School of Electrical Engineering and Computing

Department of Computing

Energy Aware Traffic Engineering

Gongqi Lin

This thesis is presented for the degree of

Doctor of Philosophy

of

Curtin University

March 2014

Declaration

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgment has been made.

This thesis contains no material that has been accepted for the award of any other degree or diploma in any university.

Signature:………………………………………….

Date:……………………..

Abstract

Current networks are typically over-provisioned to ensure low delay, high capacity and reliability. These Quality of Service (QoS) guarantees are typically achieved using high end, high power network equipment and redundancy. Their use, however, has led to concerns regarding greenhouse gas emissions, which has recently garnered a lot of attention and resulted in a number of global initiatives that aim at reducing the carbon footprint of Internet Service Providers (ISPs). Specifically, a number of recent studies estimate that power consumption related to Information and Communication Technologies (ICT) itself varies from 2% to 10% of the worldwide power consumption. This trend is expected to increase notably in the near future, *i.e.*, twice the current value in 2020. Furthermore, the telecommunication infrastructures, especially routers, generate 37% of total ICT emissions. These initiatives have motivated ISPs and researchers to design novel network algorithms and hardware to scale the usage or active time of network components according to traffic load.

One of the main techniques used to manage resources and ensure reliable performance in IP networks is intra-domain traffic engineering (TE), which involves adapting the routing of traffic to the network conditions, with the joint goals of good user performance and efficient use of network resources. Intra-domain TE uses information about the network traffic profile (traffic matrix) that specifies the expected traffic rate between ingress-egress pair in the network to manage and possibly optimize the network performance.

To this end, our work aims to utilize TE to shut down a subset of nodes and links during off-peak traffic demands in order to minimize power expenditure while satisfying network performance requirements, *i.e.*, maximum link utilization (MLU), path length and reliability. The output of our energy-aware TE is a routing policy that includes (i) a set of switching off network resources and (ii) a set of paths with their corresponding relative rate vector that specifies the fraction of traffic assigned to each path. Our policy considers three routing path models: single path, multiple paths and two disjoint paths (2DP).

In the first part of the thesis, we focus on single path routing problem in networks with bundled links each of which contains multiple cables that can be switched off independently. We propose an efficient approach - Single Path by Shortest Path First (SSPF) to power off redundant cables as long as the remaining cables provide sufficient capacity to satisfy traffic demands. SSPF routes each traffic demand using only a single path, and our extensive simulations show that this restriction does not reduce the effectiveness of our approach while significantly reducing time complexity as compared to the existing approaches. Moreover, SSPF could significantly reduce the power usage of cables used in the network while guaranteeing a given threshold of MLU.

In the second part of the thesis, we consider multiple paths routing, focusing on networks with bundled links. We design a fast heuristic, called Multiple Paths by Shortest Path First (MSPF), which aims to maximize the number of switched-off nodes and cables subject to satisfying MLU and end-to-end delay constraints. We have extensively evaluated the performance of MSPF on both real and synthetic topologies and traffic demands. Further, we have compared its performance against two state-of-the-art techniques: GreenTE, usable only when each link has one cable, and FGH, that supports bundled links but usable only for networks without MLU and path length constraints.

In the last part of the thesis, we consider using 2DP to distribute the traffic demands. We address the problem of minimizing the power usage of networks that use 2DP. Specifically, we define Energy-Aware Two Disjoint Paths Routing (EAR-2DP) problem to maximally switch off redundant cables while ensuring the availability of at least $0 \leq Q_T \leq 1.0$ fraction of all possible (s_d, t_d) 2DPs with MLU no greater than a configured threshold. We first prove that EAR-2DP is NP-complete. Then, we design a fast heuristic, called Two Disjoint Paths by Shortest Path (2DP-SP) that considers both two link-disjoint paths (2DP-L) and two node-disjoint paths (2DP-N) to solve the problem. We have extensively evaluated the performance of 2DP-SP on real and/or synthetic topologies and traffic demands for 2DP-L and 2DP-N.

Acknowledgements

I would like to thank many people who have made this thesis possible and who have made the last four years a stimulating and very enjoyable experience.

In particular, I would like to express my sincere gratitude to my supervisor, Dr. Sieteng Soh at Curtin University, for his generous guidance and invaluable advice. I would also like to express my thanks to my co-supervisors Dr. Mihai Lazarescu at the Curtin University and Dr. Kwan-Wu Chin at the University of Wollongong for their excellent comments and valuable discussion.

I must thank W. Fisher and M. Zhang for providing us the FGH and GreenTE source code and data, and the anonymous reviewers for their valuable comments that have significantly improved our published papers. In addition, special thanks must go to the China Scholarship Council (CSC) and Curtin International Postgraduate Research Scholarships (CIPRS) for providing me with a scholarship, without which I could not have accomplished my research for the thesis.

I greatly thank my colleague, Jie Peng, for her great support and friendship during the four years in the Computing Department of Curtin University.

Finally, I would like to extend my gratitude to my family for allowing me to talk endlessly about the difficulties as a PhD student. Thank you to my parents who are always my strongest backing. Last but not least, I would like to say ‘thank you’ to my girlfriend, Lnyie Lee, who has always been encouraging and supportive with love and great passion throughout the period of my studies, and has helped me a lot with the formatting of this thesis.

Contents

List of Tables

List of Figures

Published Works

This thesis is based upon several works that have been published over the course of the author's PhD, listed as follows in chronological order:

1. G. Lin, S. Soh, K. Chin and M. Lazarescu, "Power-Aware Routing in Networks with Delay and Link Utilization Constraints," in 38th *Local Computer Network (LCN)*, Clearwater Beach, USA, 2012.

2. G. Lin, S. Soh, K. Chin and M. Lazarescu, "Performance Comparisons of Energy-Aware Routings," in *PEECS*, Perth, Australia, 2012.

3. G. Lin, S. Soh, K. Chin and M. Lazarescu, "Efficient Heuristics for Energy-Aware Routing in Networks with Bundled Links," *Computer Networks*, 2013.

4. G. Lin, S. Soh, K. Chin and M. Lazarescu, "Energy-Aware Two Link- Disjoint Paths Routing," in *IEEE International Conference on High Performance Switching and Routing (HPSR)*, Taipei, Taiwan, 2013.

5. G. Lin, S. Soh, K. Chin and M. Lazarescu, "On the Effects of Energy-Aware Traffic Engineering on Routing Reliability," in *IEEE Asia Pacific Conference on Communications (APCC)*, Bali, Indonesia, 2013.

6. G. Lin, S. Soh, K. Chin and M. Lazarescu, "Reliable Green Routing by Two Disjoint Paths," in *IEEE International Conference on Communications (ICC)*, Sydney, Australia, 2014.

7. G. Lin, S. Soh, K. Chin and M. Lazarescu, "Power-Aware Routing in Networks with Quality of Services Constraints," *Transactions on Emerging Telecommunications Technologies*, (In Print, 2014).

8. G. Lin, S. Soh, K. Chin and M. Lazarescu, "Energy-Aware Two Disjoint Paths Routing," *Journal of Networks and Computer Applications*, (In Print, 2014).

Chapter 1

Introduction

We are currently entering an era of increasing concern for environment in which Internet plays an important role both as a replacement for traveling and as a medium to convey environmental information. However, the Internet itself and its related information and communication technology (ICT) consume growing significant power, and start to have an impact on global warming.

Today's computer network infrastructures around the world consume non-negligible amount of power. For example, the power usage of the network infrastructures in Italy in 2006 exceeded 1.4 terawatt per hour (TWh), which is approximately 0.7% of the total power usage [1]. Other examples include Verizon, where in 2010, consumed 10.24 TWh, and AT&T that recorded 11.14 TWh [2] power usage. British Telecom reported that the overall power consumption for its network and estate during the 2008 financial year was 2.6 TWh, making it the biggest single energy consumer in the country [3]. In 2012, small network equipment in U.S. homes, *i.e.*, modems, routers, and gateways, as shown in Table 1.1, consumed approximately 8.3 TWh of electricity, which resulted in 5 million metric tons of carbon dioxide emissions [4].

Table 1.1 Power Consumption of U.S. Residential Small Network Equipment [4]

Product Type	Average Power (W)	Average Unit Power Consumption (KWh)	Units (millions)	National Power Use (TWh)
Modems	5.7	50	40	2
Gateways	7.9	69	42	2.9
Routers	5.7	50	53	2.6
Switches	1.9	17	1	0.1
Access Points	2.6	23	2	0.1
Optical Network Terminal	16.2	142	6	0.8
Total			144	8.3

These power consumption figures are expected to increase further given that today's networks are designed to support the maximum number of customers whilst meeting their Quality of Service (QoS) requirements with insignificant considerations for energy efficiency. These goals are usually achieved by building many redundant links and adequately over-provisioning and engineering links to ensure low delays, and to absorb any rise in traffic resulting from link failures or key events. For instance, high-end Internet Protocol (IP) routers use complex multi-rack architectures that are able to support increasing network functionalities and network traffic that increases 2.5 times every 18 months [5]. These highly engineered links, however, are usually underutilized. In fact, the average link utilization in backbone networks of large Internet Service Providers (ISPs) is estimated to be around 30%-40% [6].

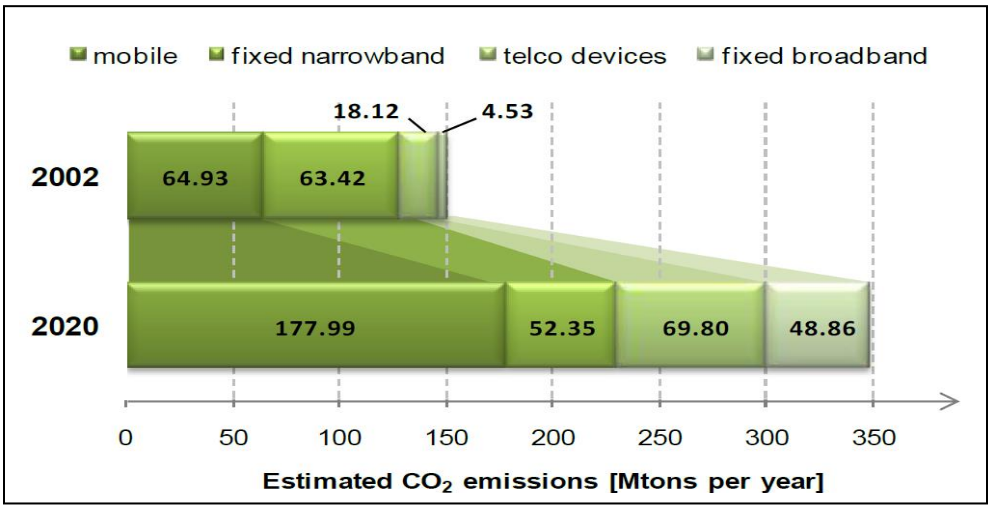


Figure 1.1 Greenhouse gas emission estimation according to GeSI [7]

Thus, much like in other areas where energy efficiency is a concern, there are two main motivations that drive the quest for "green" networking: 1) the environmental factor that aims to reduce CO_2 emission shown in Fig. 1.1; 2) the economic factor, the main incentive of each network operator, to counterbalance ever-increasing cost of power, as shown in Fig. 1.2, while keeping the network up and running at the desired service level. As shown in Fig. 1.1, the Global e-Sustainability Initiative (GeSI) reported that, by 2020, carbon footprint of networks and related infrastructures will generate nearly 350 Mtons of CO_2 emissions, and wired network devices (*e.g.*, routers, switches, *etc.*), and broadband access equipment are expected to reach 22% and 15%, respectively, of the overall network's CO_2 emissions [7]. Fig. 1.2 shows Operating Expenses (OPEX) estimation related to power costs for the European Telco's' network infrastructures in the "Business-As-Usual (BAU)" and in the Eco-scenario [8], and cumulative savings between two scenarios [8, 9]. The rapid increase in energy price and awareness of greenhouse effect will eventually trigger more restrictive government policies on the energy footprint of the ICT sector, which in turn will stimulate demands for effective energy efficient network solutions [10].

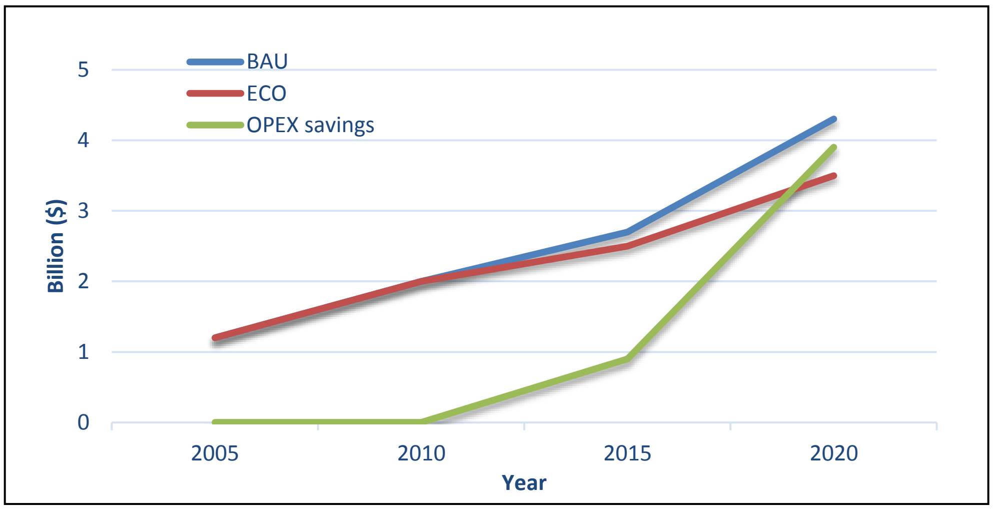


Figure 1.2 Financial estimation of power consumption [8, 9]

The described environmental and economic factors encourage telcos, ISPs and researchers to investigate possible power-aware networks that use green architectural solutions and protocols, and innovative equipment to reduce network's power consumption. Among others, major ICT companies and research bodies have invested capital on developing more energy-sustainable data centers and network infrastructures [8]. In this context, designing an energy-aware network is a challenge, because it requires radical changes in the Internet design, as well as some of the user habits. However, the initiatives are imperative since the continuous growth of the Internet unsustainably increases its total power consumption. Although green networking research is still in its infancy, a number of interesting works have already been carried out; see Chapter 2 for their details.

