].

  .
- Some of the initial results were also presented as a poster:
  - Tunnelling the Internet. ACM-W womENcourage 2019 [213]
- An earlier version of Chapter 5 was presented as a poster:
  - Genetic Algorithm for Least Cost Routing in the Networking. ACM-W womENcourage, 2020 [214].

7.4 Future Work

Although beyond the scope of this thesis, an interesting and useful enhancement to PCAT–II would be the ability to add additional constraints such as "no-go" regions that the end-user would prefer to avoid. The judicious selection of tunnels provides an opportunity for an end-user to detour their traffic around undesirable geographical regions whist being sensitive to the impact on the end-to-end delay.

Since the target topologies of this research focus on relatively small geographical areas, computational performance is reasonable. However, the cost of memory and time to do the path computation for a very large topology would be high. The PCAT algorithms could be further improved using parallel GA and SPEA variants, allowing the problem to be decomposed into concurrent mini problems. However, care would need to be taken to consider how global optimal solutions could be discovered if only localised regional data was presented to each instantiation.

7.5 Concluding Remarks

In this thesis a new framework for loose source routing via selective tunnels has been proposed. The framework provides a voluntary mechanism for end-users to have some control over the path their information takes across the internet. The technologies required to perform this task already exist. The principle new elements required, apart from the willingness of the key actors to participate, are a brokerage entity, an end-user path selection algorithm, and a tunnel authorisation and policing mechanism. This work evaluates whether such a system adds a significant benefit to the transport of preferred flows across portions of the Internet.

We believe that end-user selectable access to tunnels provides a suitable degree of choice whilst avoiding the issues of trust and would allow better management of the Internet as demands on its resources continue to grow.

The research confirms that end-users could derive meaningful benefit if the selective use of tunnelling was offered across the Internet. However, it remains an open question, beyond the scope of this research, whether users would be willing to pay for such a service.

Bibliography

[1] Xin Chen. *Energy Efficient Wired Networking*. PhD thesis, Queen Mary, University of London, 2015. Supervised by: Dr Chris Phillips.

[2] Eckart Zitzler. Computer Engineering and Communication Networks Lab (TIK) Swiss Federal Institute of Technology (ETH) Zurich Gloriastr. 35, CH–8092 Zürich, Switzerland zitzler@ tik. ee. ethz. ch.

[3] Christian Esteve Rothenberg and Andreas Roos. A review of policy-based resource and admission control functions in evolving access and next generation networks. *Journal of Network and Systems Management*, 16(1):14–45, 2008.

[4] Jeffrey R Yost. The Origin and Early History of the Computer Security Software Products Industry. *IEEE Annals of the History of Computing*, 37(2):46–58, 2015.

[5] Telecommunication Standardization Sector International Telecommunication Union. Resource and admission control functions in next generation networks. *ITU-T Draft/Prepublication Y. 2111 (formerly Y. RACF)*.

[6] Mohit Chamania, Marek Drogon, and Admela Jukan. An Open-Source Path Computation Element (PCE) Emulator: Design, Implementation, and Performance. *Journal of lightwave technology*, 30(4):414–426, 2011.

[7] Sukrit Dasgupta, Jaudelice C De Oliveira, and Jean-Philippe Vasseur. Path-Computation-Element-Based Architecture for Interdomain MPLS/GMPLS Traffic Engineering: Overview and Performance. *IEEE Network*, 21(4):38–45, 2007.

[8] Border Gateway Protocol. . Last accessed: 2020-07-10.

[9] Wajdi Louati and Djamal Zeghlache. Network-Based Virtual Personal Overlay Networks Using Programmable Virtual Routers. *IEEE Communications Magazine*, 43(8):86–94, 2005.

[10] Grzegorz Rzym, Krzysztof Wajda, and Krzysztof Rzym. Analysis of pce-based path optimization in multi-domain sdn/mpls/bgp-ls network. In *2016 18th International Conference on Transparent Optical Networks (ICTON)*, pages 1–5. IEEE, 2016.

[11] Margaret Rouse. Path Computation Element (PCE). http://searchsdn.techtarget.com/definition/Path-Computation-Element-PCE, . Last accessed: 2020-07-10.

[12] Adrian Farrel. *The Internet and Its Protocols*. Morgan Kaufmann, USA, 2004.

[13] Stefano Secci, Jean-Louis Rougier, and Achille Pattavina. On the Selection of Optimal Diverse AS-Paths for Inter-Domain IP/(G)MPLS Tunnel Provisioning. In *2008 4th International Telecommunication Networking Workshop on QoS in Multiservice IP Networks*, pages 235–241. IEEE, 2008.

[14] Internet service providers charging for premium access hold us all to ransom. https://www.theguardian.com/technology/2014/apr/28/internet-service-providers-charging-premium-access, 2008. Last accessed: 2020-07-10.

[15] David Alderson, John Doyle, Ramesh Govindan, and Walter Willinger. Toward an Optimization-Driven Framework for Designing and Generating Realistic Internet Topologies. *ACM SIGCOMM Computer Communication Review*, 33(1):41–46, 2003.

[16] Geoff Huston. Exploring Autonomous System Numbers. *The Internet Protocol Journal*, 9(1), 2006.

[17] Y Rekhter, T Li, and S Hares. Border Gateway Protocol 4 (BGP-4). rfc 4271 (draft standard). *Updated by RFC 6286*, 2006.

[18] Gary Scott Malkin and Martha E Steenstrup. Distance-vector routing. In *Routing in communications networks*, pages 83–98. 1995.

[19] Stephen A Thomas. *IPng and the TCP/IP Protocols: Implementing the Next Generation Internet*. John Wiley & Sons, Inc., 1996.

[20] Andrey Sapegin and Steve Uhlig. On the extent of correlation in BGP updates in the Internet and what it tells us about locality of BGP routing events. *Computer Communications*, 36(15-16):1592–1605, 2013.

[21] Jake Brutlag. AS65000 BGP Routing Table Analysis Report. http://bgp.potaroo.net/as2.0/bgp-active.html. Last accessed: 2020-07-10.

[22] Cisco Press. BGP Basics: Internal And External BGP. https://www.networkcomputing.com/data-centers/bgp-basics-internal-and-external-bgp/1830126875, July 2, 2017. Last accessed: 2020-07-10.

[23] Rony Kay. Pragmatic Network Latency Engineering Fundamental Facts and Analysis. *cPacket Networks, White Paper*, pages 1–31, 2009.

[24] Ed Blair Margaret Rouse, Matthew Haughn. Latency. http://whatis.techtarget.com/definition/latency, 2015. Last accessed: 2020-07-10.

[25] What are the different types of network delay? https://www.educative.io/edpresso/what-are-the-different-types-of-network-delay. Last accessed: 2021-01-01.

[26] Steve Souder. Velocity and the Bottom Line. http://radar.oreilly.com/2009/07/velocity-making-your-site-fast.html, July 1, 2009. Last accessed: 2020-07-10.

[27] Jake Brutlag. Speed Matters for Google Web Search. http://radar.oreilly.com/2009/07/velocity-making-your-site-fast.html, June 22, 2009. Last accessed: 2020-07-10.

[28] Tim Wu. NETWORK NEUTRALITY, BROADBAND DISCRIMINATION. *J. on Telecomm. & High Tech. L.*, 2:141, 2003.

[29] Mathew Honan. Inside Net Neutrality: Is your ISP filtering content? https://www.macworld.com/article/1132075/web-apps/netneutrality1.html, February 12, 2008. Last accessed: 2020-07-10.

[30] The Guardian. Net neutrality: Amazon among top internet firms planning day of action. https:

//www.theguardian.com/technology/2017/jun/06/net-neutrality-amazon-etsy-kickstarter-protest, June 6, 2017. Last accessed: 2020-07-10.

[31] Hagai-Bar-El. Protecting network neutrality: both important and hard. https://www.hbarel.com/analysis/policy/what-is-network-neutrality, August 19, 2017. Last accessed: 2020-07-10.

[32] The Guardian. Net neutrality enshrined in Dutch law. https://www.theguardian.com/technology/2011/jun/23/netherlands-enshrines-net-neutrality-law, June 23, 2011. Last accessed: 2020-07-10.

[33] Canadian Radio-television and Telecommunications Commission. Telecom Decision CRTC 2011-44. https://crtc.gc.ca/eng/archive/2011/2011-44.html, January 25, 2011. Last accessed: 2020-07-10.

[34] Wikipedia. Net Neutrality By Country. https://en.wikipedia.org/wiki/Net_neutrality_by_country, . Last accessed: 2020-07-10.

[35] Joshua Wright Ajit Pai. The Internet isn't broken. Obama doesn't need to 'fix' it. . https://www.chicagotribune.com/opinion/commentary/ct-internet-regulations-fcc-ftc-obama-broadband-perspec-0219-20150218-story.html, February 18, 2015. Last accessed: 2020-07-10.

[36] Federal Communications Commission et al. In the matter of protecting and promoting the open internet. *Washington, DC Adopted February*, 26:2015, 2015.

[37] The New York Times. F.C.C. Invokes Internet Freedom While Trying to Kill It . https://www.nytimes.com/2017/04/29/opinion/sunday/fcc-invokes-internet-freedom-while-trying-to-kill-it.html, April 29, 2017. Last accessed: 2020-07-10.

[38] Wikipedia. Net neutrality in the United States. https://en.wikipedia.org/wiki/Net_neutrality_in_the_United_States, . Last accessed: 2020-07-10.

[39] Patrick Le Callet, Sebastian Möller, Andrew Perkis, et al. Qualinet White Paper on Definition of Quality of Experience. *European net-*

work on quality of experience in multimedia systems and services (COST Action IC 1003), 3(2012), 2012.

[40] Jon Postel et al. Internet protocol. 1981.

[41] Michael Montgomery and Gustavo De Veciana. Hierarchical Source Routing Through Clouds. In *Proceedings. IEEE INFOCOM'98, the Conference on Computer Communications. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Gateway to the 21st Century (Cat. No. 98*, volume 2, pages 685–692. IEEE, 1998.

[42] Mourad Soliman, Biswajit Nandy, Ioannis Lambadaris, and Peter Ashwood-Smith. Exploring Source Routed Forwarding in SDN-Based WANs. In *2014 IEEE International Conference on Communications (ICC)*, pages 3070–3075. IEEE, 2014.

[43] Yih-Chun Hu and David B Johnson. Implicit Source Routes forOn-Demand Ad Hoc Network Routing. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 1–10, 2001.

[44] S Previdi et al. Cisco Systems, Inc.," Segment Routing with IS-IS Routing Protocol, draft-previdi-filsfils-isis-segment-routing-02". Technical report, Internet-Draft, Mar. 20, 2013, A55, 2013.

[45] Jeffrey Shafer, Brent Stephens, Michael Foss, Scott Rixner, and Alan L Cox. Axon: A Flexible Substrate for Source-routed Ethernet. In *Proceedings of the 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pages 1–11, 2010.

[46] Chuanxiong Guo, Guohan Lu, Helen J Wang, Shuang Yang, Chao Kong, Peng Sun, Wenfei Wu, and Yongguang Zhang. SecondNet: A DataCenter Network Virtualization Architecture with Bandwidth Guarantees. In *Proceedings of the 6th International COnference*, pages 1–12, 2010.

[47] Brent Stephens. *Designing Scalable Networks for Future Large Datacenters*. PhD thesis, 2012.

[48] Katerina Argyraki and David R Cheriton. Loose source routing as a mechanism for traffic policies. In *Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, pages 57–64, 2004.

[49] Luca Boccassi, Marwan M Fayed, and Mahesh K Marina. Binder: A System to Aggregate Multiple Internet Gatewaysin Community Networks. In *Proceedings of the 2013 ACM MobiCom workshop on Lowest cost denominator networking for universal access*, pages 3–8, 2013.

[50] Layer 2 Tunneling Protocol (L2TP). http://www.thenetworkencyclopedia.com/entry/layer-2-tunneling-protocol-l2tp/. Last accessed: 2020-07-10.

[51] Margaret Rouse. Layer Two Tunneling Protocol (L2TP). http://searchenterprisewan.techtarget.com/definition/Layer-Two-Tunneling-Protocol, . Last accessed: 2020-07-10.

[52] Express VPN. What is L2TP/IPsec? https://www.expressvpn.com/what-is-vpn/protocols/l2tp. Last accessed: 2020-07-10.

[53] Kireeti Kompella, George Swallow, Carlos Pignataro, Nagendra Kumar Nainar, Sam Aldrin, and Mach (Guoyi) Chen. Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures. *RFC*, 8029:1–78, 2017.