The current practice of over-provisioning network resources, coupled by significant traffic variations from peak to off-peak periods and development of green network devices, provide a unique opportunity for energy-aware network design. Traffic variation is strongly correlated with the time and location of users that use the networks. Specifically, fewer network accesses are expected at night (off-peak period) as compared to during the day (peak period) since more users would be using networks at work, and heavier traffic is generated in the city center by business entities at weekdays as compared to in suburban area with fewer network accesses from private places. The network usually provides over-provision resources, *e.g.*, link capacity to avoid overload during peak hours [11]. Such practice is applied in today's core networks and is currently favored by many ISPs as the preferred mechanism for QoS [12]. Specifically, one reason for over-provisioning is to minimize the chance of network congestion induced by traffic bursts, in addition to ensure its survival even after some failure of its devices and to maximize the performance experienced by the users, *i.e.*, QoS.

Intel Corporation has introduced the 0BASE-X concept [13], whereby line-cards are able to quickly switch from active mode, in which data can be transmitted rapidly, to idle mode to save power, and vice versa. The concept is effective in reducing power consumption of links with low utilization. As the power consumption of backbone routers and their line cards is essentially independent of link load [14], it is natural to set all under-utilized routers and line cards during off-peak periods into a sleep mode. However, current studies only concern on the switching off resources or adapting rate to reduce the power, and consider few on the performance of the network, such as path length, maximum link utilization (MLU) and reliability. Our work in this thesis considers the trade-off between the power consumption and the network performance requirements.

1.1 Aim and Approach

We consider the design of energy-aware traffic engineering (TE) in the core networks that support single path, multipath, and/or two disjoint path (2DP) routings. In general, we aim to reduce the power consumption of the whole network, rather than only a single device in the network, and propose three efficient TE techniques to balance the network's power usage and performance, *i.e.*, path length, MLU and reliability. More specifically, the aims of our works are as follows.

Aim 1 – To propose a green routing algorithm that routes each traffic demand via a single path. In most router implementations, packets that belong to a particular Transmission Control Protocol (TCP) session are routed via a single path. Our proposed algorithm, called Single Path by Shortest Path First (SSPF), aims to switch off redundant cables as long as the remaining cables provide sufficient capacity to distribute traffic demands via single path while guaranteeing the MLU constraint.

Aim 2 – To propose a green routing algorithm that routes each traffic demand via one or more paths, *i.e.*, multiple paths routing. Multiple paths routing provides not only path diversity to allocate traffic flow, but also enhances the route reliability (RR). Our proposed algorithm, called Multiple Paths by Shortest Path First (MSPF), aims to switch off idle resources (routers and/or cables) to reduce the power consumption with two QoS requirements, *i.e.*, path length and MLU. Using the same QoS requirements MSPF can save more power as compared to SSPF.

Aim 3 – To propose a green routing algorithm that routes each traffic demand via two disjoint paths (2DP). As compared to the single or multiple non-disjoint paths routing, using 2DP makes failure much less likely, and improves the network throughput for many applications, such as survivable design of telecommunication networks and restorable/reliable routing. Our proposed approach, called Two Disjoint Paths by Shortest Path (2DP-SP), aims to switch off idle links and/or nodes under the MLU constraint while guaranteeing using a given fraction of available 2DP in the network.

1.2 Contributions

This thesis has the following four main contributions.

1) It provides a survey of energy-aware TE. We have provided the survey of energy-aware TE, which describes the related work of energy-aware TE, and analyzes the existing works.

2) It presents an optimization problem, called Single Path Energy-Aware Routing (SP-EAR), as linear programming formulation, and proposes a novel solution called SSPF, to power off redundant cables as long as the remaining cables provide sufficient capacity to satisfy traffic demands. We build on the work in [15]. However, it is important to note that our work is different to [15] in two significant aspects. First, unlike the method in [15] that allows multiple paths, SSPF routes each traffic demand using only a single path. Our extensive simulations show that this restriction does not reduce the effectiveness of SSPF while significantly reducing time complexity as compared to the approach in [15]. SSPF only takes 0.385 seconds, as compared to 79.6 seconds using FGH [15] to find redundant cables to switch-off in the Abilene topology, which translates to a saving of 50%, versus 46.3% produced by FGH, in energy consumption. Second, the method in [15], while reducing energy, does not set an upper bound on link utilization. In contrast, we include the MLU $0 \leq U_T \leq 1.0$ as a constraint in our model. In the thesis, we have proposed three versions of SSPF: SSPF-1, SSPF-2 and SSPF-R. SSPF-1 and SSPF-2 are exactly the same except for heuristic functions that are used to determine candidate cables to be powered off. Given that both SSPF-1 and SSPF-2 may result in a *local minimum*, we have proposed SSPF-R to overcome the local minimum and produce better results than SSPF-1 and SSPF-2; Chapter 3 compares the performance of these three versions.

3) It presents an optimization problem, called Multiple Paths Energy-Aware Routing (MP-EAR) to maximally turn-off unnecessary nodes and cables in a network with bundled links, while satisfying two performance constraints: MLU and path length. More importantly, our problem generalizes the NP-hard

problems in [15] and [16]. We design an efficient and effective heuristic solution, *i.e.*, MSPF, to solve the generalized problem. Considering only switched off links, as compared to GreenTE [16], Chapter 4 shows that MSPF runs on average 99% faster while improving its power savings by 5% on tested topologies and traffic demands. In addition, MSPF requires only 0.35% of the run time of FGH, the fastest approach in [15], while yielding equivalent power savings. Further, we also evaluate MSPF using synthetic topologies in which routers and links can both be switched off.

4) It proposes a new optimization problem, called Energy-Aware Two Disjoint Paths Routing (EAR-2DP), to maximally turn-off unnecessary network resources, while satisfying two performance constraints: MLU and the required fractions of using 2DPs. We formulate EAR-2DP in linear programming, and prove the problem NP-complete. To the best of our knowledge, EAR-2DP is the first problem that combines disjoint paths routing and energy-aware TE in a wired network. This problem is important for some widely used applications, such as Voice over IP (VoIP). In the thesis, we propose two novel algorithms, *i.e.*, 2DP-SP-1 and 2DP-SP-2 to solve EAR-2DP. 2DP-SP-1 (2DP-SP-2) prioritizes switching off links (nodes) to nodes (links). Both algorithms identify network links and/or nodes that can be powered-off under two constraints: the threshold of MLU and the lower bound fraction of routes that use 2DP. Further, both algorithms can be used for applications that require two link-disjoint paths (2DP-L) or two node-disjoint paths (2DP-N) routing. Our extensive experiments in Chapter 5 confirm the efficiency of both 2DP-SP-1 and 2DP-SP-2 and their impact on network performance, *i.e.*, reliability and path length. For example, for GÉANT network, 2DP-SP-2 can obtain 58.27% power savings by switching off nodes and links.

1.3 Thesis Organization

This thesis is organized as follows.

Chapter 2 describes the background and notations that are used in the thesis. The chapter discusses the related works, and describes a set of networks and traffic demands that are used in all experiments in the thesis.

Chapter 3 discusses the SP-EAR routing problem, and provides the heuristics to solve the problem, *i.e.*, SSPF. This chapter also analyzes the time complexity of different versions of SSPF, and presents our evaluation methodology and results.

Chapter 4 proposes the MP-EAR routing problem, and designs a fast and efficient algorithm, *i.e.*, MSPF. This chapter analyzes the running time complexity of MSPF, and evaluates the performance of MSPF using both real and synthetic topologies and traffic matrices. It also provides performance comparisons against GreenTE [16] and FGH [15].

Chapter 5 presents EAR-2DP routing problem and its NP-completeness proof. It also includes the solutions and evaluations for the problem. The chapter proposes two different approaches, *i.e.*, 2DP-SP-1 and 2DP-SP-2, to solve EAR-2DP. Further, the chapter presents our simulation results to show the trade-off between power savings and route reliability when using several existing green routings.

Chapter 6 presents the conclusion of the thesis and some possible future works.

Chapter 2

Background

This chapter is divided into seven main sections. Section 2.1 describes network model and notations that are used throughout the thesis. Note that additional notations that are used only in specific chapters will be described in their corresponding chapters. Section 2.2 shows three path selection schemes in network routing. Section 2.3 gives an overview of Intra-domain TE, and introduces QoS issues in TE. Section 2.4 analyzes the green networking strategies, involving green networking architecture and green networking protocols. Section 2.5 focuses on the green routing in TE. This subsection reviews existing green routing algorithms in TE, and describes the practical implementation issues in green TE. Section 2.6 presents the network topologies and traffic matrices used in the thesis, and describes methods to calculate power saving in our works. Finally, Section 2.7 summarizes this chapter.

2.1 Network Model and Notations

We model an IP communication network as a weighted directed graph $G(V, E)$ where V is the set of n nodes, and E is the set of m links. Fig. 2.1 shows an example network with n=6, m=10, V={0, 1, 2, …, 5}, E={(0,1), (0,2), (1,2), (1,3), (2,3), (2,4), (3,4), (3,5), (4,1), (4,5}. Each node v represents a router with power consumption p_v, and a binary ON_v=1(0) denotes an active (inactive) status for a router v. Similar to the number of lanes and the speed limit on a road, a network link, *i.e.*, electrical cable or optical fiber, that connects two nodes is limited by how much data it can transfer per unit of time, commonly referred to as the *bandwidth* or *capacity* of a link. The capacity of a network link is generally represented by a data rate, such as 2.5 Gigabits per second (Gbps) in a core network. Each link (i, j) between nodes i and j represents a communication channel with a limited capacity $c_{ij}>0$ and power consumption p_{ij}. In Fig. 2.1, we assume all links have the same capacity, *i.e.*, c_{ij}=10 Gbps for each $(i, j)\in E$.

We consider line cards that have an active/idle toggling capability, and are connected by multiple physical cables. These cables form one logical bundled link [17] as standardized by the IEEE 802.1 AX [18]; the Medium Access Control layer treats each bundled link as a single link. Each link (i, j) consists of $w_{ij}\geq 1$ cables, each of which can be switched-off independently; we call w_{ij} the bundle size of link (i, j). In Fig. 2.2, the bundle size of link between Router A and Router B is $w_{AB}=3$. Specifically, a $c_{AB}=30$ Gbps link that consists of $w_{AB}=3$ cables has 10 Gbps capacity on each of its cables. The multi-port line-cards offer three main benefits [14]. First, network operators can manually configure several Ethernet links into one logical link via link aggregation to enlarge link bandwidth when network traffic increases; thus it is easy to gradually increase/decrease link bandwidth to satisfy increasing/decreasing traffic during peak/off-peak periods. Second, bundled links use optical switches to provide terabits bandwidth at much lower power dissipation than electronic switches. Finally, multiple physical cables could help reduce single point of failure, enhancing the network resilience. Note that when $w_{ij}=1$, one link refers to one cable. Let $n_{ij}\leq w_{ij}$ be an integer that represents the total number of powered-on cables in (i, j), and $E_r\subseteq E$ be a set of links in which each $(i, j)\in E_r$ has $n_{ij}>0$. Notice that link (i, j) is disconnected when $n_{ij}=0$.

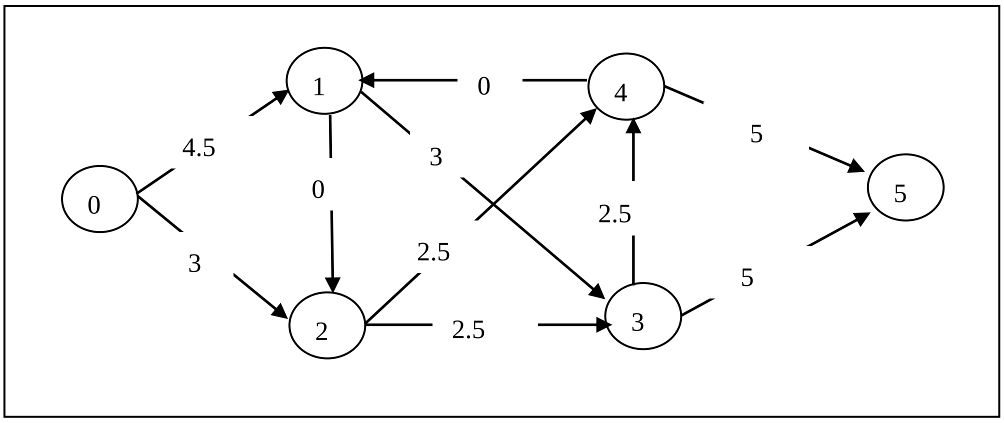


Figure 2.1 An example network $G(V,E)$

Let D be a traffic matrix that contains a set of demands, and (s_d, t_d, b_d) denote a traffic demand $d=1, 2, \ldots, |D|$ between source node $s_d\in V$ and terminal node $t_d\in V$, where b_d is the amount of traffic exchanged between these nodes. Each traffic matrix refers to packets generated by different applications, *e.g.*, surfing on the Internet or sending an email, from various source nodes to different destination nodes. We assume the network has sufficient resources, *i.e.*, link capacities, to route all demands. For example, the network in Fig. 2.1 shows the routes of all seven demands in D={(0,1,1.5), (0,3,3.0), (0,5,3.0), (2,3,1.0), (2,4,1.0), (4,5,1.0), (3,5,6.0)}. As will be described in Section 2.5.4, traffic demand changes over time, and thus routing adjustment is performed periodically.