[54] Michael Bushong Walter J. Goralski, Cathy Gadecki. Label Switching and Label Switched Paths (LSPs). https://www.dummies.com/programming/networking/juniper/label-switching-and-label-switched-paths-lsps/. Last accessed: 2020-07-10.

[55] Jing Fu and Jennifer Rexford. Efficient ip-address lookup with a shared forwarding table for multiple virtual routers. In *Proceedings of the 2008 ACM CoNEXT Conference*, pages 1–12, 2008.

[56] Yihua He, Georgos Siganos, and Michalis Faloutsos. Internet Topology, 2009.

[57] David Alderson, Lun Li, Walter Willinger, and John C Doyle. Understanding Internet Topology: Principles, Models, and Validation. *IEEE/ACM Transactions on networking*, 13(6):1205–1218, 2005.

[58] Richard G Clegg, Carla Di Cairano-Gilfedder, and Shi Zhou. A critical look at power law modelling of the Internet. *Computer Communications*, 33(3):259–268, 2010.

[59] Sally Floyd and Vern Paxson. Difficulties in simulating the internet. *IEEE/ACm Transactions on Networking*, 9(4):392–403, 2001.

[60] Ricardo V Oliveira, Beichuan Zhang, and Lixia Zhang. Observing the Evolution of Internet AS Topology. In *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 313–324, 2007.

[61] Bruno Quoitin. Topology Generation based on Network Design Heuristics. In *Proceedings of the 2005 ACM conference on Emerging network experiment and technology*, pages 278–279, 2005.

[62] Hamed Haddadi, Steve Uhlig, Andrew Moore, Richard Mortier, and Miguel Rio. Modeling Internet Topology Dynamics. *ACM SIGCOMM Computer Communication Review*, 38(2):65–68, 2008.

[63] Bruno Quoitin, Virginie Van den Schrieck, Pierre François, and Olivier Bonaventure. IGen: Generation of Router-level Internet Topologies through Network Design Heuristics. In *2009 21st International Teletraffic Congress*, pages 1–8. IEEE, 2009.

[64] Naomi A Arnold, Raul J Mondragon, and Richard G Clegg. Changing the tune: mixtures of network models that vary in time. *arXiv preprint arXiv:1909.13253*, 2019.

[65] Hamed Haddadi, Miguel Rio, Gianluca Iannaccone, Andrew Moore, and Richard Mortier. Network Topologies: Inference, Modeling, and Generation. *IEEE Communications Surveys & Tutorials*, 10(2):48–69, 2008.

[66] Micha Karoński and Andrzej Ruciński. The origins of the theory of random graphs. In *The Mathematics of Paul Erdös I*, pages 311–336. Springer, 1997.

[67] Paul Erdös and Alfréd Rényi. On random graphs I. *Math. debrecen*, 6:290–297, 1959.

[68] Paul Erdős and Alfréd Rényi. ON THE EVOLUTION OF RANDOM GRAPHS. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[69] Bernard M Waxman. Routing of Multipoint Connections. *IEEE journal on selected areas in communications*, 6(9):1617–1622, 1988.

[70] Kenneth L Calvert, Matthew B Doar, and Ellen W Zegura. Modeling Internet Topology. *IEEE Communications magazine*, 35(6): 160–163, 1997.

[71] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers. BRITE: An Approach to Universal Topology Generation . In *MASCOTS 2001, Proceedings Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 346–353. IEEE, 2001.

[72] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On Power-Law Relationships of the Internet Topology. *ACM SIGCOMM computer communication review*, 29(4):251–262, 1999.

[73] Tian Bu and Don Towsley. On distinguishing between internet power law topology generators. In *Proceedings. twenty-first annual joint conference of the ieee computer and communications societies*, volume 2, pages 638–647. IEEE, 2002.

[74] Georgos Siganos, Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. Power laws and the as-level internet topology. *IEEE/ACM Transactions on networking*, 11(4):514–524, 2003.

[75] Christopher R Palmer and J Greg Steffan. Generating Network Topologies that Obey Power Laws. In *Globecom'00-IEEE. Global Telecommunications Conference. Conference Record (Cat. No. 00CH37137)*, volume 1, pages 434–438. IEEE, 2000.

[76] William Aiello, Fan Chung, and L Lu A Random Graph Model. A Random Graph Model for Massive Graphs. In *Proc. STOC 2000*, 2001.

[77] Albert-Laszlo Barabasi and Reka Albert. Emergence of Scaling in Random Networks. *Science*, 2865439, 1999.

[78] Shi Zhou and Raúl J Mondragón. The rich-club phenomenon in the internet topology. *IEEE Communications Letters*, 8(3):180–182, 2004.

[79] Ricardo V Oliveira, Dan Pei, Walter Willinger, Beichuan Zhang, and Lixia Zhang. In Search of the Elusive Ground Truth: the Internet's AS-Level Connectivity Structure. *ACM SIGMETRICS Performance Evaluation Review*, 36(1):217–228, 2008.

[80] Alberto Medina, Ibrahim Matta, and John Byers. BRITE: A Flexible Generator of Internet Topologies. 2000.

[81] Jared Winick and Sugih Jamin. Inet-3.0: Internet topology generator. Technical report, Technical Report CSE-TR-456-02, University of Michigan, 2002.

[82] Cheng Jin, Qian Chen, and Sugih Jamin. Inet: Internet Topology Generator. 2000.

[83] Shi Zhou and Raúl J Mondragón. Accurately modeling the Internet topology. *Physical Review E*, 70(6):066108, 2004.

[84] Shi Zhou. Characterising and modelling the Internet topology—the rich-club phenomenon and the PFP model. *BT Technology Journal*, 24(3):108–115, 2006.

[85] Wolfgang Mühlbauer, Anja Feldmann, Olaf Maennel, Matthew Roughan, and Steve Uhlig. Building an AS-Topology Model that Captures Route Diversity. *ACM SIGCOMM Computer Communication Review*, 36(4):195–206, 2006.

[86] Wolfgang Mühlbauer, Steve Uhlig, Bingjie Fu, Mickael Meulle, and Olaf Maennel. In Search for An Appropriate Granularity to Model Routing Policies. *ACM SIGCOMM Computer Communication Review*, 37(4):145–156, 2007.

[87] Renata Teixeira, Aman Shaikh, Tim Griffin, and Jennifer Rexford. Dynamics of Hot-Potato Routing in IP Networks. In *Proceedings of the joint international conference on Measurement and modeling of computer systems*, pages 307–319, 2004.

[88] Young Hyun, Andre Broido, et al. Traceroute and BGP AS Path Incongruities. Technical report, Cooperative Association for Internet Data Analysis (CAIDA), 2003.

[89] Philip L. Frana and Thomas J. Misa. An Interview with Edsger W. Dijkstra. *Commun. ACM*, 53(8):41–47, August 2010. ISSN 0001-0782. doi: 10.1145/1787234.1787249. URL https://doi.org/10.1145/1787234.1787249.

[90] Jeff Doyle. *CCIE Professional Development: Routing TCP/IP*. Cisco Press, 1998.

[91] Edsger W Dijkstra et al. A Note on Two Problems in Connexion with Graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[92] Vimal L Vachhani, Vipul K Dabhi, and Harshadkumar B Prajapati. Survey of Multi-Objective Evolutionary Algorithms. In *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]*, pages 1–9. IEEE, 2015.

[93] Ruhul Sarker, Masoud Mohammadian, and Xin Yao. *Evolutionary Optimization*, volume 48. Springer Science & Business Media, 2002.

[94] David E Goldenberg. Genetic Algorithms in Search, Optimization and Machine Learning, 1989.

[95] Carlos A Coello. An Updated Survey of GA-Based Multiobjective Optimization Techniques. *ACM Computing Surveys (CSUR)*, 32(2): 109–143, 2000.

[96] M Laumann, E Zitzler, and S Bleuler. A Tutorial on Evolutionary Multiobjective Optimization, Metaheuristics for Multiobjective Optimisation,[In:]. *Lecture Notes in Economics and Mathematical Systems*, pages 3–37, 2004.

[97] Marcos LP Bueno and Gina MB Oliveira. Multiobjective evolutionary algorithms and a combined heuristic for route reconnection applied to multicast flow routing. In *2010 10th IEEE International Conference on Computer and Information Technology*, pages 464–471. IEEE, 2010.

[98] Thomas Bäck, Günter Rudolph, and Hans-Paul Schwefel. Evolutionary Programming and Evolution Strategies: Similarities and Differences. In *In Proceedings of the Second Annual Conference on Evolutionary Programming*. Citeseer, 1993.

[99] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT press, 1998.

[100] Darrell Whitley. An overview of evolutionary algorithms: practical issues and common pitfalls. *Information and software technology*, 43(14):817–831, 2001.

[101] John Henry Holland et al. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT press, 1992.

[102] Ingo Rechenberg. Evolution Strategy: Nature's Way of Optimization. In *Optimization: Methods and applications, possibilities and limitations*, pages 106–126. Springer, 1989.

[103] Thomas Back, Ulrich Hammel, and H-P Schwefel. Evolutionary Computation: Comments on the History and Current State. *IEEE transactions on Evolutionary Computation*, 1(1):3–17, 1997.

[104] Lawrence J Fogel, Alvin J Owens, and Michael J Walsh. Artificial Intelligence through Simulated Evolution. 1966.

[105] David B Fogel. A Comparison of Evolutionary Programming and Genetic Algorithms on Selected Constrained Optimization Problems. *Simulation*, 64(6):397–404, 1995.

[106] John Galletly. Evolutionary algorithms in theory and practice. *Kybernetes*, 1998.

[107] Thomas Back. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford university press, 1996.

[108] Frank Hoffmeister and Thomas Bäck. Genetic Algorithms and Evolution Strategies: Similarities and Differences. In *International Conference on Parallel Problem Solving from Nature*, pages 455–469. Springer, 1990.

[109] Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. Exploration and Exploitation in Evolutionary Algorithms: A Survey. *ACM computing surveys (CSUR)*, 45(3):1–33, 2013.

[110] Alexander Schrijver. Combinatorial Optimization: Polyhedra and Efficiency (Algorithms and Combinatorics). *Journal-Operational Research Society*, 55(9):1018–1018, 2004.

[111] Fanghan Liu, Xiaobing Tang, and Zhaohui Yang. An Encoding Algorithm Based on the Shortest Path Problem. In *2018 14th International Conference on Computational Intelligence and Security (CIS)*, pages 35–39. IEEE, 2018.

[112] James Edward Baker. Adaptive Selection Methods for Genetic Algorithms. In *Proceedings of an International Conference on Genetic Algorithms and their applications*, pages 101–111. Hillsdale, New Jersey, 1985.

[113] Runwei Cheng. *Genetic Algorithms and Engineering Optimization*. Wiley-Interscience, 2000.

[114] Chang Wook Ahn and Rudrapatna S Ramakrishna. A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations. *IEEE transactions on evolutionary computation*, 6(6): 566–579, 2002.

[115] David E Goldberg and Kalyanmoy Deb. A Comparative Analysis of Selection Schemes used in Genetic Algorithms. In *Foundations of genetic algorithms*, volume 1, pages 69–93. Elsevier, 1991.

[116] Vassil Guliashki, Hristo Toshev, and Chavdar Korsemov. Survey of Evolutionary Algorithms Used in Multiobjective Optimization . *Problems of engineering cybernetics and robotics*, 60(1):42–54, 2009.

[117] Jürgen Branke, Jurgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Roman Slowiński. *Multiobjective Optimization: Interactive and Evolutionary Approaches*, volume 5252. Springer Science & Business Media, 2008.

[118] David Edward Goldberg. *Computer-Aided Gas Pipeline Operation Using Genetic Algorithms and Rule Learning*. PhD thesis, 1984.

[119] David E Goldberg. Computer-Aided Pipeline Operation Using Genetic Algorithms and Rule Learning. PART I: Genetic Algorithms in Pipeline Optimization. *Engineering with Computers*, 3(1):35–45, 1987.

[120] Salman Yussof and Ong Hang See. Finding Multi-Constrained Path Using Genetic Algorithm. In *2007 IEEE International Conference on Telecommunications and Malaysia International Conference on Communications*, pages 713–718. IEEE, 2007.

[121] Bilal Gonen. Genetic Algorithm Finding the Shortest Path in Networks. *Reno: University of Nevada*, 2006.