Table 2.1 Summary of Notations

Variable	**Description**
$G(V, E)$, $n=\|V\|$, $m=\|E\|$	Graph G with a set of nodes V and links E.
$D=\{(s_d, t_d, b_d), d=1,\ldots, \|D\|\}$	A set of Traffic Demand D; s_d, t_d and b_d are the source, destination and volume of demand d respectively.
c_{ij}	The capacity of link (i, j).
f_{ij}	The total flow of link (i, j).
u_{ij}	The link utilization of link (i, j).
U_T	The threshold of MLU
r_{ij}	The spare capacity of link (i, j).
w_{ij}	Bundle size of link (i, j).
n_{ij}	Powered-on cables in a link (i, j).
p_{ij}	Power consumption of a cable in a link (i, j).
ON_v	0 if node v is sleeping, 1 otherwise.
p_v	Power consumption of node v.
sp_{dq}	The q^{th} simple (s_d, t_d) path.
$SP_d=\{sp_{dq}, q=1,\ldots \|SP_d\|\}$	A set of simple (s_d, t_d) paths.
MP_d	A set of routing (s_d, t_d) paths.
dp_{dl}	The l^{th} two disjoint (s_d, t_d) paths.
DP_d	A set of two disjoint (s_d, t_d) paths.
$L(sp_{dq})$	The path length of path sp_{dq}.
$B(sp_{dq})$	The minimum spare capacity of links on a path sp_{dq}.

For each demand $d \in D$, let SP_d be a set of candidate paths that can be used to route d. Specifically, $SP_d=\{sp_{dq}|\text{all } (s_d, t_d) \text{ paths for } d \text{ indexed by an integer } q>0\}$. Let $L(sp_{dq})$

denote the length of path sp_{dq} that is computed by hop count. We assume paths in SP_d are sorted in increasing path length, *i.e.*, sp_{d1} is the shortest path (SP) for demand d. As an example, for demand d=2 with s_2=0, t_2=3 and b_2=3.0, there are three candidate paths in set $SP_2=\{sp_{21}=(0,1,3), sp_{22}=(0,2,3), sp_{23}=(0,1,2,3)\}$ with $L(sp_{21})=L(sp_{22})=2$, $L(sp_{23})=3$. Each traffic demand d can be routed via a single path, multiple non-disjoint paths, or 2DP; see Section 2.2 for their details. Single path routing routes each demand d via one $sp_{dq}\in SP_d$; as discussed in Section 2.2.1, the selected path is preferably the shortest path to reduce end-to-end delay, *i.e.*, select sp_{dq} that has sufficient capacity to route traffic volume b_d with minimum $L(sp_{dq})$. On the other hand, multipath routing routes each demand d through one or more paths, denoted as MP_d. Specifically, $MP_d\subseteq SP_d$ is a set of (s_d, t_d) paths that are used to route demand d, *i.e.*, $MP_d=\{sp_{dq}|$one or more paths in SP_d are used to route traffic demand d, $q=1\ldots|MP_d|$ }. In multiple path routing, we compute the end-to-end path length $L(MP_d)=max\{L(sp_{dq})|sp_{dq}\in MP_d\}$. Two ($s_d$, t_d) paths are link-disjoint if they have no common links; we call such pair 2DP-L. Let DP_d be a set of all possible 2DP-L for demand d, *i.e.*, $DP_d=\{dp_{dl}|$all (s_d, t_d) 2DP-L for d indexed by an integer number $l>0\}$. Note that $dp_{dl}=\{sp_{dx}, sp_{dy}\}$, where sp_{dx}, $sp_{dy}\in SP_d$ have no common links. In Fig. 2.1, $DP_2=\{dp_{21}=\{sp_{21}, sp_{22}\}\}$ since sp_{21} and sp_{22} contain no common links. Similar to 2DP-L, we call two (s_d, t_d) paths 2DP-N if they have no common nodes. Since sp_{21} and sp_{22} contain no common nodes except the source and the destination nodes, $dp_{21}=\{sp_{21}, sp_{22}\}$ is also 2DP-N.

As defined in [19], a traffic flow, such as specific transport connection or a media stream, is a sequence of packets sent from a particular source to a particular unicast or multicast destination. Let f_{ij} be the total flow on (i, j). For example, Fig. 2.1 shows that the total flow on links (0,1) and (3,5) are f_{01}=4.5 and f_{35}=5.0. Note that we compute the total number of powered-on cables in link (i, j) as $n_{ij} = \lceil f_{ij} / (c_{ij} / w_{ij}) \rceil$.

The utilization of link (i, j), represented by u_{ij}, is defined as the ratio of used bandwidth to link capacity, *i.e.*, $u_{ij}=f_{ij}/c_{ij}$. Further, the MLU of the network is computed by selecting the maximum value of u_{ij} among all link (i, j) in E, *i.e.*, MLU=$max\{u_{ij}|\ (i, j) \in E\}$. The average MLU in backbone networks of large ISPs is estimated to be around 30-40% [36]. Let $0 \leq U_T \leq 1.0$ be the threshold of link utilization, *i.e.*, $u_{ij} \leq U_T$, and r_{ij} be the remaining/spare capacity on link (i, j), computed as $r_{ij}=(n_{ij}/w_{ij})U_T c_{ij}-f_{ij}$. Let B($sp_{dq}$) be the spare capacity of any path $sp_{dq} \in SP_d$, calculated by taking the smallest r_{ij}, for each link $(i, j) \in sp_{dq}$. Consider Fig. 2.1 with $c_{ij}=10$, $w_{ij}=2$ and $U_T=0.5$ as an example. A typical routing algorithm, *i.e.*, 2DP routing, described in Chapter 5, without power saving, routes 7 traffic demands in D as shown in the Fig. 2.1. For path sp_{21}, $f_{01}=4.5$ and $f_{13}=3$, $r_{01}=U_T{*}c_{01}-f_{01}=$ 0.5*10.0-4.5=0.5, $r_{13}=2$, $u_{01}=f_{01}/c_{01}=4.5/10.0=0.45 \leq U_T$, and $u_{13}=0.3 \leq U_T$. Further, taking the minimum spare capacity of its two links, B(sp_{21})=0.5. Table 2.1 summarizes the notations.

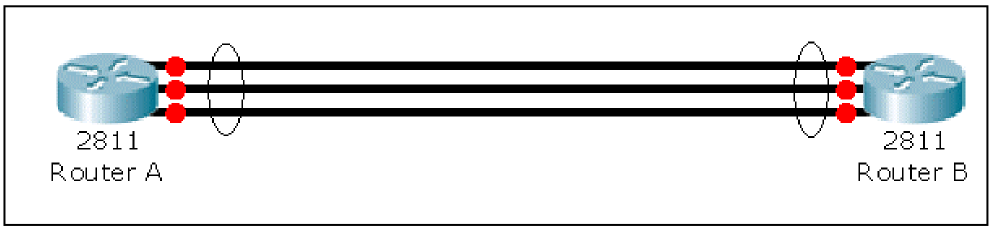


Figure 2.2 A bundled link with three cables

2.2 Network Routing

The main task of a communication network is to *flow* or *route* traffic from a source router to a destination router through their communicating links. To do that, we need to determine a route between the source router and the destination router using a routing protocol. The Open Shortest Path First (OSPF) [20] and Intermediate System to Intermediate System (IS-IS) [21] are currently two most widely used intra-domain routing protocols in the Internet. OSPF is a dynamic routing protocol that can quickly detect topological changes. Specifically, possibly due to router interface failures and/or link failures in the Autonomous System, OSPF can calculate a new loop-free path during a short convergent period [20]. IS-IS routing protocol, operating by reliably flooding link state information throughout routers within a network, is used in TCP/IP [21]. Note that, both OSPF and IS-IS use *Dijkstra*'s algorithm [22] for computing the best path through the network. However, besides best-effort paths, routing traffic in specific environment needs to consider different requirements, *i.e.*, path length, load balancing, and reliability, and thus an efficient path selection scheme should optimize the three factors. The following subsections discuss three path selection schemes, *i.e.*, single path, multiple paths, and disjoint paths.

2.2.1 Single Path Routing

There are three main advantages of using single path routing. Firstly, an end-to-end traffic in most applications in Internet, wireless networks, or overlay, in most cases is sent over a single path because splitting the traffic may cause packet reassembly problem at the receiver and thus is generally avoided. Secondly, throughput of the TCP can be affected significantly due to out of order packets [23]. For example, the Multiprotocol Label Switching (MPLS) is generally used with a working configuration that avoids splitting demands, and in most router implementations, packets that belong to a particular TCP session (*i.e.*, going to a specific destination in terms of IP address of the end computer) are routed on a specific shortest-path (even if multiple shortest paths are available) [24]. Lastly, Proposition 4.1 in [24] shows that when the number of demands is much larger than the number of edges, which occurs in most practical cases, most demands can be routed through single paths.

For the single path routing, it is important to route each demand via its shortest path to reduce end-to-end delay. Therefore, generating end-to-end shortest path, called *single shortest path problem* is fundamental to all routing problems in communication networks. Some practical implementation, such as Cisco, uses the value $1/c_{ij}$ as a link weight for each link (i, j), and the single shortest path problem is to generate a path with the minimum total weight. One can use *Dijkstra*'s algorithm to efficiently solve the single shortest path problem in $O(n^2)$. In recent years, several versions of *Dijkstra*'s algorithm have been suggested to reduce the time complexity for non-fully meshed networks to $O(m+nlogn)$, making these versions of *Dijkstra*'s algorithm more suitable to generate the shortest paths in most communication networks [25]. The shortest path routing in large network requires fully automated routing protocols for two main reasons. Firstly, frequent manual set up of routing tables in large networks are difficult and time consuming, if feasible. Secondly, fast rerouting is required since network configuration may change, *e.g.*, due to outages, during short periods. Therefore, the development of fast routing algorithms to compute single shortest path represents the top priority in the current networks.

2.2.2 Multiple Paths Routing

All paths between a given pair of source and destination nodes in multiple paths routing have at least one different link or node, and each of them must be loop-free [26]. As compared to a single path routing, multiple paths routing offers the following four benefits. Firstly, multiple paths routing can enhance end-to-end reliability by providing multiple alternative paths when network links and nodes fail. Secondly, multiple paths routing can satisfy the QoS requirements, *e.g.*, MLU no larger than 40%, by distributing traffic flow via different paths. Thirdly, multiple paths routing can reduce the network congestion by splitting flow into multiple different paths during the peak time period. Finally, multiple paths routing is good for load balancing, which aims to optimize resource usage, maximize throughput, and avoid resource overload.

In the thesis, we use multiple paths in Chapter 4 and Chapter 5. In practical implementation, we use k-shortest (s_d, t_d) paths as candidate paths for a traffic demand d. Hoffman and Pavley [27], Bellman and Kalaba [28] and Sakarovitch [29] propose algorithms to find k shortest paths, each of which may contain loop. *Yen* [30] presents an efficient algorithm to rank k-shortest loopless (s_d, t_d) paths in a network with the worst-case computational complexity of $O(kn(m+nlogn))$. The algorithm is desirable since upper bound time complexity increases linearly with the value of k.

To deploy a traffic flow in a MPLS network, we often need to represent the flow as the union of paths from the source to the sink, such that each path is coupled with the amount of flow that it carries. In general, there may be many ways to represent a flow as a union of source-sink paths based on the different requirements. Firstly, although all these representations route the same flow, they may differ substantially in their path latency. Secondly, minimizing the number of paths for routing traffic is desirable for reducing resource consumption. Thus, multiple paths routing needs to consider possible requirements, referred to as limited demand split. The requirement of limited split of the demand volumes is to assure that the volumes are realized with a predetermined number of non-zero path-flows, *e.g.*, equal split among *k* paths, use shorter paths first, or split flow arbitrary among *k* paths.

2.2.3 Disjoint Paths Routing

Multiple paths between a given pair of source and destination nodes in a network are called link disjoint if they have no common (*i.e.*, overlapping) links, and node disjoint if, besides the source and destination nodes, they have no common nodes. As an example in Figure 2.1, 0-1-3-4-5 and 0-2-3-5 are two link disjoint paths (2DP-L), while 0-1-3-5 and 0-2-4-5 are two node disjoint paths (2DP-N).

Multiple link-/node-disjoint paths routing could add more benefits to multipath routing since link-/node-disjoint paths routing can increase resiliency against link/node failures and throughput of network. Thus, the problem of finding disjoint paths in a network has been given much attention in the literature due to its theoretical as well as practical significance to many applications, such as survivable design of telecommunication networks and restorable/reliable routing. With the development of optical networks and development of MPLS or Generalized Multi-Protocol Label Switching networks, the disjoint paths problem is receiving renewed interest as fast restoration after a network failure is crucial in such kind of networks. In robust communication networks, a connection usually consists of 2DP-L or 2DP-N: one active path, and one backup path. A service flow is redirected to the backup path if the active path fails.