[122] Salman Yussof, Rina Azlin Razali, and Ong Hang See. A parallel Genetic Algorithm for Shortest Path Routing Problem. In *2009 International Conference on Future Computer and Communication*, pages 268–273. IEEE, 2009.

[123] Zongyan Xu, Haihua Li, and Ye Guan. A study on the shortest path problem based on improved genetic algorithm. In *2012 Fourth International Conference on Computational and Information Sciences*, pages 325–328. IEEE, 2012.

[124] Ching-Sheng Chiu. A Genetic Algorithm for Multiobjective Path Optimisation Problem. In *2010 Sixth International Conference on Natural Computation*, volume 5, pages 2217–2222. IEEE, 2010.

[125] Mitsuo Gen and Lin Lin. A weighted sum-based genetic algorithms for bicriteria network design problem. In *Proc of the 3th International Conf on Info. and Mana. Sci*, volume 22, pages 419–125, 2004.

[126] Zhendong Liu, Yawei Kong, and Bin Su. An improved genetic algorithm based on the shortest path problem. In *2016 IEEE international conference on information and automation (ICIA)*, pages 328–332. IEEE, 2016.

[127] Sakshi Arora. An Archive Enhanced Hybrid Genetic Algorithm for Shortest Path Routing Problem. In *Proceedings of the Sixth International Conference on Computer and Communication Technology 2015*, pages 147–151, 2015.

[128] Mitsuo Gen, Runwei Cheng, and Shumuel S Oren. Network design techniques using adapted genetic algorithms. *Advances in Engineering Software*, 32(9):731–744, 2001.

[129] Arthur Warburton. Approximation of pareto optima in multiple-objective, shortest-path problems. *Operations research*, 35(1):70–79, 1987.

[130] Praveen Kumar Shukla and Surya Prakash Tripathi. A review on the interpretability-accuracy trade-off in evolutionary multiobjective fuzzy systems (emofs). *Information*, 3(3):256–277, 2012.

[131] Praveen Kumar Shukla and Surya Prakash Tripathi. A survey on interpretability-accuracy (ia) trade-off in evolutionary fuzzy systems. In *2011 Fifth International Conference on Genetic and Evolutionary Computing*, pages 97–101. IEEE, 2011.

[132] A Kaur and PK Shukla. A Review on Evolutionary Multiobjective Optimization for Routing Problem of Computer Networks. 2013.

[133] Antonin Ponsich, Antonio Lopez Jaimes, and Carlos A Coello Coello. A Survey on Multiobjective Evolutionary Algorithms for the Solution of the Portfolio Optimization Problem and Other Finance and Economics Applications. *IEEE Transactions on Evolutionary Computation*, 17(3):321–344, 2012.

[134] Carlos A Coello Coello and Gary B Lamont. *Applications of Multi-Objective Evolutionary Algorithms*, volume 1. World Scientific, 2004.

[135] Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms*, volume 16. John Wiley & Sons, 2001.

[136] Anirban Mukhopadhyay, Ujjwal Maulik, Sanghamitra Bandyopadhyay, and Carlos Artemio Coello Coello. A Survey of Multiobjective EvolutionaryAlgorithms for Data Mining: Part I. *IEEE Transactions on Evolutionary Computation*, 18(1):4–19, 2013.

[137] Anirban Mukhopadhyay, Ujjwal Maulik, Sanghamitra Bandyopadhyay, and Carlos A Coello Coello. Survey of Multiobjective Evolutionary Algorithms for Data Mining: Part II. *IEEE Transactions on Evolutionary Computation*, 18(1):20–35, 2013.

[138] Laetitia Jourdan, David Corne, Dragan Savic, and Godfrey Walters. Hybridising Rule Induction and Multi-Objective Evolutionary Search for Optimising Water Distribution Systems. In *Fourth International Conference on Hybrid Intelligent Systems (HIS'04)*, pages 434–439. IEEE, 2004.

[139] Yezid Donoso, Carolina Alvarado, Alfredo Perez, and Ivan Herazo. A Multi-Objective Solution Applying MOEA in Optical Networks. In *2007 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making*, pages 360–367. IEEE, 2007.

[140] Wei Peng and Qingfu Zhang. Network Topology Planning Using MOEA/D with Objective-Guided Operators. In *Proceedings of the 12th International Conference on Parallel Problem Solving from Nature - Volume Part II*, PPSN'12, page 62–71, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 9783642329630.

[141] J Yazdi, S Mohammadiun, R Sadiq, SAA Salehi Neyshabouri, and A Alavi Gharahbagh. Assessment of Different MOEAs for Rehabilitation Evaluation of Urban Stormwater Drainage Systems–Case study: Eastern Catchment of Tehran. *Journal of Hydro-environment Research*, 21:76–85, 2018.

[142] D Lubin Balasubramanian and V Govindasamy. Study on Evolutionary Approaches for Improving the Energy Efficiency of Wireless Sensor Networks Applications. *EAI Endorsed Transactions on Internet of Things*, 5(20), 2020.

[143] Shuai Wang and Jing Liu. Constructing Robust Cooperative Networks using a Multi-Objective Evolutionary Algorithm. *Scientific reports*, 7:41600, 2017.

[144] Adriana Menchaca-Mendez and Carlos A Coello Coello. MD-MOEA : A New MOEA based on the Maximin Fitness Functionand Euclidean Distances between Solutions. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 2148–2155. IEEE, 2014.

[145] Huanlai Xing, Zhaoyuan Wang, Tianrui Li, Hui Li, and Rong Qu. An Improved MOEA/D Algorithm for Multi-Objective Multicast Routing With Network Coding. *Applied Soft Computing*, 59:88–103, 2017.

[146] Joshua Knowles, Martin Oates, and David Corne. Advanced multi-objective evolutionary algorithms applied to two problems in telecommunications. *BT Technology Journal*, 18(4):51–65, 2000.

[147] JD Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. *Ph. D. Dissertation, Vanderbilt University*, 1984.

[148] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. *TIK-report*, 103, 2001.

[149] Abdullah Konak, David W Coit, and Alice E Smith. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety*, 91(9):992–1007, 2006.

[150] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32–49, 2011.

[151] Tina Yu and Lawrence Davis. An Introduction to Evolutionary Computation in Practice. In *Evolutionary Computation in Practice*, pages 1–8. Springer, 2008.

[152] CA Coello Coello. Evolutionary Multi-objective Optimization: A Historical View of the Field. *IEEE computational intelligence magazine*, 1(1):28–36, 2006.

[153] Nidamarthi Srinivas and Kalyanmoy Deb. Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary computation*, 2(3):221–248, 1994.

[154] Jeffrey Horn, Nicholas Nafpliotis, and David E Goldberg. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence*, pages 82–87. Ieee, 1994.

[155] Carlos M Fonseca, Peter J Fleming, et al. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In *Icga*, volume 93, pages 416–423. Citeseer, 1993.

[156] Joshua D Knowles and David W Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary computation*, 8(2):149–172, 2000.

[157] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. A Fast Elitist Non-Dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *International conference on parallel problem solving from nature*, pages 849–858. Springer, 2000.

[158] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE transactions on evolutionary computation*, 18(4):577–601, 2013.

[159] Himanshu Jain and Kalyanmoy Deb. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: handling constraints and extending to an adaptive approach. *IEEE Transactions on evolutionary computation*, 18(4):602–622, 2013.

[160] Carlos A Coello Coello, Gary B Lamont, David A Van Veldhuizen, et al. *Evolutionary Algorithms for Solving Multi-Objective Problems*, volume 5. Springer, 2007.

[161] Jorge Crichigno and Benjamín Barán. Multiobjective multicast routing algorithm for traffic engineering. In *Proceedings. 13th International Conference on Computer Communications and Networks (IEEE Cat. No. 04EX969)*, pages 301–306. IEEE, 2004.

[162] Divya Kumar, Divya Kashyap, KK Mishra, and AK Mishra. Routing path determination using qos metrics and priority based evolutionary optimization. In *2011 IEEE International Conference on High Performance Computing and Communications*, pages 615–621. IEEE, 2011.

[163] Pham Tran Anh Quang, Jean-Michel Sanner, Cedric Morin, and Yassine Hadjadj-Aoul. Multi-objective multi-constrained qos routing in large-scale networks: A genetic algorithm approach. In *2018 International conference on smart communications in network technologies (SaCoNeT)*, pages 55–60. IEEE, 2018.

[164] Baoxian Zhang, Jie Hao, and Hussein T Mouftah. Bidirectional multi-constrained routing algorithms. *IEEE Transactions on Computers*, 63(9):2174–2186, 2013.

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

[166] Carlos Lozano-Garzon, Miguel Camelo, Pere Vila, and Yezid Donoso. A multi-objective routing algorithm for wireless mesh network in a smart cities environment. *Journal of Networks*, 10 (1):60, 2015.

[167] DK Lobiyal, Sunita Prasad, et al. An elitist nondominated sorting genetic algorithm for qos multicast routing in wireless networks. *Swarm and Evolutionary Computation*, 33:85–92, 2017.

[168] Lin Lin and Mitsuo Gen. An effective evolutionary approach for bicriteria shortest path routing problems. *IEEJ Transactions on Electronics, Information and Systems*, 128(3):416–423, 2008.

[169] Eckart Zitzler and Lothar Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE transactions on Evolutionary Computation*, 3(4): 257–271, 1999.

[170] Eckart Zitzler and Lothar Thiele. Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Case Study. In *International conference on parallel problem solving from nature*, pages 292–301. Springer, 1998.

[171] Michael Lahanas, Natasa Milickovic, Dimos Baltas, and Nikolaos Zamboglou. Application of Multiobjective Evolutionary Algorithms for Dose Optimization Problems in Brachytherapy. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 574–587. Springer, 2001.

[172] AT&T - Statistics & Facts. https://www.statista.com/topics/1252/atundt/. Last accessed: 2020-07-10.

[173] CAIDA. Topology Research. http://www.caida.org/research/topology/. Last accessed: 2020-07-10.

[174] Bradley Huffaker, Marina Fomenkov, et al. Internet Topology Data Comparison. Technical report, Cooperative Association for Internet Data Analysis (CAIDA), 2012.

[175] D Plummer, B Lheureux, M Cantara, and T Bova. Cloud Services Brokerage is Dominated by Three Primary Roles. *Gartner Research Note G*, 226509:23, 2011.

[176] Evangelos Markakis, Anargyros Sideris, George Alexiou, Athina Bourdena, Evangelos Pallis, George Mastorakis, and Constandinos X Mavromoustakis. A Virtual Network Functions Brokering Mechanism. In *2016 International Conference on Telecommunications and Multimedia (TEMU)*, pages 1–5. IEEE, 2016.

[177] Imane Haddar, Brahim Raouyane, and Mostafa Bellafkih. Toward a Service Broker for Telecom Service Integration in IMS Network. In *2016 International Conference on Electrical and Information Technologies (ICEIT)*, pages 338–342. IEEE, 2016.

[178] Yang Shuai, Zhang Qianli, and Li Xing. A Tunnel Broker based IPv6 Access System for a Small Scale Network with IPv4 Upstream. In *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference*, pages 206–210. IEEE, 2016.

[179] David Griffin, Jason Spencer, Jonas Griem, Mohamed Boucadair, Pierrick Morand, Michael Howarth, Ning Wang, George Pavlou, Abolghasem Asgari, and Panos Georgatsos. Interdomain Routing through QoS-Class Planes. *IEEE Communications Magazine*, 45 (2):88–95, 2007.

[180] Xavi Masip-Bruin, Marcelo Yannuzzi, Rene Serral-Gracia, Jordi Domingo-Pascual, Jose Enriquez-Gabeiras, Maria Angeles

Callejo, Michel Diaz, Florin Racaru, Giovanni Stea, Enzo Mingozzi, et al. The EuQoS System: A Solution for QoS Routing in Heterogeneous Networks. *IEEE Communications Magazine*, 45(2): 96–103, 2007.

[181] Marcelo Yannuzzi, Alexandre Fonte, Xavier Masip-Bruin, Edmundo Monteiro, Sergi Sànchez-López, Marilia Curado, and Jordi Domingo-Pascual. A Proposal for Inter-Domain QoS Routing based on Distributed Overlay Entities and QBGP. In *Quality of Service in the Emerging Networking Panorama*, pages 257–267. Springer, 2004.

[182] Akmal Khan, Taekyoung Kwon, Hyun-chul Kim, and Yanghee Choi. As-level Topology Collection through Looking Glass Servers. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 235–242, 2013.