The problem of finding link/node disjoint paths can be viewed as a special case of the minimum cost flow problem as demonstrated in [31, 32]. In [31], Suurballe proposed an algorithm to find *k* link-disjoint paths with minimal total length using a path augmentation method. The basic idea of Suurballe's algorithm is to find 2DP-L based on the shortest path and a shortest augmenting path. In [33], the algorithm in [31] is extended to the Suurballe-Tarjan (S-T) algorithm for finding 2DP-L from one source node to *n* destination nodes using only a single *Dijkstra*-like computation. To find *n* pairs of disjoint paths, the S-T algorithm requires $O(m\log(1+m/n)n)$ time; where *n* is the number of destination nodes and *m* is the number of links.

Building on the S-T algorithm, Kar *et al.* [34] and Kodialam *et al.* [35, 36] develop new algorithms to find 2DP that can serve as active and backup paths for routing that guarantees bandwidth. In [37], Liang extended the S-T algorithm to find 2DPs between a source and a destination node with the following performance constraints: network load and routing cost. Similar to multiple paths routing, there are three splitting methods to route traffic using 2DP: equal split among 2DP, use shorter path first, and split flow arbitrary among 2DP. The detail of using disjoint paths will be discussed in Chapter 5.

2.3 Intra-domain Traffic Engineering

2.3.1 Overview of TE

Intra-domain networks are established and operated by ISPs that control the location of network routers and links. An ISP aims to move packets efficiently through its network, either for its owned general packets or packets in transit; see Fig. 2.3. TCP that guarantees reliable data connectivity requires network host to adjust its sending rate to the available bandwidth on the path from each source to destination, which in turn forces routers to compute new paths when network topology changes. Mechanisms using TCP can enhance network robustness but may sacrifice the network's efficiency. For example, real-time applications, such as VoIP call, may use a route with high propagation delay even when a low-latency path is available. Further, a particular link in a network may be congested despite the presence of under-utilized links in other parts of the network. In this context, TE [38] is used to improve network performance, *e.g.*, path delay, link utilization and reliability, and efficiently utilizes network resources. In other words, a major goal of TE is to facilitate efficient and reliable network operations while simultaneously optimizing network resource utilization and traffic performance.

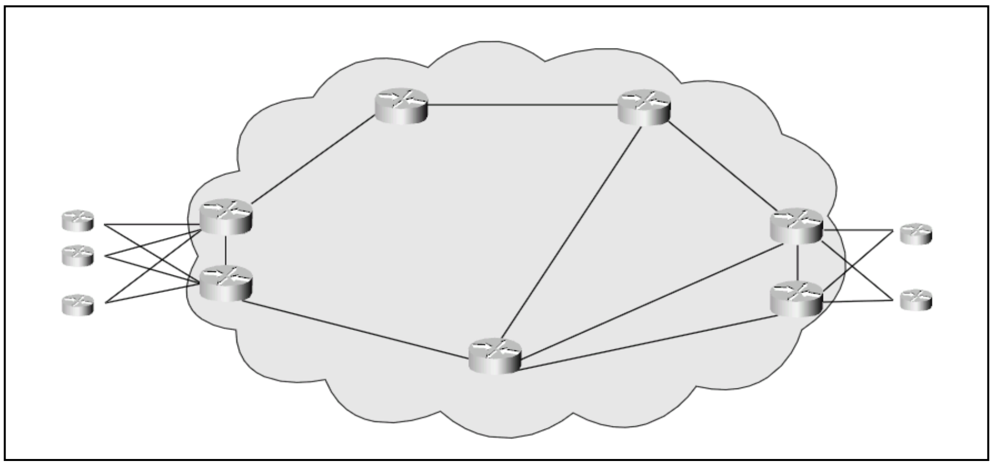


Figure 2.3 Intra-domain IP network

The authors in [39] describe three main steps to TE: measure, model, and control.

Measure:

In the measure step, information related to network topology and traffic matrix is collected. The topology and configuration information are available from router configuration data (link capacity and Interior Gateway Protocol (IGP) parameters) and Simple Network Management Protocol (status of the network elements). A traffic matrix describes a network-wide total traffic volume carried within a domain. Each element of such a matrix, *i.e.*, a traffic demand, describes a volume of traffic between every (s_d, t_d) pair of ingress and egress points over a given time interval. Understanding the variability of Internet traffic in backbone networks is essential to better plan and manage existing networks, as well as to design next generation networks. For example, a wide variety of network design problems, *i.e.*, routing protocols [15, 16, 40], load balancing schemes [41], IGP link weight setting algorithms [42, 43], reliability and failure analysis [44], and multiple paths routing [45] require a traffic matrix as input in order to carry out performance evaluation. Specifically, network operators require traffic matrices as inputs to apply the routing method under test, which can help determine the allocation of resulting loads on all network links.

Model:

In the modeling step, a model for estimating the flow allocations is built by configuring IGP parameters. In other words, TE needs to predict the traffic flow with network routing configuration. In current architecture of TE, routing model firstly computes the shortest path to route each traffic demand to ensure efficient and loopless routing, which generates a set of paths for each pair of routers. These paths can be combined with network topology information and the traffic matrix to estimate the fraction of traffic on each link for each demand.

Control:

In the control step, the operators reconfigure the network settings on routers using three sub-steps. First, the router renews its link-state database, and notifies new weight to other routers in a network when the weight changes. Second, each router updates its link-state database after receiving the new link-state advertisement from source router. Finally, each router computes the new shortest paths, and updates its routing table. Note that a service provider needs a human operator to oversee the reconfiguration process due to operation complexity of a large IP network. In particular, network operators can reconfigure static link weights according to the different applications.

2.3.2 Quality of Service in TE

Network QoS is based on the service level agreements (SLAs) between network customers and providers, which set the terms and conditions on behalf of both providers and customers for providing and requesting/accessing services, respectively. QoS guarantees provide new business opportunities while presenting new challenges for current networks. The guarantees are important for network with limited resources, *e.g.*, link capacity. For example, real-time streaming multimedia applications, such as VoIP, online games and IP-TV, require fixed bit rate and are delay sensitive. Therefore, one needs to consider QoS requirements while optimizing network resources using TE. This thesis considers three types of QoS, *i.e.*, network reliability, link utilization, and path length.

Network reliability, defined as the ability of an IP network to recover quickly and smoothly from one or a series of failures or disruptions, is becoming increasingly important QoS requirement. Some major carriers, such as AT&T, BT and NTT, consider network reliability as one of the most important metrics while deploying communication services [46]. Some researchers [44, 47, 48] include network reliability in TE. Zhang *et al*. [47] integrate failure recovery with load balancing by rerouting some fraction of the traffic after a single edge failure. Reference [48] proposes an architecture that uses local rerouting to handle up to F link failures subject to link capacity constraints, while [44] uses end-to-end routing that does not require link state flooding and dynamic router reconfigurations. References [36, 49] generate link/node-disjoint paths subject to given QoS requirements, *e.g.*, bandwidth and/or path delay. Their works produce primary and restoration paths that satisfy the QoS constraints; the link/node-disjoint paths routing are resilient to link or node failures. The authors of [50] include reliability as a metric in their 2DP computation. They present a problem, called multiple constrained link-disjoint path pair, to find link/node-disjoint paths in multiple dimensions, and prove the problem is NP-complete when multiple link metrics are used.

To satisfy link utilization requirement, the authors of [41] propose a method to dynamically split traffic over multiple, not necessarily disjoint, paths. A load balancer [41] is used to split each (s_d, t_d) traffic demand among its available paths with the objective of minimizing the MLU. The balancer selects the paths, starting from the shortest, to route the split traffic flows. In contrast, considering the network fault-tolerance, the authors of [51] propose to split and route flow evenly among multiple link-disjoint paths to provide higher bandwidth protection against component failures. Specifically, their work involves selecting working or active and protection or alternative paths from the set of disjoint paths that are pre-computed for each (s_d, t_d) pair. However, reserving protection paths requires more resources, which incur low link utilization while does not address the case when working paths cannot not route split flow. Similar to the method in [51], the thesis considers equal-cost multipath (ECMP) [52] over 2DP to offer higher throughput and fault-tolerance. ECMP is a commonly deployed technique where routers keep track of all shortest paths, and evenly split traffic amongst them [53].

To preserve the interactivity of video applications, *e.g.*, IP-TV and VoIP, data delivery with a low latency is required. However, the current Internet is not well equipped to support the delivery of delay-sensitive traffic for two reasons [54]. First, the research on setting end-to-end path delay constraint explicitly has not received enough attention, and the existing solutions are not practical. Second, transmissions are vulnerable to transient periods of congestion that cause temporary delays or losses, which degrade the quality of live audio or video streams. In [55], the authors construct a TE system based on minimum-delay routing. However, this work doesn't give exact constraint on the path delay as in [56].

2.4 Green Networking

The aim of green networking is to minimize network power consumption while maintaining the level of QoS, *e.g.*, delay, bandwidth and reliability, required by applications. As discussed in Section 1.1, reducing power consumption has direct effect in reducing greenhouse gas emissions [57], a noble fight to preserve our environment. Reducing network energy, by using only required network resources at any time, is feasible since network operators commonly over-provision bandwidth and network traffic varies significantly between peak and off-peak periods. This section focuses on green networking architecture and protocols.

2.4.1 Green Networking Architecture

Green networking architecture design addresses the problem for deploying routers over a set of Point of Presence (PoP) to minimize network power usage [14]. Recently, two different approaches are proposed to build a green architecture: an incremental approach based on existing infrastructures [58], [59] [60] and a clean-slate approach that redesign a new architecture completely [61], [14], [62]. In [58], a centralized global management approach schedules resources *on* or *off* during low or peak periods, respectively, to maximize network power savings. The authors in [59] propose an activity/sleep model and use a simple opportunistic “wake on arrival” strategy as part of their routing protocol. A Markov model of a state-dependent service rate queue is used in [60] to evaluate the mean packet delay, the time spent in a power-saving low link data rate, and the oscillation of link data rates. For clean-slate approaches, the authors in [61] provide efficient architectures that combine optical transport and packet processing to balance the power consumption and network performance. The approach in [14] associates power consumption cost with multi-commodity flow under the performance and robustness constraints. A similar approach in [62] evaluates the tradeoff between power consumption and network performance, *e.g.*, fault-tolerance.

2.4.2 Green Networking Protocols

There are three main directions to design new green protocols: virtualization, proportional computing and resource consolidation. Virtualization uses a set of mechanisms such that more than one service can be performed on the same network resources such as network routers and line cards. On the other hand, proportional computing calculates power usage of each device in proportion to the amount of work performed on it. Since our work in the thesis uses the resource consolidation technique, we refer interested readers to [63] and [64] for a detailed survey of the virtualization technique, and [65], [66] and [67] for discussion on proportional computing.

Resource consolidation approach uses a subset of network resources to reallocate network traffics during off-peak period while guaranteeing the required level of performance. The approach sets unused resources into sleeping or standby states to reduce energy since devices in sleeping states use very low energy. The protocol configuration in [68] and [69] uses the consolidation approach in its energy management strategy, and integrates resource consolidation into its network management platforms. The platforms control all devices centrally, and change related configuration of devices, *e.g.*, to change route settings, to switch between active/sleep modes. Some networks [70, 71] use a proxy, called Network Connectivity Proxy, to switch-on each sleeping device. The main objective of network connectivity proxy is to temporarily takeover the low-level network presence tasks, *e.g.*, Address Resolution Protocol, for sleeping devices, and to power on the device when it is necessary.

2.5 Green Traffic Engineering

2.5.1 Motivation and Problem

Current networks are typically over-provisioned, significantly exceeding their average utilization, to ensure low delays, redundancy and reliability during busy or rush hour load [20]. As described in [16], the average link utilization of Abilene, a large US education backbone, is only 2%, with MLU fluctuates between 10% and 20%. This situation commonly emerges in large commercial networks. While the average link utilization is so low, network power consumption, without energy-aware routing, stays constant, wasting so much energy during off-peak periods.

Green TE problem aims to switch off idle nodes and links in a network such that all traffic demand requirements are met and some network performance constraints, *e.g.*, MLU and path delay, are satisfied. Green TE problem is a special case of the multi-commodity capacitated network design problem (MCND), a known NP-complete problem [72]. MCND’s reduction, from the Satisfiability problem, uses as many commodities as there are clauses; the authors in [73] present a reduction of the Satisfiability problem to the two-commodity integral flow in directed graphs (D2CIF). Several heuristics [74-76] and branch-and-cut methods [56, 77, 78] have been proposed to solve this problem. Reference [79] studies a 0-1 reformulation of MCND, and shows that extended linking inequalities, derived from variable disaggregation techniques, are equivalent to residual capacity inequalities. The authors of [79] provide a heuristic method that produces a lower bound for MCND with a value that is equivalent to one computed by a linear programming (LP). Note that their method can be used to generate the initial routes for energy aware routing as an alternative to the LP approach used in [15].

Green networking research was emanated from the seminal work of Gupta *et al*. [12]. The authors examined the power consumption of networking devices and discussed their impact on network protocols if they are put to sleep. They showed that packets routed through networks with coordinated sleeping, called the network-wide approach, require protocol changes, whilst those with uncoordinated sleeping, called the link layer approach, only require local information. In a subsequent work, Gupta *et al*. [80] explore this idea in a wired local area network setting. However, as argued in [16], this approach is not applicable to backbone networks that have short packet interval times. Nevertheless, their works [12, 80] have inspired recent research on conserving energy in networks. There has been a handful of works on green TE, some of which uses distributed optimization [81-87] while others utilize centralized optimization [15, 16, 40, 88-96].