[183] Gonca Gürsun, Natali Ruchansky, Evimaria Terzi, and Mark Crovella. Routing State Distance: A Path-based Metric for Network Analysis. In *Proceedings of the 2012 Internet Measurement Conference*, pages 239–252, 2012.

[184] Priya Mahadevan, Dmitri Krioukov, Marina Fomenkov, Xenofontas Dimitropoulos, KC Claffy, and Amin Vahdat. The Internet AS-level Topology: Three Data Sources and One Definitive Metric. *ACM SIGCOMM Computer Communication Review*, 36(1):17–26, 2006.

[185] Yihua He, Georgos Siganos, Michalis Faloutsos, and Srikanth Krishnamurthy. Lord of the Links: A Framework for Discovering Missing Links in the Internet Topology. *IEEE/ACM Transactions On Networking*, 17(2):391–404, 2008.

[186] Ricardo Oliveira, Dan Pei, Walter Willinger, Beichuan Zhang, and Lixia Zhang. The (in) Completeness of the Observed Internet AS-level Structure. *IEEE/ACM Transactions on Networking*, 18(1): 109–122, 2009.

[187] F Begtasevic and P Van Mieghem. Measurements of the Hopcount in Internet. In *PAM2001, A workshop on Passive and Active Measurements, Amsterdam, the Netherlands, April 23-24, 2001*, 2001.

[188] Ramaswamy Ramaswamy, Ning Weng, and Tilman Wolf. Characterizing Network Processing Delay. In *IEEE Global Telecommunica-*

tions Conference, 2004. GLOBECOM'04., volume 3, pages 1629–1634. IEEE, 2004.

[189] Amgad Zeitoun, Chen-Nee Chuah, Supratik Bhattacharyya, and Christophe Diot. An AS-Level Study of Internet Path Delay Characteristics. In *IEEE Global Telecommunications Conference, 2004. GLOBECOM'04.*, volume 3, pages 1480–1484. IEEE, 2004.

[190] Baek-Young Choi, Sue Moon, Zhi-Li Zhang, Konstantina Papagiannaki, and Christophe Diot. Analysis of Point-to-Point Packet Delay in an Operational Network. *Computer networks*, 51(13): 3812–3827, 2007.

[191] Patrik Carlsson, Doru Constantinescu, Adrian Popescu, Markus Fiedler, and Arne A Nilsson. Delay Performance in IP Routers. In *2nd International Working Conference (HET-NETs' 04)*, 2004.

[192] Habiba Akter and Chris Phillips. Are internet tunnels worthwhile? In *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*, pages 1–6. IEEE, 2018.

[193] Habia Akter and Chris Phillips. Tunnelling the internet. *Journal of Telecommunications and the Digital Economy*, 7(1), 2019.

[194] Masaharu Munetomo. An Adaptive Network Routing Algorithm Employing Path Genetic Operators. In *Proc. 7th International Conference on Genetic Algorithms*, pages 643–649. Morgan Kaufmann, 1997.

[195] Masaharu Munetomo, Yoshiaki Takai, and Yoshiharu Sato. An Intelligent Network Routing Algorithm by a Genetic Algorithm. In *ICONIP (1)*, pages 547–550, 1997.

[196] Masaharu Munetomo, Yoshiaki Takai, and Yoshiharu Sato. A Migration Scheme for the Genetic Adaptive Routing Algorithm. In *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)*, volume 3, pages 2774–2779. IEEE, 1998.

[197] Masaharu Munetomo. Designing Genetic Algorithms for Adapting Routing Algorithms in the Internet. In *Proceedings of GECCO*, volume 99, 1999.

[198] M. Munetomo, N. Yamaguchi, K. Akama, and Y. Sato. Empirical investigations on the genetic adaptive routing algorithm in the internet. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, volume 2, pages 1236–1243 vol. 2, 2001. doi: 10.1109/CEC.2001.934332.

[199] Xavier Hue. Genetic Algorithms for Optimization: Background and Applications. *Edinburgh Parallel Computing Centre*, 10, 1997.

[200] George Harik, Erick Cantú-Paz, David E Goldberg, and Brad L Miller. The Gambler's Ruin Problem, Genetic Algorithms, and the Sizing of Populations. *Evolutionary Computation*, 7(3):231–253, 1999.

[201] William G Macready and David H Wolpert. Bandit Problems and the Exploration/ Exploitation Tradeoff. *IEEE Transactions on evolutionary computation*, 2(1):2–22, 1998.

[202] David E Goldberg, Kalyanmoy Deb, James H Clark, et al. Genetic Algorithms, Noise, and the Sizing of Populations. *COMPLEX SYSTEMS-CHAMPAIGN-*, 6:333–333, 1992.

[203] Jun Inagaki, Miki Haseyama, and Hideo Kitajima. A Genetic Algorithm for Determining Multiple Routes and Its Applications. In *1999 IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 6, pages 137–140. IEEE, 1999.

[204] Seth Pettie and Vijaya Ramachandran. Computing shortest paths with comparisons and additions. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 267–276, 2002.

[205] Seth Pettie and Vijaya Ramachandran. Computing undirected shortest paths with comparisons and additions. Technical report, Tech Report TR01-12, Univ. of Texas at Austin, 2001.

[206] Peter Mooney and Adam Winstanley. An Evolutionary Algorithm for Multicriteria path Optimization Problems. *International Journal of Geographical Information Science*, 20(4):401–423, 2006.

[207] SB Pattnaik, S Mohan, and VM Tom. Urban bus transit route network design using genetic algorithm. *Journal of transportation engineering*, 124(4):368–375, 1998.

[208] Vlasis K Koumousis and Panos G Georgiou. Genetic algorithms in discrete optimization of steel truss roofs. *Journal of Computing in Civil Engineering*, 8(3):309–325, 1994.

[209] Chinmoy Ghorai, Swapan Shakhari, and Indrajit Banerjee. A spea-based multimetric routing protocol for intelligent transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[210] Germán A Montoya, Horacio Riveros Ardila, Sebastián Rivera Díaz, and Yezid Donoso. Multi-objective evolutive algorithm for efficient coverage problem of wsn using a spea approach. In *Proceedings of the 2011 international conference on applied, numerical and computational mathematics, and Proceedings of the 2011 international conference on Computers, digital communications and computing*, pages 27–32. World Scientific and Engineering Academy and Society (WSEAS), 2011.

[211] Theni Koduvilarpatti. Strength Pareto Evolutionary Algorithm based Multi-objective Optimization for Shortest Path Routing Problem in Computer Networks. *Journal of Computer Science*, 7 (1):17–26, 2011.

[212] Uri Yael Rozenworcel and Benjamín Barán. Routing in periodic dynamic networks using a multi-objective evolutionary algorithm. 2004.

[213] Habiba Akter. Tunnelling the Internet. https://womencourage.acm.org/2019/wp-content/uploads/2019/07/womENcourage_2019_paper_105.pdf, 2019.

[214] Habiba Akter. Genetic algorithm for least cost routing in the network.

Appendices

Appendix A

Initial Tool

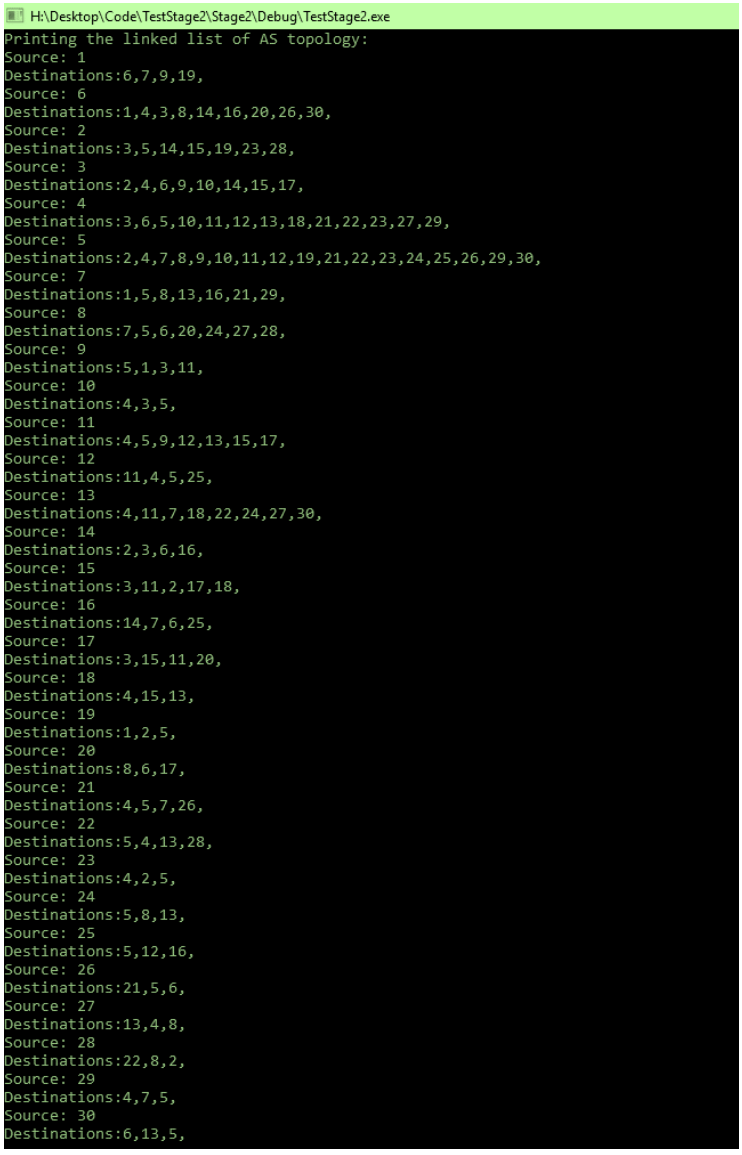
PFP Generated AS Topology of 30 ASes

Node-1 Node-2
Node-2 Node-3
Node-1 Node-4
Node-1 Node-3
Node-1 Node-5
Node-5 Node-6
Node-6 Node-1
Node-6 Node-7
Node-5 Node-3
Artificial-Node-0 Node-5
Artificial-Node-0 Node-1
Artificial-Node-0 Node-4
Artificial-Node-1 Node-1
Artificial-Node-1 Node-6
Artificial-Node-1 Node-5
Artificial-Node-2 Node-1
Artificial-Node-2 Artificial-Node-1
Artificial-Node-2 Node-5
Artificial-Node-3 Node-5
Artificial-Node-3 Node-1
Artificial-Node-3 Node-6
Artificial-Node-4 Node-1
Artificial-Node-4 Node-6
Artificial-Node-4 Artificial-Node-0
Artificial-Node-5 Artificial-Node-4

Artificial-Node-5 Artificial-Node-2
Artificial-Node-5 Node-1
Artificial-Node-6 Artificial-Node-2
Artificial-Node-6 Artificial-Node-0
Artificial-Node-6 Node-6
Artificial-Node-7 Artificial-Node-0
Artificial-Node-7 Node-6
Artificial-Node-7 Node-5
Artificial-Node-8 Node-5
Artificial-Node-8 Node-1
Artificial-Node-8 Artificial-Node-6
Artificial-Node-9 Artificial-Node-7
Artificial-Node-9 Node-5
Artificial-Node-9 Node-1
Artificial-Node-10 Node-6
Artificial-Node-10 Artificial-Node-8
Artificial-Node-10 Node-2
Artificial-Node-11 Node-6
Artificial-Node-11 Node-1
Artificial-Node-11 Node-5
Artificial-Node-12 Node-1
Artificial-Node-12 Node-6
Artificial-Node-12 Node-4
Artificial-Node-13 Node-6
Artificial-Node-13 Artificial-Node-3
Artificial-Node-13 Node-1
Artificial-Node-14 Node-6
Artificial-Node-14 Artificial-Node-0
Artificial-Node-14 Node-5
Artificial-Node-15 Node-6
Artificial-Node-15 Node-1
Artificial-Node-15 Artificial-Node-0
Artificial-Node-16 Artificial-Node-6
Artificial-Node-16 Node-1
Artificial-Node-16 Artificial-Node-2
Artificial-Node-17 Node-5
Artificial-Node-17 Node-1
Artificial-Node-17 Node-6
Artificial-Node-18 Artificial-Node-4
Artificial-Node-18 Node-6

Artificial-Node-18 Artificial-Node-7
Artificial-Node-19 Node-1
Artificial-Node-19 Node-7
Artificial-Node-19 Artificial-Node-17
Artificial-Node-20 Artificial-Node-4
Artificial-Node-20 Node-5
Artificial-Node-20 Artificial-Node-9
Artificial-Node-21 Node-1
Artificial-Node-21 Node-4
Artificial-Node-21 Node-5
Artificial-Node-22 Node-5
Artificial-Node-22 Node-1
Artificial-Node-22 Artificial-Node-4