2.5.2 Distributed Green TE

Vasic *et al*. [81] present EATe, a technique that takes power consumption into account while achieving the same traffic rates between source and destination nodes as energy oblivious approaches. However, they assume fixed end-to-end paths, which make their approach non-flexible and hard to operate. The authors in [82] propose an energy-aware source routing protocol for a cognitive packet network. However, their method is usable only for smart packet networks [82]. The authors of [83] propose a distributed method that selectively switches off links in an IP-based network to save power. A distributed strategy, proposed in [84], generates an energy-aware network topology and weight metrics for each time period. In their design [84], the network control and management system is responsible for populating a historical demand matrix for each time period. In [85], the authors proposed REsPoNse by introducing a distributed low-complex on-line component on routers, which monitors the load state of links and immediately reacts to congestion situations. The solution that pre-computes on-demand paths, in addition to pre-computed always-on paths, is activated as soon as link congestion is detected. In [86], a distributed method based on an ant colony optimization is proposed, which exploits an ant colony-based self-adaptive power saving routing scheme. The scheme, referred as A-ESR, automatically aggregates the incoming flows on specific heavily loaded links and switches off the other lightly loaded links. However, A-ESR may not converge due to possible routing instabilities and their consequent packet losses. Coiro *et al.* [87] propose a distributed energy-aware TE, called DAISIES, for packet-switched connection-oriented network, *e.g.*, an IP/MPLS network. The distributed method in [87] adopts a routing-based approach that searches the best path for each traffic demand; the actual link switch-off/on is a consequence of routing decisions. However, the authors [87] do not consider any QoS requirements.

2.5.3 Centralized Green TE

In [90], the authors propose a centralized algorithm that exploits the algebraic connectivity of a network to find the set of links that can be put into standby state, and thus generates an energy-aware network topology. Such topology-oriented solution is a planned operation, performed statically by a network administrator for each time period. Thus, when the algorithm incorrectly removes a sub-optimal link, it will never backtrack to correct its mistake for the period. Other centralized optimization solutions on energy-aware TE use shortest paths routing [15, 16, 40, 88, 89, 91-93]. However, these studies do not consider links with bundled cables except [15]. Amaldi *et al.* [92] use SP routing with changing OSPF link weights. There are two heuristics proposed in [92]. The iterative greedy approach is used in the first searching method. The second one is based on a mixed integer linear programming (MILP) formulation and is composed of two phases. The first phase minimizes the power consumption by solving MILP formulation that assumes fully splitting routing. The second phase finds a set of OSPF links weights such that the traffic demand can be routed on the active (energy-aware) topology. Takeshita *et al.* [93] focus on the optimal solution of the green TE problem. This paper provides an approach to create all combinations of the network link topologies in order to find a topology with the minimum link set that satisfies bandwidth constraints under SP routing. However, though lower as compared with conventional solutions, its computation time exponentially increases with the number of network nodes, demonstrating that optimal solution of green TE is infeasible in practice. As an example, the authors [93] estimate the computation time to be about 10^6 and 10^{16} seconds for networks with 60 and 100 links respectively. In [88], the authors formulate the problem of power consumption as multi-commodity minimum cost flow problems, and provide methods to switch off routers according to policies such as random, least link, least-flow and most-power. Their algorithms in [88], running on a centralized controller, select the minimum set of devices that must be switched on to meet current traffic demands. Further, they consider a scenario that reconfigures the network periodically, *e.g.*, every 30 minutes, to match the daily traffic variation in current backbone networks. Note that fewer network reconfigurations reduce potential power saving but mitigate latencies incurred when changing power state. The authors of [88] consider different node types, *i.e.*, access and backbone nodes. The approach in [89] uses shortest paths routing protocol to find network elements (routers and weighted links) that can be switched off to minimize the total power consumption while guaranteeing a given MLU constraint. For their power saving problem [91], the authors use an integrated IP layer strategy that is compatible with OSPF. Their approach aims to generate a subset of IP links to be powered off, when network traffic decreases, subject to satisfying a given link load constraint.

The authors of [16] propose an intra-domain TE mechanism, called GreenTE, which maximizes the number of links that can be put into sleep to minimize power consumption under two performance constraints: MLU and packet delay. They modeled the problem as a mixed integer program and proposed a heuristic algorithm. For each demand *d*, their algorithm requires *k*-shortest (s_d, t_d) paths as the input, and uses the AMPL/CPLEX solver [97] to generate routes for all traffic demands that use the minimum number of switch-on links/nodes, and hence minimizes power usage, subject to the given constraints. Note that the computational time of the algorithm increases significantly with increasing values of *k*. However, unlike our work and that in [15], GreenTE considers each link with only a single cable. Further, similar to [15], their model [16] may route a demand through multiple paths.

Recent studies [15, 40, 94, 96] focus on centralized energy-aware routing with bundled links. Fisher *et al.* [15] consider each core router connected by multiple physical cables that form one logical bundled link, and propose to turn-off redundant cables. They propose three heuristics, which differ in their time complexity. However, these heuristics require using an LP solver (AMPL/CPLEX) a number of times, *i.e.*, $O(m^2)$ to $O(m^3)$ times, where *m* is the number of links, and therefore is expensive. Further, their solution does not address two important issues. Specifically, their solution might re-route traffic demands through longer paths that incur delays beyond a tolerable limit, and it might push each link’s utilization above the acceptable upper limit. Note that in practice, link utilization is set below 50% to accommodate traffic shifts and increase network fault tolerance.

The proposed minimization problem in [94] considers the power consumption of router processors and traffic load through routers. Specifically, the approach in [94] aims to switch off bundled links. However, since the authors of [94] consider backbone network scenario, their solution does not power off network routers. Further, they exclude MLU and path length constraints in the minimization problem.

The same authors also propose an approach in [95] to switch off both nodes and links with MLU constraint. However, their solution excludes bundled links and path length constraint. The authors in [96] propose dynamic local heuristic threshold-based algorithms, called DLHT, to minimize the power consumption in networks with bundled links while reducing the overflow risk and burst traffic. Using a fixed utilization threshold for MLU and a fixed sub-link adding strategy, DLHT obtains a good tradeoff between power saving and overflow. However, this work ignores nodes' power consumption, and excludes the path length constraint.

2.5.4 General Green Routing Problems and Existing Solutions

An informal description of the general energy-aware routing problem considered in the thesis is the following. Given: 1) a physical network topology that includes nodes and links, in which links have a known capacity; 2) the traffic demand exchanged by all source/destination node pairs at a given time; 3) the power consumption of each node and link, find the set of nodes and links that can be switched off so that the total power saving is maximized, subject to flow conservation and performance requirement, such as MLU, path delay and reliability constraints.

There are some existing solutions [15, 16, 88, 89, 96] that are provided to solve the related green routing problems. This subsection discusses the similarities and differences among five existing green routing solutions related to my work, namely FGH [15], DLHT [96], GreenTE [16], OPT-V [88], and MILP-BA [89]. Specifically, the thesis considers four main properties: 1) bundled links, 2) QoS constraints, 3) types of routing path, and 4) path selection. The details of each property are as follows.

1) Bundle Links

An advantage of using bundled links is that they allow network operators to easily upgrade the capacity of their network, and thus support network resiliency in case of cable failures and congestion. However, during off-peak periods, where the full network capacity is not required, there is a clear incentive, in terms of energy cost reduction, to power off cables. Moreover, each cable is assumed to have the same bandwidth and reliability, meaning a larger bundle size increases link bandwidth as well as reliability. FGH and DLHT consider each link to have $w_{ij} \geqslant 1$ cable(s), and thus achieves the said benefits as compared to GreenTE, OPT-V, and MILP-BA, which use w_{ij}=1. In other words, when GreenTE, OPT-V, and MILP-BA switch off a link, the entire line-card would be put to sleep, and has less flexibility in rerouting traffic since it cannot switch off each cable in a link independently. The algorithms developed in the thesis consider green routing problems with bundled links.

2) QoS Constraints

GreenTE aim to route all demands while satisfying two QoS constraints, *i.e.*, MLU and path hop counts. In contrast, FGH does not require any of the two constraints, and thus it excludes the link utilization constraint and sets U_T=100%. Both OPT-V and MILP-BA consider only the link utilization constraint without the path delay constraint. As mentioned in Section 2.3.2, QoS requirements set the terms and conditions on behalf of both providers and customers for providing and requesting/accessing services, respectively. The algorithms proposed in the thesis considers different QoS requirements in green routing problems.

3) Types of Routing Path

Multipath routing provides additional resiliency by providing fast (or simultaneous) access to backup paths. GreenTE, FGH, and OPT-V allow each demand to be routed through multiple paths while MILP-BA restricts each demand through shortest paths. However, none of them uses single path routing and link/node disjoint paths. As mentioned in Section 2.2.1, the benefits of single path routing give motivation to develop the solution to solve green routing problems in the thesis. The use of disjoint paths in Section 5 of the thesis in routing further improves the network resiliency as compared to the multipath routing in the existing solutions.

4) Path Selections

FGH and GreenTE use a LP solver to generate paths for all demands after switching off cables/links, and therefore the algorithms cannot explicitly select more reliable paths for a set of demands. Consequently, both FGH and GreenTE cannot be used for applications that require paths with reliability constraints or other requirements such as single path routing or disjoint path routing. Further, the use of LP may lead the heuristic methods in FGH and GreenTE to local minima, preventing the algorithms from generating higher energy savings. The algorithms proposed in the thesis explicitly generate the best routes to maximize energy savings while satisfying their respective required constraints. In particular, each algorithm first generates k (s_d, t_d) shortest paths for each demand d using *Yen*'s algorithm [30], which in turn is used to effectively selects the shortest (two link-disjoint) path (paths) that mimimize swicthed-on cables/nodes while satisfying link utilization and path delay constraints.

As discussed, in essence, the algorithms proposed in the thesis improve the efficiency, effectiveness as well as the versatility of the existing green routing solutions. In particular, we avoid using LP in our algorithms such that they can produce higher energy savings as well as providing the ability to select only single path and disjoint paths in the routing; the SSPF and 2DP-SP algorithms in Chapter 3 and Chapter 5 use single paths and disjoint paths respectively. Further, all algorithms in the thesis consider links with multiple cables.

2.5.5 Implementation Issues

This thesis considers green routing algorithms that run on a logically centralized controller in the Network Operation Center (NOC). The NOC, among others, collects input information, *e.g.*, network topology and traffic matrix, from routers, and uses the green routing algorithms to get new routing configurations, and disseminates the results to routers. Each router will switch on/off some line-cards or ports according to the green routing solutions and set up MPLS tunnels for data forwarding if needed. In the following, we briefly discuss three possible implementation issues.

First, the controller rapidly collects information of network topology from each router which floods OSPF’s Link State Advertisements (LSAs) whenever link state of each router changes. For collection of traffic matrix, it is hard to directly measure real-time traffic in large networks. Thus, some studies, such as GreenTE [16], collect link load information from routers and compute the traffic matrix locally. When information that includes network topology and traffic matrix is collected, the controller announces the information via LSAs.

Second, once its green routing algorithm generates outputs, the controller selects the subset of resources that must be powered on to meet the current traffic demand, and distributes this information to routers via TE Metric attribute [98]. To minimize packet loss during routing transition, the time to switch on/off routers or links should be minimized. In practice, such on/off ability requires hardware support, *i.e.*, line-cards that can go into sleep or active state in milliseconds [59]. Each router should wait for all alternative paths to be completely configured before turning off any link. Further, the router should turn on the link immediately when it wakes up a line-card; it should not transfer the data immediately until both ends are ready. Consequently, running green routing too often will reduce the network’s responsiveness to traffic variation. We consider a deployment scenario that configures network only every 30 minutes since in current backbone networks there is insignificant traffic variation in less than 30 minutes duration [90].

Finally, data forwarding is an important task in green network routing. If green routing algorithms give a path that happens to be the shortest path, the traffic is simply transferred as native IP packets by OSPF; otherwise, Label Switching Path (LSP) is set up to let the non-shortest path to carry traffic through MPLS tunnels.

2.6 Simulation Environments

This section describes network topologies, traffic matrices, and power saving calculations used in simulations performed in Chapter 3 to Chapter 5.

2.6.1 Network Topologies

We consider two types of nodes in network: transit node and access node. Access nodes are source or destination nodes while transit nodes are neither sources nor destinations of traffic. Note that each access node also acts as a transit node to route multi-hop traffic demands. Further, a transit (access) node can (cannot) be switched-off. Simulations in Chapter 3 to 5 consider switching off only network links and/or both network links and nodes. For the former case, the simulations use networks containing only access nodes, shown in Table 2.2 and Table 2.3, while for the latter they utilize networks containing both access and transit nodes, shown in Table 2.4 and 2.5. For link capacity, like in a typical ISP network, we consider links between transit nodes have higher capacity (10Gbps) than those from access to transit nodes (2.5Gbps). In the thesis, we use four real topologies, *i.e.*, Abilene [99], GÉANT [100], Sprint and AT&T [101], and three topologies from [15], *i.e.*, F_Abilene, Wax50 and Hier50. Further, we use GT-ITM [102] to generate seven topologies; hereafter we call the ten non-real topologies as *synthetic* topologies.

Table 2.2 shows four real topologies, *i.e.*, Abilene [99], GÉANT [100], Sprint [101], and AT&T [101], which contain only access nodes. The table also shows each network usage, *i.e.*, research or commercial, and its location, *i.e.*, US or Europe. The Abilene network has 12 nodes and 30 links, while GÉANT network has 23 nodes and 74 links. Rocketfuel [101] provides PoP-level topologies of commercial ISPs, such as Sprint and AT&T. Sprint network has 52 nodes and 168 links, while AT&T network has 115 nodes and 296 links. Table 2.3 summarizes seven synthetic, access nodes only, topologies with nodes ranging from 10 to 100 and links from 28 to 434. The F_Abilene in Table 2.3 that we obtain from [15] and the Abilene in Table 2.2 that we obtain from [99] are different in their link connections and capacities; in the thesis we consider the former a synthetic topology. We obtain a two level hierarchical graph (Hier50), and the Waxman graph (Wax50) from [15]; they represent large topologies in our simulations. In the Waxman graph, the probability that two nodes are connected by a link decays exponentially by the distance between them. Further, we used GT-ITM [102] to generate three random graphs Geo10, Geo30, and Geo50 that represent small, medium and large topologies respectively; Geo50 contains 50 nodes and 434 links. We also used GT-ITM [102] to generate a large hierarchical graph (Hier100) that has 100 nodes and 286 links. Topologies in Table 2.2 are used for all simulations in Chapter 3 to 5, and those in Table 2.3 only for Chapter 3.