Console Output for AS Topology



Output for AS Topology

ASBR topology for the AS-level topology of A

Source BR: 30_5
Destination BRs: 30_6, 30_13, 5_30,
Source BR: 30_13

Destination BRs: 30_6, 30_5, 13_30,
Source BR: 30_6
Destination BRs: 30_13, 30_5, 6_30,
Source BR: 29_5
Destination BRs: 29_4, 29_7, 5_29,
Source BR: 29_7
Destination BRs: 29_4, 29_5, 7_29,
Source BR: 29_4
Destination BRs: 29_7, 29_5, 4_29,
Source BR: 28_2
Destination BRs: 28_22, 28_8, 2_28,
Source BR: 28_8
Destination BRs: 28_22, 28_2, 8_28,
Source BR: 28_22
Destination BRs: 28_8, 28_2, 22_28,
Source BR: 27_8
Destination BRs: 27_13, 27_4, 8_27,
Source BR: 27_4
Destination BRs: 27_13, 27_8, 4_27,
Source BR: 27_13
Destination BRs: 27_4, 27_8, 13_27,
Source BR: 26_6
Destination BRs: 26_21, 26_5, 6_26,
Source BR: 26_5
Destination BRs: 26_21, 26_6, 5_26,
Source BR: 26_21
Destination BRs: 26_5, 26_6, 21_26,
Source BR: 25_16
Destination BRs: 25_5, 25_12, 16_25,
Source BR: 25_12
Destination BRs: 25_5, 25_16, 12_25,
Source BR: 25_5
Destination BRs: 25_12, 25_16, 5_25,
Source BR: 24_13
Destination BRs: 24_5, 24_8, 13_24,
Source BR: 24_8
Destination BRs: 24_5, 24_13, 8_24,
Source BR: 24_5
Destination BRs: 24_8, 24_13, 5_24,
Source BR: 23_5

Destination BRs: 23_4, 23_2, 5_23,
Source BR: 23_2
Destination BRs: 23_4, 23_5, 2_23,
Source BR: 23_4
Destination BRs: 23_2, 23_5, 4_23,
Source BR: 22_28
Destination BRs: 22_5, 22_4, 22_13, 28_22,
Source BR: 22_13
Destination BRs: 22_5, 22_4, 22_28, 13_22,
Source BR: 22_4
Destination BRs: 22_5, 22_13, 22_28, 4_22,
Source BR: 22_5
Destination BRs: 22_4, 22_13, 22_28, 5_22,
Source BR: 21_26
Destination BRs: 21_4, 21_5, 21_7, 26_21,
Source BR: 21_7
Destination BRs: 21_4, 21_5, 21_26, 7_21,
Source BR: 21_5
Destination BRs: 21_4, 21_7, 21_26, 5_21,
Source BR: 21_4
Destination BRs: 21_5, 21_7, 21_26, 4_21,
Source BR: 20_17
Destination BRs: 20_8, 20_6, 17_20,
Source BR: 20_6
Destination BRs: 20_8, 20_17, 6_20,
Source BR: 20_8
Destination BRs: 20_6, 20_17, 8_20,
Source BR: 19_5
Destination BRs: 19_1, 19_2, 5_19,
Source BR: 19_2
Destination BRs: 19_1, 19_5, 2_19,
Source BR: 19_1
Destination BRs: 19_2, 19_5, 1_19,
Source BR: 18_13
Destination BRs: 18_4, 18_15, 13_18,
Source BR: 18_15
Destination BRs: 18_4, 18_13, 15_18,
Source BR: 18_4
Destination BRs: 18_15, 18_13, 4_18,
Source BR: 17_20

Destination BRs: 17_3, 17_15, 17_11, 20_17,
Source BR: 17_11
Destination BRs: 17_3, 17_15, 17_20, 11_17,
Source BR: 17_15
Destination BRs: 17_3, 17_11, 17_20, 15_17,
Source BR: 17_3
Destination BRs: 17_15, 17_11, 17_20, 3_17,
Source BR: 16_25
Destination BRs: 16_14, 16_7, 16_6, 25_16,
Source BR: 16_6
Destination BRs: 16_14, 16_7, 16_25, 6_16,
Source BR: 16_7
Destination BRs: 16_14, 16_6, 16_25, 7_16,
Source BR: 16_14
Destination BRs: 16_7, 16_6, 16_25, 14_16,
Source BR: 15_18
Destination BRs: 15_3, 15_11, 15_2, 15_17, 18_15,
Source BR: 15_17
Destination BRs: 15_3, 15_11, 15_2, 15_18, 17_15,
Source BR: 15_2
Destination BRs: 15_3, 15_11, 15_17, 15_18, 2_15,
Source BR: 15_11
Destination BRs: 15_3, 15_2, 15_17, 15_18, 11_15,
Source BR: 15_3
Destination BRs: 15_11, 15_2, 15_17, 15_18, 3_15,
Source BR: 14_16
Destination BRs: 14_2, 14_3, 14_6, 16_14,
Source BR: 14_6
Destination BRs: 14_2, 14_3, 14_16, 6_14,
Source BR: 14_3
Destination BRs: 14_2, 14_6, 14_16, 3_14,
Source BR: 14_2
Destination BRs: 14_3, 14_6, 14_16, 2_14,
Source BR: 13_30
Destination BRs: 13_4, 13_11, 13_7, 13_18, 13_22, 13_24, 13_27, 30_13
Source BR: 13_27
Destination BRs: 13_4, 13_11, 13_7, 13_18, 13_22, 13_24, 13_30, 27_13,
Source BR: 13_24
Destination BRs: 13_4, 13_11, 13_7, 13_18, 13_22, 13_27, 13_30, 24_13,
Source BR: 13_22

Destination BRs: 13_4, 13_11, 13_7, 13_18, 13_24, 13_27, 13_30, 22_13,
Source BR: 13_18
Destination BRs: 13_4, 13_11, 13_7, 13_22, 13_24, 13_27, 13_30, 18_13,
Source BR: 13_7
Destination BRs: 13_4, 13_11, 13_18, 13_22, 13_24, 13_27, 13_30, 7_13,
Source BR: 13_11
Destination BRs: 13_4, 13_7, 13_18, 13_22, 13_24, 13_27, 13_30, 11_13,
Source BR: 13_4
Destination BRs: 13_11, 13_7, 13_18, 13_22, 13_24, 13_27, 13_30, 4_13,
Source BR: 12_25
Destination BRs: 12_11, 12_4, 12_5, 25_12,
Source BR: 12_5
Destination BRs: 12_11, 12_4, 12_25, 5_12,
Source BR: 12_4
Destination BRs: 12_11, 12_5, 12_25, 4_12,
Source BR: 12_11
Destination BRs: 12_4, 12_5, 12_25, 11_12,
Source BR: 11_17
Destination BRs: 11_4, 11_5, 11_9, 11_12, 11_13, 11_15, 17_11,
Source BR: 11_15
Destination BRs: 11_4, 11_5, 11_9, 11_12, 11_13, 11_17, 15_11,
Source BR: 11_13
Destination BRs: 11_4, 11_5, 11_9, 11_12, 11_15, 11_17, 13_11,
Source BR: 11_12
Destination BRs: 11_4, 11_5, 11_9, 11_13, 11_15, 11_17, 12_11,
Source BR: 11_9
Destination BRs: 11_4, 11_5, 11_12, 11_13, 11_15, 11_17, 9_11,
Source BR: 11_5
Destination BRs: 11_4, 11_9, 11_12, 11_13, 11_15, 11_17, 5_11,
Source BR: 11_4
Destination BRs: 11_5, 11_9, 11_12, 11_13, 11_15, 11_17, 4_11,
Source BR: 10_5
Destination BRs: 10_4, 10_3, 5_10,
Source BR: 10_3
Destination BRs: 10_4, 10_5, 3_10,
Source BR: 10_4
Destination BRs: 10_3, 10_5, 4_10,
Source BR: 9_11
Destination BRs: 9_5, 9_1, 9_3, 11_9,
Source BR: 9_3

Destination BRs: 9_5, 9_1, 9_11, 3_9,
Source BR: 9_1
Destination BRs: 9_5, 9_3, 9_11, 1_9,
Source BR: 9_5
Destination BRs: 9_1, 9_3, 9_11, 5_9,
Source BR: 8_28
Destination BRs: 8_7, 8_5, 8_6, 8_20, 8_24, 8_27, 28_8,
Source BR: 8_27
Destination BRs: 8_7, 8_5, 8_6, 8_20, 8_24, 8_28, 27_8,
Source BR: 8_24
Destination BRs: 8_7, 8_5, 8_6, 8_20, 8_27, 8_28, 24_8,
Source BR: 8_20
Destination BRs: 8_7, 8_5, 8_6, 8_24, 8_27, 8_28, 20_8,
Source BR: 8_6
Destination BRs: 8_7, 8_5, 8_20, 8_24, 8_27, 8_28, 6_8,
Source BR: 8_5
Destination BRs: 8_7, 8_6, 8_20, 8_24, 8_27, 8_28, 5_8,
Source BR: 8_7
Destination BRs: 8_5, 8_6, 8_20, 8_24, 8_27, 8_28, 7_8,
Source BR: 7_29
Destination BRs: 7_1, 7_5, 7_8, 7_13, 7_16, 7_21, 29_7,
Source BR: 7_21
Destination BRs: 7_1, 7_5, 7_8, 7_13, 7_16, 7_29, 21_7,
Source BR: 7_16
Destination BRs: 7_1, 7_5, 7_8, 7_13, 7_21, 7_29, 16_7,
Source BR: 7_13
Destination BRs: 7_1, 7_5, 7_8, 7_16, 7_21, 7_29, 13_7,
Source BR: 7_8
Destination BRs: 7_1, 7_5, 7_13, 7_16, 7_21, 7_29, 8_7,
Source BR: 7_5
Destination BRs: 7_1, 7_8, 7_13, 7_16, 7_21, 7_29, 5_7,
Source BR: 7_1
Destination BRs: 7_5, 7_8, 7_13, 7_16, 7_21, 7_29, 1_7,
Source BR: 5_30
Destination BRs: 5_2, 5_4, 5_7, 5_8, 5_9, 5_10, 5_11, 5_12, 5_19, 5_21, 5_22, 5_23, 5_24, 5_25, 5_26, 5_29, 30_5,
Source BR: 5_29
Destination BRs: 5_2, 5_4, 5_7, 5_8, 5_9, 5_10, 5_11, 5_12, 5_19, 5_21, 5_22, 5_23, 5_24, 5_25, 5_26, 5_30, 29_5,
Source BR: 5_26

Destination BRs: 5_2, 5_4, 5_7, 5_8, 5_9, 5_10, 5_11, 5_12, 5_19, 5_21, 5_22, 5_23, 5_24, 5_25, 5_29, 5_30, 26_5,
Source BR: 5_25
Destination BRs: 5_2, 5_4, 5_7, 5_8, 5_9, 5_10, 5_11, 5_12, 5_19, 5_21, 5_22, 5_23, 5_24, 5_26, 5_29, 5_30, 25_5,
Source BR: 5_24
Destination BRs: 5_2, 5_4, 5_7, 5_8, 5_9, 5_10, 5_11, 5_12, 5_19, 5_21, 5_22, 5_23, 5_25, 5_26, 5_29, 5_30, 24_5,
Source BR: 5_23
Destination BRs: 5_2, 5_4, 5_7, 5_8, 5_9, 5_10, 5_11, 5_12, 5_19, 5_21, 5_22, 5_24, 5_25, 5_26, 5_29, 5_30, 23_5,
Source BR: 5_22
Destination BRs: 5_2, 5_4, 5_7, 5_8, 5_9, 5_10, 5_11, 5_12, 5_19, 5_21, 5_23, 5_24, 5_25, 5_26, 5_29, 5_30, 22_5,
Source BR: 5_21
Destination BRs: 5_2, 5_4, 5_7, 5_8, 5_9, 5_10, 5_11, 5_12, 5_19, 5_22, 5_23, 5_24, 5_25, 5_26, 5_29, 5_30, 21_5,
Source BR: 5_19
Destination BRs: 5_2, 5_4, 5_7, 5_8, 5_9, 5_10, 5_11, 5_12, 5_21, 5_22, 5_23, 5_24, 5_25, 5_26, 5_29, 5_30, 19_5,
Source BR: 5_12
Destination BRs: 5_2, 5_4, 5_7, 5_8, 5_9, 5_10, 5_11, 5_19, 5_21, 5_22, 5_23, 5_24, 5_25, 5_26, 5_29, 5_30, 12_5,
Source BR: 5_11
Destination BRs: 5_2, 5_4, 5_7, 5_8, 5_9, 5_10, 5_12, 5_19, 5_21, 5_22, 5_23, 5_24, 5_25, 5_26, 5_29, 5_30, 11_5,
Source BR: 5_10
Destination BRs: 5_2, 5_4, 5_7, 5_8, 5_9, 5_11, 5_12, 5_19, 5_21, 5_22, 5_23, 5_24, 5_25, 5_26, 5_29, 5_30, 10_5,
Source BR: 5_9
Destination BRs: 5_2, 5_4, 5_7, 5_8, 5_10, 5_11, 5_12, 5_19, 5_21, 5_22, 5_23, 5_24, 5_25, 5_26, 5_29, 5_30, 9_5,
Source BR: 5_8
Destination BRs: 5_2, 5_4, 5_7, 5_9, 5_10, 5_11, 5_12, 5_19, 5_21, 5_22, 5_23, 5_24, 5_25, 5_26, 5_29, 5_30, 8_5,
Source BR: 5_7
Destination BRs: 5_2, 5_4, 5_8, 5_9, 5_10, 5_11, 5_12, 5_19, 5_21, 5_22, 5_23, 5_24, 5_25, 5_26,

Least cost path calculated from source AS 1 to each of the AS topology of Appendix A

Your source domain is: 1
Your destination domain is: 6
distance: 1
path: 1_6 > 6_1
Your source domain is: 1
Your destination domain is: 2
distance: 3
path: 1_19 > 19_1 > 19_2 > 2_19
Your source domain is: 1
Your destination domain is: 3
distance: 6
path: 1_6 > 6_1 > 6_3 > 3_6
Your source domain is: 1
Your destination domain is: 4
distance: 6
path: 1_6 > 6_1 > 6_4 > 4_6
Your source domain is: 1
Your destination domain is: 5
distance: 3
path: 1_19 > 19_1 > 19_5 > 5_19
Your source domain is: 1 Your destination domain is: 7 distance: 1
path: 1_7 > 7_1
Your source domain is: 1
Your destination domain is: 8
distance: 6
path: 1_6 > 6_1 > 6_8 > 8_6
Your source domain is: 1
Your destination domain is: 9
distance: 1
path: 1_9 > 9_1
Your source domain is: 1
Your destination domain is: 10
distance: 8
path: 1_19 > 19_1 > 19_5 > 5_19 > 5_10 > 10_5
Your source domain is: 1
Your destination domain is: 11

distance: 6
path: 1_9 > 9_1 > 9_11 > 11_9
    Your source domain is: 1
Your destination domain is: 12
distance: 8
path: 1_19 > 19_1 > 19_5 > 5_19 > 5_12 > 12_5
    Your source domain is: 1
Your destination domain is: 13
distance: 6
path: 1_7 > 7_1 > 7_13 > 13_7
    Your source domain is: 1
Your destination domain is: 14
distance: 6
path: 1_6 > 6_1 > 6_14 > 14_6
    Your source domain is: 1
Your destination domain is: 15
distance: 8
path: 1_19 > 19_1 > 19_2 > 2_19 > 2_15 > 15_2
    Your source domain is: 1
Your destination domain is: 16
distance: 6
path: 1_6 > 6_1 > 6_16 > 16_6
    Your source domain is: 1
Your destination domain is: 17
distance: 11
path: 1_6 > 6_1 > 6_3 > 3_6 > 3_17 > 17_3
    Your source domain is: 1
Your destination domain is: 18
distance: 11
path: 1_6 > 6_1 > 6_4 > 4_6 > 4_18 > 18_4
    Your source domain is: 1
Your destination domain is: 19
distance: 1
path: 1_19 > 19_1
    Your source domain is: 1
Your destination domain is: 20
distance: 6
path: 1_6 > 6_1 > 6_20 > 20_6
    Your source domain is: 1
Your destination domain is: 21

distance: 6
path: 1_7 > 7_1 > 7_21 > 21_7
Your source domain is: 1
Your destination domain is: 22
distance: 8
path: 1_19 > 19_1 > 19_5 > 5_19 > 5_22 > 22_5
Your source domain is: 1
Your destination domain is: 23
distance: 8
path: 1_19 > 19_1 > 19_2 > 2_19 > 2_23 > 23_2
Your source domain is: 1
Your destination domain is: 24
distance: 8
path: 1_19 > 19_1 > 19_5 > 5_19 > 5_24 > 24_5
Your source domain is: 1
Your destination domain is: 25
distance: 8
path: 1_19 > 19_1 > 19_5 > 5_19 > 5_25 > 25_5
Your source domain is: 1
Your destination domain is: 26
distance: 6
path: 1_6 > 6_1 > 6_26 > 26_6
Your source domain is: 1
Your destination domain is: 27
distance: 11
path: 1_6 > 6_1 > 6_4 > 4_6 > 4_27 > 27_4
Your source domain is: 1
Your destination domain is: 28
distance: 8
path: 1_19 > 19_1 > 19_2 > 2_19 > 2_28 > 28_2
Your source domain is: 1
Your destination domain is: 29
distance: 6
path: 1_7 > 7_1 > 7_29 > 29_7
Your source domain is: 1
Your destination domain is: 30
distance: 6
path: 1_6 > 6_1 > 6_30 > 30_6

AS Topologies for node degree 3

Topology 1

Node-1 Node-2
Node-2 Node-3
Node-1 Node-4
Node-1 Node-3
Node-1 Node-5
Node-5 Node-6
Node-6 Node-1
Node-6 Node-7
Node-5 Node-3
Artificial-Node-0 Node-5
Artificial-Node-0 Node-1
Artificial-Node-0 Node-4
Artificial-Node-1 Node-1
Artificial-Node-1 Node-6
Artificial-Node-1 Node-5
Artificial-Node-2 Node-1
Artificial-Node-2 Artificial-Node-1
Artificial-Node-2 Node-5
Artificial-Node-3 Node-5
Artificial-Node-3 Node-1
Artificial-Node-3 Node-6
Artificial-Node-4 Node-1
Artificial-Node-4 Node-6
Artificial-Node-4 Artificial-Node-0
Artificial-Node-5 Artificial-Node-4
Artificial-Node-5 Artificial-Node-2
Artificial-Node-5 Node-1
Artificial-Node-6 Artificial-Node-2
Artificial-Node-6 Artificial-Node-0
Artificial-Node-6 Node-6
Artificial-Node-7 Artificial-Node-0
Artificial-Node-7 Node-6
Artificial-Node-7 Node-5
Artificial-Node-8 Node-5
Artificial-Node-8 Node-1
Artificial-Node-8 Artificial-Node-6

Artificial-Node-9 Artificial-Node-7
Artificial-Node-9 Node-5
Artificial-Node-9 Node-1
Artificial-Node-10 Node-6
Artificial-Node-10 Artificial-Node-8
Artificial-Node-10 Node-2
Artificial-Node-11 Node-6
Artificial-Node-11 Node-1
Artificial-Node-11 Node-5
Artificial-Node-12 Node-1
Artificial-Node-12 Node-6
Artificial-Node-12 Node-4
Artificial-Node-13 Node-6
Artificial-Node-13 Artificial-Node-3
Artificial-Node-13 Node-1
Artificial-Node-14 Node-6
Artificial-Node-14 Artificial-Node-0
Artificial-Node-14 Node-5
Artificial-Node-15 Node-6
Artificial-Node-15 Node-1
Artificial-Node-15 Artificial-Node-0
Artificial-Node-16 Artificial-Node-6
Artificial-Node-16 Node-1
Artificial-Node-16 Artificial-Node-2
Artificial-Node-17 Node-5
Artificial-Node-17 Node-1
Artificial-Node-17 Node-6
Artificial-Node-18 Artificial-Node-4
Artificial-Node-18 Node-6
Artificial-Node-18 Artificial-Node-7
Artificial-Node-19 Node-1
Artificial-Node-19 Node-7
Artificial-Node-19 Artificial-Node-17
Artificial-Node-20 Artificial-Node-4
Artificial-Node-20 Node-5
Artificial-Node-20 Artificial-Node-9
Artificial-Node-21 Node-1
Artificial-Node-21 Node-4
Artificial-Node-21 Node-5
Artificial-Node-22 Node-5

Artificial-Node-22 Node-1
Artificial-Node-22 Artificial-Node-4

Topology 2

Node-1 Node-2
Node-2 Node-4
Node-1 Node-3
Node-1 Node-7
Node-2 Node-6
Node-3 Node-7
Node-5 Node-1
Node-7 Node-5
Node-4 Node-6
Artificial-Node-0 Node-1
Artificial-Node-0 Node-4
Artificial-Node-0 Node-6
Artificial-Node-1 Node-7
Artificial-Node-1 Node-1
Artificial-Node-1 Node-2
Artificial-Node-2 Artificial-Node-0
Artificial-Node-2 Node-2
Artificial-Node-2 Artificial-Node-1
Artificial-Node-3 Node-5
Artificial-Node-3 Node-2
Artificial-Node-3 Artificial-Node-1
Artificial-Node-4 Node-2
Artificial-Node-4 Node-1
Artificial-Node-4 Node-4
Artificial-Node-5 Node-2
Artificial-Node-5 Node-6
Artificial-Node-5 Artificial-Node-1
Artificial-Node-6 Artificial-Node-1
Artificial-Node-6 Node-7
Artificial-Node-6 Artificial-Node-2
Artificial-Node-7 Node-7
Artificial-Node-7 Artificial-Node-2
Artificial-Node-7 Artificial-Node-1
Artificial-Node-8 Artificial-Node-2
Artificial-Node-8 Artificial-Node-7

Artificial-Node-8 Node-4
Artificial-Node-9 Node-7
Artificial-Node-9 Artificial-Node-3
Artificial-Node-9 Artificial-Node-5
Artificial-Node-10 Artificial-Node-7
Artificial-Node-10 Node-6
Artificial-Node-10 Node-4
Artificial-Node-11 Node-2
Artificial-Node-11 Artificial-Node-1
Artificial-Node-11 Artificial-Node-8
Artificial-Node-12 Artificial-Node-2
Artificial-Node-12 Artificial-Node-1
Artificial-Node-12 Node-7
Artificial-Node-13 Artificial-Node-7
Artificial-Node-13 Node-7
Artificial-Node-13 Node-6
Artificial-Node-14 Node-4
Artificial-Node-14 Node-2
Artificial-Node-14 Artificial-Node-1
Artificial-Node-15 Artificial-Node-1
Artificial-Node-15 Node-2
Artificial-Node-15 Artificial-Node-5
Artificial-Node-16 Node-6
Artificial-Node-16 Node-2
Artificial-Node-16 Artificial-Node-13
Artificial-Node-17 Artificial-Node-8
Artificial-Node-17 Artificial-Node-4
Artificial-Node-17 Artificial-Node-11
Artificial-Node-18 Artificial-Node-12
Artificial-Node-18 Node-1
Artificial-Node-18 Node-2
Artificial-Node-19 Artificial-Node-2
Artificial-Node-19 Node-7
Artificial-Node-19 Node-2
Artificial-Node-20 Node-5
Artificial-Node-20 Artificial-Node-13
Artificial-Node-20 Node-2
Artificial-Node-21 Artificial-Node-2
Artificial-Node-21 Artificial-Node-1
Artificial-Node-21 Node-5

Artificial-Node-22 Artificial-Node-5
Artificial-Node-22 Node-2
Artificial-Node-22 Node-6

Topology 3

Node-1 Node-3
Node-2 Node-3
Node-2 Node-4
Node-5 Node-4
Node-5 Node-2
Node-6 Node-1
Node-6 Node-7
Node-7 Node-5
Artificial-Node-0 Node-2
Artificial-Node-0 Node-6
Artificial-Node-0 Node-5
Artificial-Node-1 Node-5
Artificial-Node-1 Node-1
Artificial-Node-1 Node-2
Artificial-Node-2 Node-2
Artificial-Node-2 Node-1
Artificial-Node-2 Node-5
Artificial-Node-3 Node-2
Artificial-Node-3 Node-6
Artificial-Node-3 Node-7
Artificial-Node-4 Node-2
Artificial-Node-4 Node-6
Artificial-Node-4 Node-1
Artificial-Node-5 Node-5
Artificial-Node-5 Node-2
Artificial-Node-5 Artificial-Node-2
Artificial-Node-6 Node-5
Artificial-Node-6 Node-7
Artificial-Node-6 Node-6
Artificial-Node-7 Artificial-Node-3
Artificial-Node-7 Node-1
Artificial-Node-7 Node-5
Artificial-Node-8 Node-5
Artificial-Node-8 Artificial-Node-3