Table 2.2 Real Network Topologies with only Access Nodes

Network	**Usage**	**Location**	**Nodes**	**Links**
Abilene	Research	US	12	30
GÉANT	Research	Europe	23	74
Sprint	Commercial	US	52	168
AT&T	Commercial	US	115	296

Table 2.3 Synthetic Network Topologies with only Access Nodes

Name	**Type**	**Nodes**	**Edges**	**Demands**
F_Abilene	Backbone	39	28	253
Hier50	Hierarchical	50	148	2450
Wax50	Waxman	50	169	2450
Hier100	Hierarchical	100	286	9900
Geo10	Random	10	28	90
Geo30	Random	30	136	870
Geo50	Random	50	434	2450

To evaluate the performance of green routings that switch-off both network node and link, *i.e.*, algorithms in Chapter 4 and Chapter 5, we generate two types of synthetic topologies that contain both access and transit nodes. The first type of synthetic topologies shown in Table 2.4 is generated from the real topologies in Table 2.2. Each synthetic topology in Table 2.4 is generated by randomly selecting some nodes from its respective real topology in Table 2.2 as transit nodes, *e.g.*, R_Abilene corresponds to Abilene, R_GÉANT to GÉANT and R_Sprint to Sprint. We use the topologies in Table 2.4 for algorithms in Chapter 5. Table 2.5 lists the second type of synthetic topologies that we generated using GT-ITM [102]; we use the topologies in Chapter 4. Each topology simulates a typical ISP network that uses bidirectional links; as a consequence, switching-off link (*i*, *j*) will switch-off its reverse link (*j*, *i*).

As shown in Table 2.5, TS8_56 has 56 nodes that include 8 transit nodes while TS23_161 and TS40_280 have 23 transit nodes and 40 transit nodes respectively.

Table 2.4 Real Network Topologies with Two Types of Nodes

Network	**Type**	**Transit Nodes**	**Access Nodes**	**Links**
R_Abilene	Synthetic	2	10	30
R_ GÉANT	Synthetic	13	10	74
R_Sprint	Synthetic	26	26	168

Table 2.5 Synthetic Network Topologies with Two Types of Nodes

Network	**Type**	**Transit Nodes**	**Access Nodes**	**Links**
TS8_56	Hierarchical	8	48	132
TS23_161	Hierarchical	23	138	536
TS40_280	Hierarchical	40	240	1988

2.6.2 Traffic Matrices

A traffic matrix can be generated directly from real measurement of the network traffic flow [103]. However, direct measurements need an additional infrastructure support, which require extra budget for expensive instrument to collect required data. Further, network carriers view their topologies and traffic matrices as proprietary [104], and thus very few real topologies and traffic data are available to the research community at large. Therefore, estimating an Internet traffic matrix has received considerable attention. In [101], the authors present measurement techniques to get high quality ISP maps, *e.g.*, AT&T, Sprint, Telstra, while using as few measurements as possible. The authors in [104] and [105] make use of synthetically generated traffic matrices to evaluate the resulting performance of the scheme being designed. In the thesis, we use both real and synthetic traffic matrices. For example, we use real traffics when they are publicly available, *i.e.*, traffic matrices for Abilene and GÉANT, for their respective real topologies in Table 2.2. Specifically, for Abilene, we use the 288 traffic matrices measured on September 5, 2004 for every five minutes for duration of 24 hours – all of which are provided by the authors of [99]. For GÉANT, its traffic matrices were collected on May 5, 2005 for every 15 minutes; we obtained the 24*4=96 traffic matrices from the authors of [100].

For the remaining topologies in Table 2.3 to Table 2.5, we either use synthetic traffic matrices that we obtain from the authors in [15] or use GT-ITM [102] to generate synthetic traffic matrices; as shown in Table 2.3, the traffic demands for each topology range from 90 to 9900. Specifically, we used the traffics provided by the authors in [15] for the synthetic topologies F_Abilene, Hier50 and Wax50 in Table 2.3. Then, we generate each synthetic traffic matrix following gravity model [106], entropy model [43], or uniform traffic pattern. Specifically, for Sprint and AT&T in Table 2.2, we randomly generated a traffic matrix using the gravity model, and scaled the traffic to obtain 10 different traffic matrices. The generated traffic matrices is such that when traffic is routed using the SP, the MLU of the topology is 10%, 20%, 30%, …, 90%, 100%. In this thesis, we refer the traffic matrix that results in a MLU of X% using SP routing as SP(X%). For example, we refer the traffic matrix that results in a MLU of 30% using SP routing as MLU 30% under SP (SP(30%)), MLU of 50% as MLU 50% under SP (SP(50%)), *etc*. For each random graph and Hier100 in Table 2.3, we consider traffic demands between each (s_d, t_d) pair in the network; *i.e.*, Hier100 with 100 nodes has 100*(100-1)=9900 traffic demands. Each traffic flow is generated using the classical entropy model for urban traffic, as described in [43]. The model computes each traffic flow b_d=10*rn_1*rn_2, where rn_1 and rn_2 are two random numbers between 0 and 1; thus, $0 \leq b_d \leq 10$. Finally, for each synthetic network in Table 2.5, we consider a uniform traffic pattern in which each source access node s_d transmits a randomly generated traffic flow $1 \leq b_d \leq 100$ *Mbps* to each terminal access node t_d. Therefore, there are $(n-n_c-1)*(n-n_c-1)$ traffic demands, where n_c is the number of core nodes, *e.g.*, TS8_56 has (56-8-1)*(56-8-1) = 2209.

2.6.3 Power Saving Calculation

The thesis uses two models of power saving (PS) calculations following the models in [15], [16], and [88]. Following FGH [15], power savings of bundled link networks, with bundle size $w_{ij} \geq 1$, is computed as:

$$PS=(1-\frac{\sum_{(i,j)\in E} n_{ij}}{\sum_{(i,j)\in E} w_{ij}})\times 100\% \tag{2.1}$$

Recall that n_{ij} is the number of powered-on cables in a link (i, j). Note that Equation (2.1) considers only switching off cables while routers are always switched-on. Since our work in Chapter 3 considers such model, we use Equation (2.1) in the chapter.

When both nodes and cables can be switched-off, power saving should also include the power consumption of active nodes; therefore, for this case, PS is calculated as:

$$PS=(1-\frac{\sum_{(i,j)\in E} n_{ij}p_{ij}+\sum_{v\in V} ON_v p_v}{\sum_{(i,j)\in E} w_{ij}p_{ij}+\sum_{v\in V} p_v})\times 100\% \tag{2.2}$$

Recall that p_{ij} is the power consumption of each cable in a link (i, j), and p_v is the power consumption of node v that has active/inactive status ON_v. The power consumption of line-cards used in our simulations is specified in [117]. We assume that OC-192/STM-64 (10Gbps) line card is used for all links and thus the maximum delay of single-hop is around 200 ns [118]. Both works in [16] and [88] aim to maximally switch off links and nodes. Since a link only involves one cable, the works use Equation (2.2) with w_{ij}=1 and n_{ij} is set to either 1 (the link is powered on) or 0 (the link is powered-off); Chapter 5 in this thesis uses Equation (2.2) with such setting, while Chapter 4 sets integer variables $n_{ij}\geq 0$ and $w_{ij}\geq 1$.

2.7 Summary

This chapter covers related works for this thesis. Section 2.1 covers network model and notations, and provides an example to describe the related notations. Section 2.2 discusses the network routing, *i.e.*, single path routing, multiple paths routing and disjoint paths routing. Section 2.3 overviews intra-domain TE, including QoS in TE. Section 2.4 shows green networking strategies, which include architecture and protocol design. Section 2.5 gives the observation of green TE, and discusses the state-of-the-art centralized and distributed energy-aware routing algorithms. Section

2.6 summarizes the simulation topologies and traffic matrix used in the thesis, and presents the calculation of power consumption in the thesis. In the following three chapters, we propose three network routing algorithms by path selection schemes with QoS constrains, such as single path routing, multiple path routing, and 2DP routing.

Chapter 3

Single Path Green Routing

In this chapter, we propose an efficient approach - SSPF to power off redundant cables as long as the remaining cables provide sufficient capacity to route each demand via a single path. Specifically, we formulate an optimization problem to generate (i) a minimum set of powered on or active cables, and (ii) a set of single paths, constructed using only the set in (i), each of which is used to route a traffic demand in D, subject to a required MLU $0 \leq U_T \leq 1.0$. We propose three versions of SSPF: SSPF-1, SSPF-2 and SSPF-R. SSPF-1 and SSPF-2 are exactly the same except for heuristic functions that are used to determine candidate cables to be powered off. Since SSPF-1 and SSPF-2 may enter a *local minimum*, we have proposed SSPF-R to reduce the possibility of going into local minima to improve the performance of SSPF-1 and SSPF-2. Our work in this chapter has been published in [40].

The approach in this chapter builds on the work in [15]. However, our work is different to [15] in two significant aspects. First, unlike the method in [15] that allows multiple paths, our approach routes each traffic demand via only a single path. Our extensive simulations in Section 3.4 show that this restriction does not reduce the effectiveness of our approach. Second, the method in [15], while reducing energy, does not set an upper bound on link utilization. In contrast, as described in Section 3.1, we include the MLU $0 \leq U_T \leq 1.0$ as a constraint in our model. Note that an ISP usually limits $U_T \leq 0.5$ [16], and therefore our approach is more practical than that in [15]. Simulations in Section 3.4 show that our approach requires significantly less time complexity as compared to the approach in [15]. For the Abilene topology, our heuristic approach takes 0.385 seconds, as compared to 79.6 seconds using FGH [15], to find redundant cables to switch-off, which translates to a saving of 50%, versus 46.3% produced by FGH, in power consumption.

The layout of this chapter is as follows. Section 3.1 formulates the problem at hand. Section 3.2 describes three efficient heuristics to solve the proposed problem, and Section 3.3 analyzes the time complexity of SSPF-1, SSPF-2 and SSPF-R. Section 3.4 presents our evaluation methodology and results. Finally, Section 3.5 concludes this chapter.

3.1 Problem Statement

Given *G*(*V*, *E*) and a traffic demand set *D*, where each demand is to be routed along a single path, the Single Path Energy-Aware Routing (SP-EAR) problem is to generate (i) the minimum number of powered on or active cables, and (ii) the set of paths that satisfies traffic demands *D* using only these powered on cables, subject to a required threshold of link utilization, *i.e.*, $0 \le U_T \le 1.0$. Let x_{ij}^d be a binary variable that is set to 1 (0) when a fraction of traffic flow b_d for demand *d* is routed (not routed) through link (*i*, *j*), and $f_{ij} = \sum_{d \in D} b_d x_{ij}^d$. The optimization problem is formalized in the following Integer Linear Programming (ILP) formulation:

Minimize

$$\sum_{(i,j) \in E} n_{ij} \tag{3.1}$$

Subject to:

$$\sum_{(i,j) \in E} x_{ij}^d - \sum_{(j,i) \in E} x_{ji}^d = \begin{cases} 1, & i = s_d \\ -1, & i = t_d \\ 0, & Otherwise \end{cases}, \forall i \in V, d \in D \tag{3.2}$$

$$f_{ij} = \sum_{d \in D} b_d x_{ij}^d \le (n_{ij} / w_{ij}) U_T c_{ij}, \forall (i,j) \in E \tag{3.3}$$

$$0 \le n_{ij} \le w_{ij}, \forall (i,j) \in E \tag{3.4}$$

The objective in the ILP is to minimize the number of powered-on cables, as per Equation (3.1). Constraint (3.2) ensures flow conservation, and requires each demand to be routed along a single path. Equation (3.3) constraints the total flow on each link to be less than the capacity provided by active cables for a given link and a given link utilization threshold U_T, and Equation (3.4) bounds the number of active cables to be less than the bundle size of each link.

As mentioned in [15], the presented formulation is equivalent to the simple two-commodity integral flow in directed graphs (simple D2CIF) problem [73], which is NP complete. Therefore, in the following sections, we propose a number of heuristics to solve the ILP.

3.2 Single Path by Shortest Path First

This section describes our heuristic approach: SSPF. Subsection 3.2.1 describes two versions of SSPF: SSPF-1 and SSPF-2; subsection 3.2.2 describes its third version, SSPF-R. This section also provides an example to illustrate SSPF.

3.2.1 SSPF

Our greedy heuristic approach, SSPF in Fig. 3.1, produces a set of paths P_D that can be used to route all demands in D through all powered on cables in link set E_r, *i.e.*, $P_D=\{MP_d \mid d\in D\}$. SSPF uses E_r and n_{ij} of each $(i, j)\in E_r$ to compute its power saving. In addition, SSPF creates a First-In-First-Out (FIFO) history log, Q, that stores the sequence of removed cables. Specifically, Q is a sequence of a pair $((i, j), nc)$ that denotes the number of switched off cables, nc, in (i, j). As described in Section 3.2.2, our SSPF-R algorithm needs the information in the set Q to avoid local minima. SSPF initializes $Q=\varnothing$, $E_r=E$, and $fix((i, j))$=false for each $(i, j)\in E$. Note that $fix((i, j))$=false means that it is still possible to turnoff one or more cables in (i, j) while satisfying all demands in D. Then, it executes the following three main steps. In **Step 1**, SSPF uses each shortest path sp_{d1} to route the traffic flow of each demand d; sp_{d1} can be computed using *Dijkstra*'s algorithm [22] and we assume unitary link delay, *i.e.*, each path length is measured in hop count. We assume the network has sufficient link capacity to route all demands in D through their shortest paths.

```
Algorithm SSPF
Begin
    $Q \leftarrow \varnothing$, $E_r \leftarrow E$, $fix((i, j)) \leftarrow$ false for each $(i, j) \in E_r$; //initialize the linkstatus
1)  For each demand $d \in D$ do
        Generate shortest path $sp_{d1}$, and route its $b_d$ through $sp_{d1}$;
        Store $sp_{d1}$ into $P_D$;
    End-For
2)  For each $(i, j) \in E_r$ do
        $n_{ij} \leftarrow \lceil f_{ij} / (c_{ij} / w_{ij}) \rceil$;
        If $n_{ij} < w_{ij}$ then // there is unused cable
                If $n_{ij}=0$ then //no cable in the link is used
                    $E_r \leftarrow E_r - \{(i, j)\}$; // turn-off all cables in $(i, j)$
                $Q \leftarrow Q + \{((i, j), w_{ij} - n_{ij})\}$;   // turn off $nc=(w_{ij}-n_{ij})$ cables in$(i, j)$
    End-For
3)  While $|E_r|>0$ do
        Use argmax() or H-Select-e() to select a link $(y, z) \in E_r$ that has $fix((y, z))$=false;
        If GH-Flow($E_r$, $(y, z)$)=true then
            $Q \leftarrow Q + \{(y, z),1\}$   //Turn off one cable in $(y, z)$
              $n_{yz} \leftarrow n_{yz} - 1$
            If $n_{yz}=0$ then
                $E_r \leftarrow E_r - \{(y, z)\}$;
            Set $fix((i, j)) \leftarrow$ false $\forall (i, j) \in E_r$; // restart Step 3
        Else
            $fix((i, j)) \leftarrow$ true;
            If $fix((i, j))$=true, for $\forall (i, j) \in E_r$ then // check the status of each link in $E_r$
                break; //End while loop
    End-While
    Delete all switched of cables in $Q$ from their corresponding links in $G$.
End
```

Figure 3.1 Algorithm SSPF

As an example, consider the network in Fig. 3.2 with a set *D* containing eight traffic demands 1 to 8, (0, 2, 4.2), (0, 5, 1.05), (0, 6, 0.95), (0, 7, 2.25), (0, 10, 8.5), (4, 5, 3.35), (4, 6, 4.35), (10, 5, 1.55) respectively, with w_{ij}=2 and c_{ij}=10. Step 1 will generate eight (s_d, t_d) paths for all demands in *D, i.e.*, P_D={MP_1=((0, 2)), MP_2=((0, 2), (2, 5)), MP_3=((0, 3), (3, 6)), MP_4=((0, 1), (1, 4), (4, 7)), MP_5=((0, 8), (8, 9), (9, 10)), MP_6=((4, 5)), MP_7=((4, 6)), MP_8=((10, 5))}. Figure 3.2 shows the total flow f_{ij} for each link (*i*, *j*); see the number without bracket for each link.

In Step 2, SSPF first calculates the total number of cables for each link needed to route all demands in Step 1. Fig. 3.2 shows the total number of needed cables, n_{ij}, for each link (i, j); see each integer in bracket. As an example, for link (4, 6), f_{46}=4.35, and n_{46} is calculated as $\lceil 4.35/(10/2) \rceil$=1; thus one cable in the link is unused. In essence, Step 2 aims to switch off the maximal number of unused cables from each link, whilst ensuring the remaining cables are capable of meeting all traffic demands. If n_{ij}=0, *i.e.*, link (i, j) is never used, the link is removed from E_r. As an example in Figure 3.2, the step removes link (9, 6) from E_r. This step also stores the number of switched off cables nc>0 for each link (i, j), *i.e.*, each pair ((i, j), nc), into Q; for the example in Fig. 3.2, Q=(((9, 6), 2), ((0, 3), 1), ((3, 6), 1), ((2, 5), 1), ((0, 1), 1), ((1, 4), 1), ((4, 7), 1), ((4, 5), 1), ((4, 6), 1)).

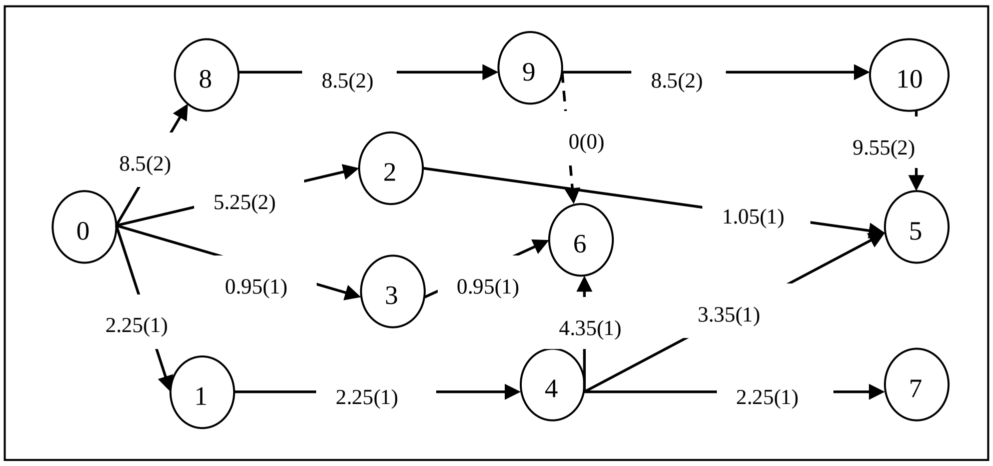


Figure 3.2 An example network for SSPF

In **Step 3**, SSPF iteratively selects a candidate link (i, j) from E_r and aims to switch off one of its cables. For this step, SSPF considers two different functions to determine the candidate link. SSPF version 1, called SSPF-1, uses the *argmax* function from [15], while its version 2, called SSPF-2, uses our heuristic function, ***H-Select-e*()**; both versions are exactly the same except for the two functions. The *argmax* function selects a non-fixed link (i, j), *i.e.*, *fix*((i, j))=false, that has the largest spare capacity r_{ij} while ***H-Select-e*()** selects a non-fixed (i, j) with the smallest average flow per demand, *i.e.*, (i, j) with the smallest f_{ij}/θ, where $\theta \leq |D|$ is the total number of traffic demands that use link (i, j). The latter function assumes that demand with less flow is easier to re-route onto an alternative path. For Fig. 3.2, *argmax* will select link (0, 2) because it has the largest r_{02}=(2/2)*1.0*10-5.25=4.75, and *H-select-e*() function will select link (0, 3) because it has the smallest f_{ij}/θ =0.95/1.

```
GH-Flow(E_r,(i, j))
Begin
    Status ← true; // The status of function
1)  For each demand $d \in D$ do
        If $sp_{d1} \in MP_d$ contains link $(i, j)$ then
            $r_{ij} \leftarrow r_{ij}+b_d$ , $\forall (i, j) \in sp_{d1}$;    // returns the resource
            If $(n_{ij}-1)=0$ then // the path is disconnected when one cable in $(i, j)$ is off
                Replace $sp_{d1}$ with a new $(s_d, t_d)$ shortest path for $d$ on $G(V, E_r)$;
            Store $sp_{d1}$ into TP; //a new or original path $sp_{d1}$ after deleting a cable
    End-For
2)  For each $TP_d \in TP$ do
    //we need to consider the threshold of link utilization constraint $U_T$
        If $TP_d$ has enough capacity to route $b_d$ for $d$ then
            $r_{ij} \leftarrow r_{ij}-b_d$, $\forall (i, j) \in TP_d$;//update the remaining capacity for each link in $TP_d$
            Replace $sp_{d1} \in MP_d$ with $TP_d$;
        Else
            Generate the $k$-shortest $(s_d, t_d)$ paths $SP_d$ for demand $d$;
            For each $sp_{dq}$ in $SP_d$ do// there are at most k different path $sp_{dq}$
                If $sp_{dq}$ has enough capacity to route $b_d$ then
                    $r_{ij} \leftarrow r_{ij}-b_d$, $\forall (i, j) \in sp_{dq}$;
                    Replace $sp_{d1} \in MP_d$ with $sp_{dq}$;
                    Go to 2);
            End-For
            Status ← false; // rerouting is not possible
            Go to 3);
    End-For
3)  Return Status;
End
```

Figure 3.3 Function GH-Flow()

Note that both functions may select a link that might lead to a *local minimum*. Therefore, we propose SSPF-R in Section 3.2.2 to heuristically restore all cables in a link and select a candidate cable in another link to avoid local minima.

For each selected link (*y*, *z*), the algorithm uses our Greedy Heuristic function,

***GH-Flow*()** in Fig. 3.3, to check if deleting a cable is feasible, *i.e.*, the remaining n_{yz}-1 cables in (y, z) and all cables in the other links in E_r can still meet the flow of all demands. The function re-routes all paths that use (y, z) to all possible paths. One cable in (y, z) will be switched off if re-routing is feasible, and a record, ((y, z), 1), is created and stored in Q, and each $fix((i, j))$ is reset to false for $(i, j)\in E_r$. Any unused link (i, j), *i.e.*, n_{ij}=0, is removed from E_r. Further, this step ensures that the flow of each (s_d, t_d) demand d is routed only through a single path sp_{dq} in G. Step 3 is repeated until it is not possible to turn off any remaining cable, *i.e.*, $fix((i, j))$=true for all (i, j), and thus SSPF terminates after turning off all cables in Q from G. The details of ***GH-Flow*()** is described as follows.

Step 1 of ***GH-Flow*()** finds each sp_{d1}, for $d\in D$, that contains the candidate link (i, j), and adds the capacity of each link in sp_{d1} with the previously allocated b_d for demand d. As an example, when (2, 5) is selected, sp_{21}=((0, 2), (2, 5)) is affected and thus the value of r_{02} and r_{25} is increased by b_2=1.05. If the path is disconnected when the cable is turned off, *i.e.*, n_{ij}-1=0, the step generates a new sp_{21}. Following the previous example, n_{25}-1=0, *i.e.*, the link cannot carry any traffic when its only cable is switched off. As the result, sp_{21}=((0, 2), (2, 5)) is disconnected, and thus the step generates a new shortest path sp_{21}=((0, 1), (1, 4), (4, 5)) for the affected demand 2. Step 1 stores either the original or new sp_{21} in a temporary path set TP; let us call each path in TP for demand d as TP_d. Since removing the cable in (2, 5) only affects demand 2, TP={TP_2}.

In Step 2 of ***GH-Flow*()**, the function routes the flow of each affected demands in D through its corresponding paths in TP. Notice that the route of the flow from each unaffected demand remains unchanged. The function uses path $\text{TP}_d\in\text{TP}$ if it can be used to route demand d, and subtract the capacity of each link in the path by its flow, b_d. Following the above example, traffic demand d=2 can be rerouted through a new path TP_2=((0, 1), (1, 4), (4, 5)) and thus the step subtracts b_d=1.05 from r_{01}, r_{14}, and r_{45}. However, if any links in the shortest path cannot support the demand, the function generates $k\geq1$ shortest paths for demand d; this can be carried out using

Yen's algorithm [30]. ***GH-Flow*()** aims to route the flow using the shortest possible path among the *k* paths. When there is more than one path with the same length, the function selects one randomly. If a flow in demand *d* can be routed using any of the *k*-shortest paths, we subtract the capacity of each link in the path by b_d. However, if none of the path has sufficient capacity to route the flow, the function knows that the deleted cable needs be turned on to meet all demands in *D*. As an example, assume *argmax* selects (0, 3), which disconnects sp_{31}=((0, 3), (3, 6)), and thus demand *d*=3 selects new shortest path sp_{31}=((0, 1), (1, 4), (4, 6)). Notice that demand *d*=3, with b_3=0.95, cannot be rerouted through new sp_{31} since r_{46}=5-4.35=0.65<0.95, and thus the step cannot switch off the cable in (0, 3). Function ***GH-Flow*()** returns false when at least one affected demand cannot be rerouted. Notice that the function does not roll-back the routes of any demands that have been successfully rerouted through their corresponding paths in TP to their original routes. As an alternative, the function may continue routing the flow of each remaining demand. Note that the running time of this alternative is longer, and therefore is not suggested.