Artificial-Node-8 Artificial-Node-7
Artificial-Node-9 Node-6
Artificial-Node-9 Node-5
Artificial-Node-9 Node-1
Artificial-Node-10 Node-1
Artificial-Node-10 Node-5
Artificial-Node-10 Artificial-Node-3
Artificial-Node-11 Artificial-Node-2
Artificial-Node-11 Node-6
Artificial-Node-11 Node-2
Artificial-Node-12 Node-6
Artificial-Node-12 Node-5
Artificial-Node-12 Artificial-Node-1
Artificial-Node-13 Artificial-Node-6
Artificial-Node-13 Artificial-Node-5
Artificial-Node-13 Artificial-Node-8
Artificial-Node-14 Artificial-Node-11
Artificial-Node-14 Artificial-Node-5
Artificial-Node-14 Artificial-Node-2
Artificial-Node-15 Artificial-Node-14
Artificial-Node-15 Artificial-Node-8
Artificial-Node-15 Node-1
Artificial-Node-16 Artificial-Node-10
Artificial-Node-16 Node-5
Artificial-Node-16 Artificial-Node-6
Artificial-Node-17 Node-5
Artificial-Node-17 Artificial-Node-2
Artificial-Node-17 Node-6
Artificial-Node-18 Artificial-Node-2
Artificial-Node-18 Node-7
Artificial-Node-18 Artificial-Node-5
Artificial-Node-19 Node-2
Artificial-Node-19 Artificial-Node-13
Artificial-Node-19 Node-5
Artificial-Node-20 Artificial-Node-3
Artificial-Node-20 Artificial-Node-8
Artificial-Node-20 Node-5
Artificial-Node-21 Node-6
Artificial-Node-21 Artificial-Node-3
Artificial-Node-21 Artificial-Node-11

Artificial-Node-22 Node-6
Artificial-Node-22 Node-1
Artificial-Node-22 Artificial-Node-1

Topology 4

Node-1 Node-6
Node-2 Node-3
Node-3 Node-4
Node-4 Node-6
Node-5 Node-2
Node-5 Node-4
Node-6 Node-3
Node-7 Node-1
Node-7 Node-5
Artificial-Node-0 Node-7
Artificial-Node-0 Node-5
Artificial-Node-0 Node-6
Artificial-Node-1 Node-5
Artificial-Node-1 Node-1
Artificial-Node-1 Node-3
Artificial-Node-2 Node-4
Artificial-Node-2 Node-3
Artificial-Node-2 Node-5
Artificial-Node-3 Node-4
Artificial-Node-3 Node-5
Artificial-Node-3 Artificial-Node-1
Artificial-Node-4 Artificial-Node-3
Artificial-Node-4 Node-4
Artificial-Node-4 Node-5
Artificial-Node-5 Node-4
Artificial-Node-5 Artificial-Node-3
Artificial-Node-5 Node-7
Artificial-Node-6 Node-2
Artificial-Node-6 Node-3
Artificial-Node-6 Node-6
Artificial-Node-7 Node-3
Artificial-Node-7 Artificial-Node-3
Artificial-Node-7 Node-2

Artificial-Node-8 Artificial-Node-6
Artificial-Node-8 Node-7
Artificial-Node-8 Node-6
Artificial-Node-9 Node-3
Artificial-Node-9 Artificial-Node-7
Artificial-Node-9 Artificial-Node-3
Artificial-Node-10 Node-4
Artificial-Node-10 Artificial-Node-7
Artificial-Node-10 Artificial-Node-5
Artificial-Node-11 Node-1
Artificial-Node-11 Node-2
Artificial-Node-11 Node-5
Artificial-Node-12 Artificial-Node-0
Artificial-Node-12 Node-6
Artificial-Node-12 Artificial-Node-9
Artificial-Node-13 Node-4
Artificial-Node-13 Node-5
Artificial-Node-13 Node-7
Artificial-Node-14 Node-5
Artificial-Node-14 Node-4
Artificial-Node-14 Artificial-Node-5
Artificial-Node-15 Node-4
Artificial-Node-15 Node-2
Artificial-Node-15 Node-5
Artificial-Node-16 Node-5
Artificial-Node-16 Artificial-Node-0
Artificial-Node-16 Artificial-Node-5
Artificial-Node-17 Node-5
Artificial-Node-17 Artificial-Node-4
Artificial-Node-17 Artificial-Node-8
Artificial-Node-18 Artificial-Node-13
Artificial-Node-18 Node-5
Artificial-Node-18 Node-6
Artificial-Node-19 Artificial-Node-5
Artificial-Node-19 Node-4
Artificial-Node-19 Artificial-Node-0
Artificial-Node-20 Artificial-Node-14
Artificial-Node-20 Artificial-Node-0
Artificial-Node-20 Node-2
Artificial-Node-21 Node-4

Artificial-Node-21 Node-7
Artificial-Node-21 Node-5
Artificial-Node-22 Node-6
Artificial-Node-22 Artificial-Node-5
Artificial-Node-22 Node-5

Topology 5

Node-1 Node-3
Node-2 Node-5
Node-3 Node-4
Node-4 Node-1
Node-5 Node-3
Node-5 Node-4
Node-6 Node-2
Node-7 Node-2
Node-7 Node-4
Artificial-Node-0 Node-3
Artificial-Node-0 Node-4
Artificial-Node-0 Node-6
Artificial-Node-1 Artificial-Node-0
Artificial-Node-1 Node-4
Artificial-Node-1 Node-6
Artificial-Node-2 Node-1
Artificial-Node-2 Node-4
Artificial-Node-2 Node-3
Artificial-Node-3 Node-4
Artificial-Node-3 Node-1
Artificial-Node-3 Artificial-Node-0
Artificial-Node-4 Node-4
Artificial-Node-4 Node-2
Artificial-Node-4 Node-3
Artificial-Node-5 Artificial-Node-2
Artificial-Node-5 Node-4
Artificial-Node-5 Artificial-Node-1
Artificial-Node-6 Artificial-Node-2
Artificial-Node-6 Artificial-Node-1
Artificial-Node-6 Node-3
Artificial-Node-7 Node-2

Artificial-Node-7 Artificial-Node-2
Artificial-Node-7 Node-4
Artificial-Node-8 Artificial-Node-1
Artificial-Node-8 Artificial-Node-4
Artificial-Node-8 Artificial-Node-0
Artificial-Node-9 Node-4
Artificial-Node-9 Artificial-Node-2
Artificial-Node-9 Node-2
Artificial-Node-10 Node-2
Artificial-Node-10 Node-4
Artificial-Node-10 Artificial-Node-2
Artificial-Node-11 Node-2
Artificial-Node-11 Artificial-Node-2
Artificial-Node-11 Artificial-Node-0
Artificial-Node-12 Node-4
Artificial-Node-12 Artificial-Node-2
Artificial-Node-12 Node-5
Artificial-Node-13 Node-2
Artificial-Node-13 Node-4
Artificial-Node-13 Artificial-Node-3
Artificial-Node-14 Artificial-Node-6
Artificial-Node-14 Node-4
Artificial-Node-14 Artificial-Node-3
Artificial-Node-15 Node-4
Artificial-Node-15 Node-2
Artificial-Node-15 Artificial-Node-2
Artificial-Node-16 Artificial-Node-13
Artificial-Node-16 Node-3
Artificial-Node-16 Artificial-Node-0
Artificial-Node-17 Artificial-Node-2
Artificial-Node-17 Artificial-Node-13
Artificial-Node-17 Artificial-Node-16
Artificial-Node-18 Node-1
Artificial-Node-18 Artificial-Node-4
Artificial-Node-18 Node-4
Artificial-Node-19 Node-6
Artificial-Node-19 Node-4
Artificial-Node-19 Artificial-Node-13
Artificial-Node-20 Node-4
Artificial-Node-20 Artificial-Node-4

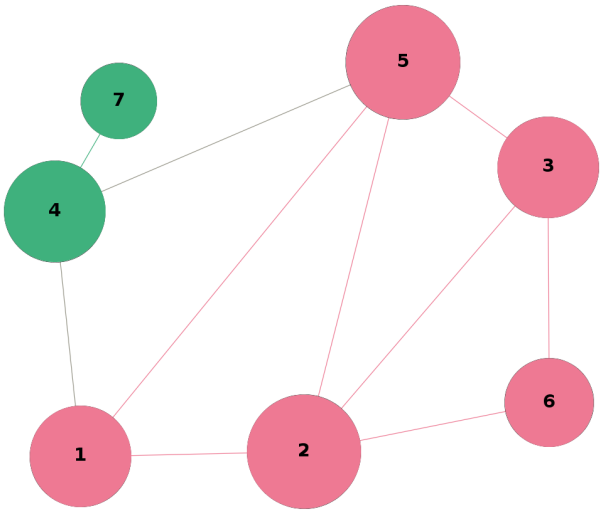
Artificial-Node-20 Artificial-Node-3
Artificial-Node-21 Node-4
Artificial-Node-21 Node-2
Artificial-Node-21 Artificial-Node-18
Artificial-Node-22 Node-4
Artificial-Node-22 Artificial-Node-2
Artificial-Node-22 Artificial-Node-0

Appendix B

Path Computation Algorithm for Tunnels (PCAT)

Validation

A Small Topology of 7 ASes



A Sample Topology of 7 ASes

All Possible Paths from AS1 B

Possible Paths from AS1 to Other ASes in the Topology

Source	Destination	Possible Path(s)
1	3	1_2>2_1>2_3>3_2
1	3	1_2>2_1>2_5>5_2>5_3>3_5
1	3	1_2>2_1>2_6>6_2>6_3>3_6
1	3	1_5>5_1>5_3>3_5
1	3	1_5>5_1>5_2>2_5>2_3>3_2
1	3	1_5>5_1>5_2>2_5>2_6>6_2>6_3>3_6
1	3	1_4>4_1>4_5>5_4>5_3>3_5
1	3	1_4>4_1>4_5>5_4>5_2>2_5>2_3>3_2
1	3	1_4>4_1>4_5>5_4>5_2>2_5>2_6>6_2>6_3>3_6
1	2	1_2>2_1
1	2	1_5>5_1>5_3>3_5>3_2>2_3
1	2	1_5>5_1>5_3>3_5>3_6>6_3>6_2>2_6
1	2	1_5>5_1>5_2>2_5
1	2	1_4>4_1>4_5>5_4>5_3>3_5>3_2>2_3
1	2	1_4>4_1>4_5>5_4>5_3>3_5>3_6>6_3>6_2>2_6
1	2	1_4>4_1>4_5>5_4>5_2>2_5
1	5	1_2>2_1>2_3>3_2>3_5>5_3
1	5	1_2>2_1>2_5>5_2
1	5	1_2>2_1>2_6>6_2>6_3>3_6>3_5>5_3
1	5	1_5>5_1
1	5	1_4>4_1>4_5>5_4
1	4	1_2>2_1>2_3>3_2>3_5>5_3>5_4>4_5
1	4	1_2>2_1>2_5>5_2>5_4>4_5
1	4	1_2>2_1>2_6>6_2>6_3>3_6>3_5>5_3>5_4>4_5
1	4	1_5>5_1>5_4>4_5
1	4	1_4>4_1
1	7	1_2>2_1>2_3>3_2>3_5>5_3>5_4>4_5>4_7>7_4
1	7	1_2>2_1>2_5>5_2>5_4>4_5>4_7>7_4
1	7	1_2>2_1>2_6>6_2>6_3>3_6>3_5>5_3>5_4>4_5>4_7>7_4
1	7	1_5>5_1>5_4>4_5>4_7>7_4
1	7	1_4>4_1>4_7>7_4
1	6	1_2>2_1>2_3>3_2>3_6>6_3
1	6	1_2>2_1>2_5>5_2>5_3>3_5>3_6>6_3
1	6	1_2>2_1>2_6>6_2
1	6	1_5>5_1>5_3>3_5>3_2>2_3>2_6>6_2
1	6	1_5>5_1>5_3>3_5>3_6>6_3
1	6	1_5>5_1>5_2>2_5>2_3>3_2>3_6>6_3
1	6	1_5>5_1>5_2>2_5>2_6>6_2
1	6	1_4>4_1>4_5>5_4>5_3>3_5>3_2>2_3>2_6>6_2
1	6	1_4>4_1>4_5>5_4>5_3>3_5>3_6>6_3
1	6	1_4>4_1>4_5>5_4>5_2>2_5>2_3>3_2>3_6>6_3
1	6	1_4>4_1>4_5>5_4>5_2>2_5>2_6>6_2

Results and Evaluation

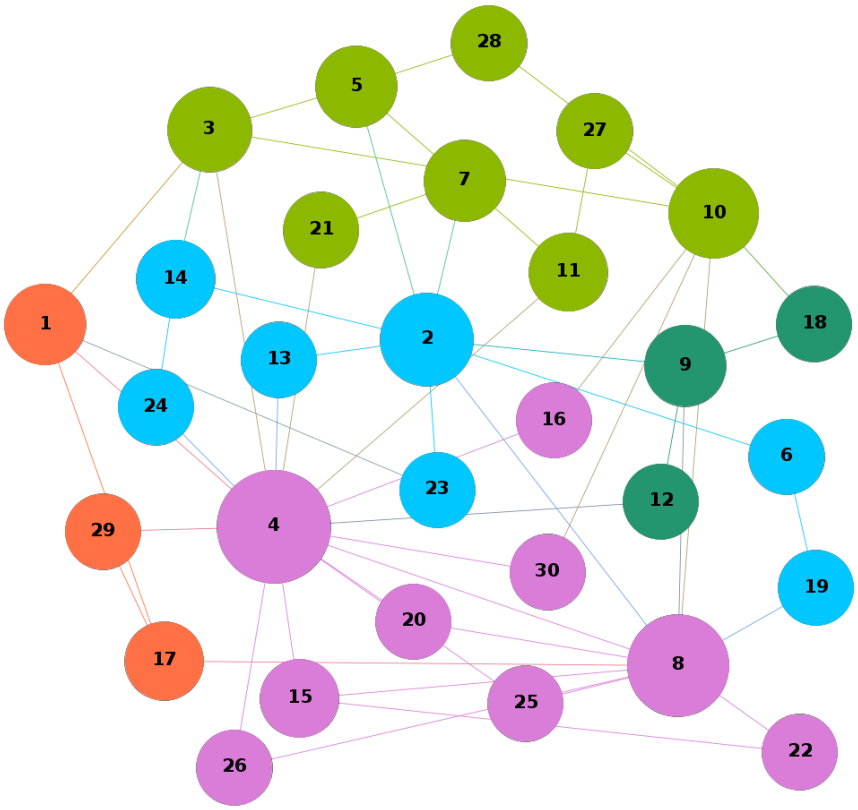
Five topologies with the same network properties

Topology 1

Topology 1: AS-level Topology of 30 ASes

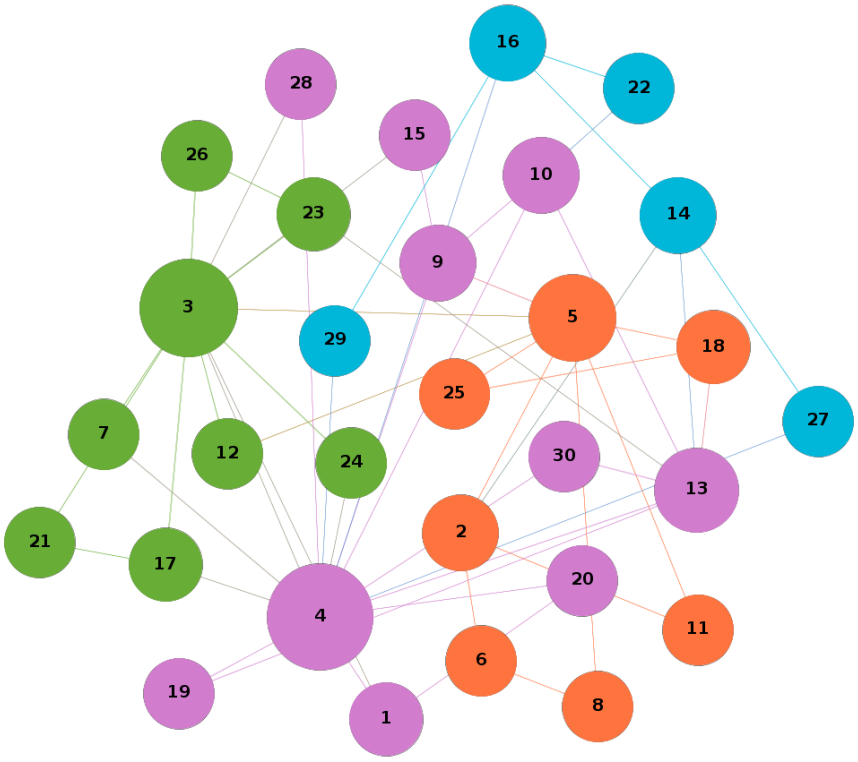
24;4;3
25;4;3
26;4;3
27;4;5
20;3;4
21;3;17
22;2;3;28;29
23;4;10
28;22;3
29;4;22
1;3;4;9
3;1;4;5;8;10;12;14;15;17;18;20;21;22;24;25;26;28;30
2;5;6;7;12;22
5;2;3;4;13;14;16;27
4;3;1;5;7;8;9;10;11;13;15;17;19;20;23;24;25;26;27;29
7;2;4;11;30
6;2;16
9;4;1;19
8;4;3
11;7;4
10;4;3;23
13;4;5
12;2;3;18
15;3;4
14;5;3
17;4;3;21
16;5;6
19;4;9
18;3;12
30;3;7

Topology 2



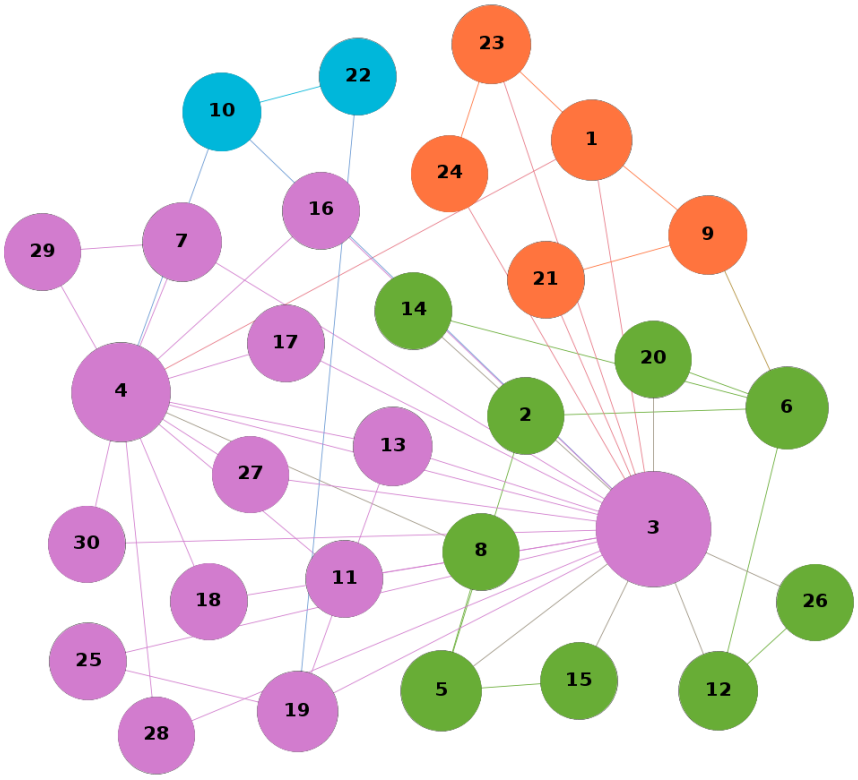
Topology 2: AS-level Topology of 30 ASes

Topology 3



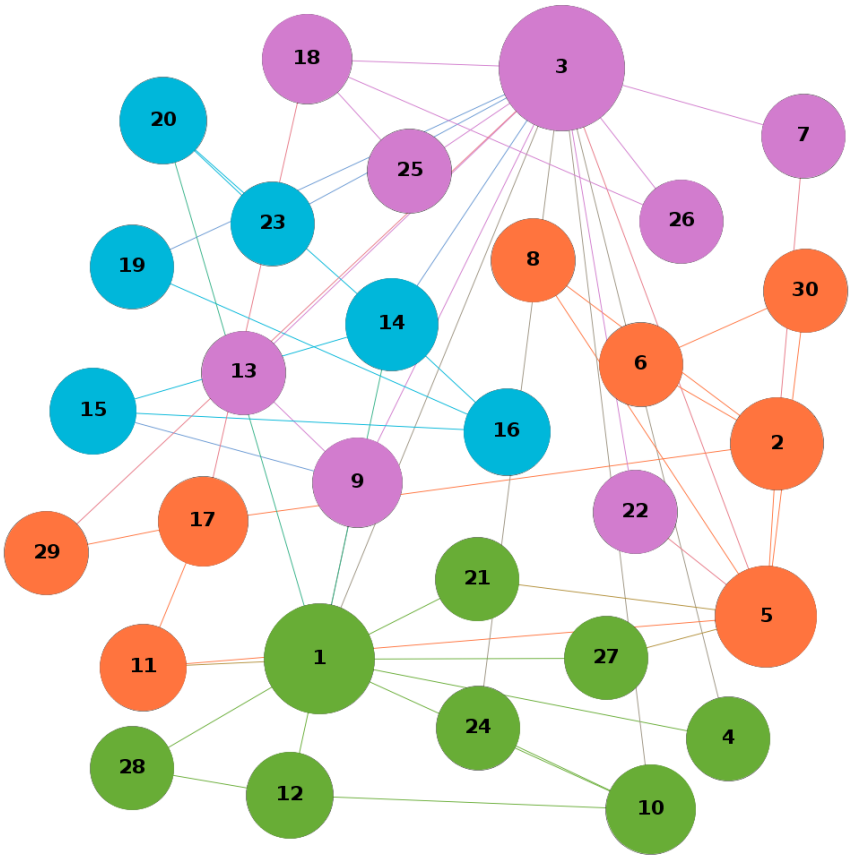
Topology 3: AS-level Topology of 30 ASes

Topology 4



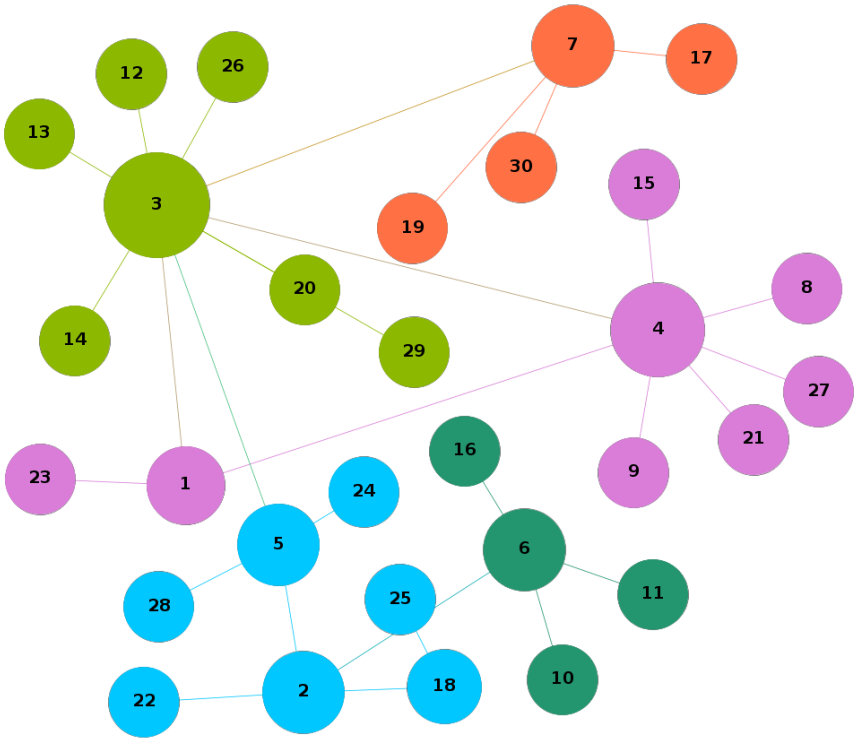
Topology 4: AS-level Topology of 30 ASes

Topology 5



Topology 5: AS-level Topology of 30 ASes

A 30-AS topology of average node degree of 2



AS-level Topology of 30 ASes (average node degree:2)