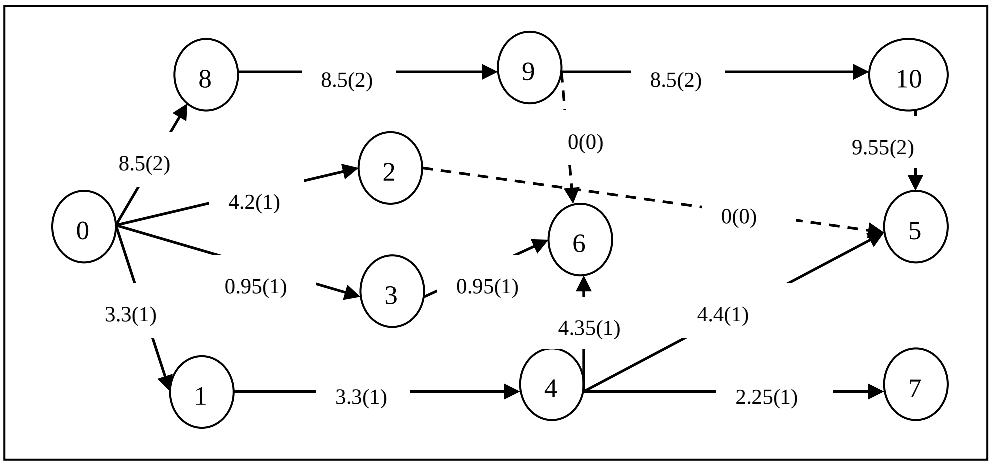


Figure 3.4 SSPF-1 solution for the network in Figure 3.2

As shown in Fig. 3.4, SSPF-1 generates a sequence of switched off cables *Q*=(((9, 6), 2), ((0, 3), 1), ((3, 6), 1), ((2, 5), 1), ((0, 1), 1), ((1, 4), 1), ((4, 7), 1), ((4, 5), 1), ((4, 6), 1), ((2, 5), 1), ((0, 2), 1)), and a path set P_D={MP_1=((0, 2)), MP_2=((0, 1), (1, 4), (4, 5)), MP_3=((0, 3), (3, 6)), MP_4=((0, 1), (1, 4), (4, 7)), MP_5=((0, 8), (8, 9), (9, 10)), MP_6=((4, 5)), MP_7=((4, 6)), MP_8=((10, 5))} to route the eight demands, where switched-off links are denoted by dashed lines. For the example, SSPF-1 is able to switch off 12 cables of 28 total cables in the network, and thus saves 42.9% of power usage.

3.2.2 SSPF-R

We propose a heuristic algorithm called SSPF-R to improve the optimality of SSPF. Similar to FGH [15], either SSPF-1 or SSPF-2 may select a candidate link that leads to local minima. The authors of [15] proposed EGH and BGH to heuristically solve the problem. However, they showed that EGH and BGH do not improve FGH significantly whilst incurring a significantly higher time complexity, particularly for solving large networks, *i.e.*, Wax50 and Hier50 in Table 2.3. Our efficient SSPF-R, shown in Fig. 3.5, greedily avoids local minima. The algorithm repeatedly assumes that the deleted cables, sequenced in Q, leads to a local minimum, and aims to correct the mistake by sequentially restoring each deletion in Q at a time, from the least recent deletion, while assuming that the remaining deleted cables were correct decisions that will lead to a global minimum. For each of constant number $\Re\leq|Q|$ iterations, **Step 1** sets E_r and Q^* as a copy of E and Q respectively. Note that in our simulation, outlined in Section 3.4, we set $\Re$ to $|Q|/2$ since running SSPF-R is faster while producing the same results as compared to using $\Re=|Q|$. **Step 2** aims to correct each possible non-optimal cable deletion by sequentially restoring one cable in pair $((a, b), nc)\in Q$ at a time while assuming that the remaining deleted cables in Q^* were correct decisions, leading to a global minimum. If the link had no cable, *i.e.*, disconnected, the step connects it back, *i.e.*, includes it in E_r. The step also initializes the status of all links, *i.e.*, setting each $fix((i, j))$=false. As an example, recall that SSPF-1 in Section 3.2.1 generates Q=(((9, 6), 2), ((0, 3), 1), ((3, 6), 1), ((2, 5), 1), ((0, 1), 1), ((1, 4), 1), ((4, 7), 1), ((4, 5), 1), ((4, 6), 1), ((2, 5), 1), ((0, 2), 1)). Step 2 sequentially attempts to restore one cable starting from pairs ((9, 6), 2) to ((2, 5), 1).

```
Algorithm SSPF-R
Begin
    $Q^* \leftarrow \varnothing$, $Best\_Q \leftarrow Q$;
    For $x \leftarrow 1$ to $\Re$ do
    1) $Q^* \leftarrow Q$, $E_r \leftarrow E$;
    2) Remove $x^{th}$ pair $((a, b), nc)$ from $Q$, i.e., $Q^* \leftarrow Q^*-\{(a, b), nc\}$;
       $n_{ab} \leftarrow n_{ab}+1$; // add one cable to $(a, b)$;
       If $n_{ab} = 1$then      //$(a, b)$was disconnected
           $E_r \leftarrow E_r+\{(a, b)\}$;    //add the link
       Set $fix((i, j)) \leftarrow$ false $\forall(i, j)\in E_r$;
    3) While $|E_r| > 0$ do
           Call $argmax()$ or $H\text{-}Select\text{-}e(G)$ to select a link $(y, z) \neq (a, b)$ from $E_r$;
           If $GH\text{-}Flow(E_r, (y, z))$=true then
               $Q^* \leftarrow Q^*+\{(y, z), 1\}$;
               $n_{yz} \leftarrow n_{yz}-1$; // Remove one cable every time
               If $n_{yz}=0$ then
                   $E_r \leftarrow E_r-\{(y, z)\}$; //delete the link
               Set $fix((i, j)) \leftarrow$ false $\forall(i, j)\in E_r$;
           Else
               $fix((y, z)) \leftarrow$ true;
               If $\forall(i, j)\in E_r$ has $fix((i, j))$=true then
                   break; //End while loop
       End-While
    4) If $Q^*$contains more deleted cables than $Best\_Q$ then
           $Best\_Q \leftarrow Q^*$
    End-For
End
```

Figure 3.5 Algorithm SSPF-R

For each attempt to restore a cable in (a, b), **Step 3** uses either *argmax* or *H-Select-e*() to select one cable from the candidate link (y, z)≠(a, b). Note that Step 3 in SSPF-R is similar to Step 3 in SSPF. If ***GH-Flow*()** function in Step 3 is able to re-route traffic flow from (y, z), the cable is deleted, and Step 3 is repeated using another candidate link (y, z)≠(a, b) until all remaining links in E_r are checked, *i.e.*, *fix*((i, j))=true for each link (i, j). Note that the status of link (y, z) is set to true if ***GH-Flow*()** fails to route affected traffic demands with one less cable in (y, z). Restoring a cable in (a, b) generates a better power saving if ***GH-Flow*()** could turn-off more than one cable while restoring one cable in (a, b). For this case, **Step 4** updates *Best_Q* with the better result, which will be returned when SSPF-R terminates.

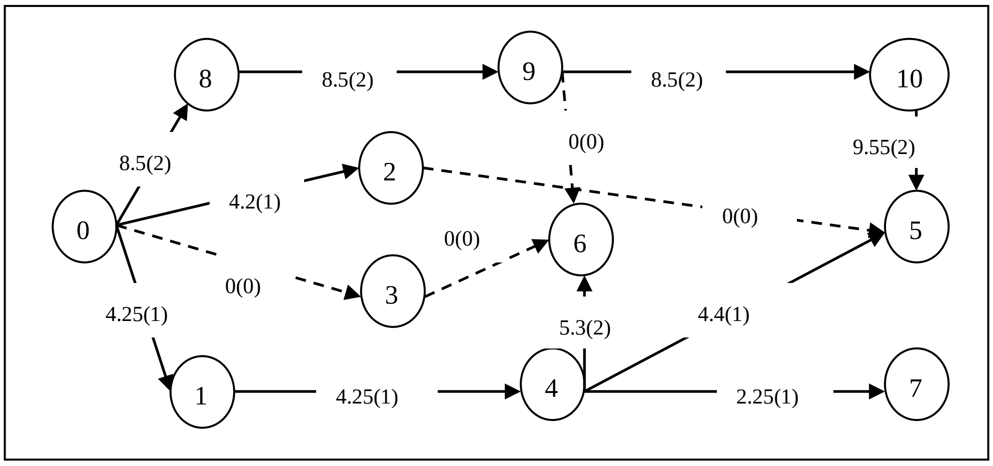


Figure 3.6 SSPF-R solution based on Figure 3.4

To illustrate SSPF-R, consider Fig. 3.4 that was generated by SSPF-1. Notice that SSPF-R finds that cable deletions in the sequence from ((9, 6), 2) to ((4, 5), 1) in *Q* are optimal and thus, the *Best_Q* equals *Q*. For ((4, 6), 1), Step 2 sets n_{46}=1+1=2, and *argmax* in Step 3 selects (0, 3), which affect sp_{31}=((0, 3), (3, 6)) that was generated in SSPF-1. Thus, ***GH-Flow*()** generates a new path sp_{31}=((0, 1), (1, 4), (4, 6)) that has sufficient capacity to route demand 3. In another iteration, *argmax* generates link (3, 6). However, since no route is affected by deleting one cable in (3, 6), ***GH-Flow*()** also returns true. Thus, SSPF-R is able to switch-off two cables, *i.e.*, one in (0, 3) and another in (3, 6) when one cable in (4, 6) is restored, with a gain of one that further reduces the power saving result when using SSPF-1; see Fig. 3.4 versus Fig. 3.6.

3.3 Time Complexity

3.3.1 SSPF

In Step 1, SSPF uses *Dijkstra*'s algorithm to generate all-pair shortest paths in $O(n^3)$, and route the flows in $O(|D|*m=mn^2)$. Note that n and m are the total number of nodes and links in G respectively, and $|D|\leq n(n-1)$ is the total number of traffic demands; thus Step 1 has a time complexity no more than $O(mn^2)$ since in general $m\geq n$. Step 2 requires searching all links in G and therefore has time complexity of $O(m)$. Step 3 for SSPF-1 uses the *argmax* function [15] which takes $O(m)$ to find each candidate link. On the other hand, Step 3 for SSPF-2 uses ***H-Select-e*()** that has $O(|D|)=O(n^2)$ time complexity. The ***GH-Flow*()** function has the worst case time complexity of $O(n^3(m+nlogn))$; the detailed time complexity analysis of the function is provided below. Since Step 3 is repeated m times, it has complexity of $O(m*(m + n^3(m+nlogn)))=O(mn^3(m+nlogn))$ for SSPF-1, and $O(m*(n^2+n^3(m+nlogn)))= O(mn^3(m+nlogn))$ for SSPF-2; both versions require the same time complexity. Thus, SSPF has worst case time complexity of $O(n^3+m+mn^3(m+nlogn))=O(mn^3(m+ nlogn))$.

The time complexity of function ***GH-Flow*()** is calculated as follows. The worst case of Step 1 requires running *Dijkstra*'s algorithm for all-pair (s_d, t_d) shortest paths, and thus takes $O(n^3)$ time. However, our simulation in Section 3.4 shows that on average, each cable deletion affects only 2% of demands. In the worst case, Step 2 is repeated $O(|D|=n^2)$ time. However, as in Step 1, the loop in Step 2 is repeated only for 2% of demands, far less than the worst case. If path TP_d has sufficient capacity to route the traffic demand then it only needs $O(m)$ time to update flow of all links on the path, since it has at most m links. Otherwise, we use *Yen*'s algorithm to generate k-shortest paths for demand d, which has time complexity $O(kn(m+nlogn))$. The worst case scenario is when the feasible path is the last of the ordered k-shortest paths; this case requires $O(km)$. Therefore this sub-step requires $O(kn(m+nlogn)+km)=O(kn(m+ nlogn))$, and the worst case time complexity of Step 2 is $O(n^2*kn(m+nlogn))= O(kn^3(m+nlogn))$. Note that we can consider k a constant since our simulation in Section 3.4 uses $k\leq 10$. Thus, ***GH-Flow*()** has the worst case time complexity of $O(n^3 + n^3(m+nlogn))= O(n^3(m+nlogn))$.

As reported in [15], FGH and its improved version – EGH and BGH need to call a LP solver up to $O(m^2)$ and $O(m^3)$ times respectively, thus the total time complexity of FGH is $O(m^2n^3)$ since the time complexity of running LP is $O(n^3)$ [121]. In contrast, SSPF only uses *GH-Flow*(), which utilizes *Dijkstra*'s algorithm and *Yen*'s algorithm to generate (s_d, t_d) shortest paths and k (s_d, t_d)-shortest paths respectively for each traffic demand d. Thus, the time complexity of SSPF depends directly on the traffic matrix size $|D|=O(n^2)$. Our simulations in Section 3.4 show that SSPF runs significantly faster than FGH for various networks with up to 9900 traffic demands.

3.3.2 SSPF-R

The most time consuming step in SSPF-R is Step 3. As described in Section 3.3.1, *argmax* requires $O(m)$ while ***H-Select-e*()** requires $O(n^2)$; here SSPF-R uses either function. Step 3 also takes $O(n^3(m+nlogn))$; see its calculation in Section 3.3.1. This step is repeated $O(m)$ times for each deleted cable, and SSPF-R considers $\Re$ consecutive cables. Thus, SSPF-R has computational time complexity of $O(\Re*m*(m+n^3(m+nlogn)))$ when using *argmax*, and $O(\Re*m*(n^2+n^3(m+nlogn)))$ when using ***H-Select-e*()** functions. In either case, its complexity is $O(\Re mn^3(m+nlogn)))$. Since $\Re\leq|Q|\leq m$, its worst case time complexity becomes $O(m^2n^3(m+nlogn)))$.

3.4 Evaluation

In this section, we evaluate the performance of SSPF; namely, SSPF-1, SSPF-2, and SSPF-R. We first describe our experimental setup in Section 3.4.1. Then, in Section 3.4.2, we evaluate the performance of SSPF, and its versions, using both synthetic and realistic topologies for MLU constraint U_T=1.0. For this case, their performance is evaluated in terms of power saving and running time for w_{ij}=1 (Section 3.4.2.1) and for variable w_{ij} (Section 3.4.2.2). Further, we analyze the effects of switching off link on link utilization (Section 3.4.2.3), and on path length (Section 3.4.2.4). In Section 3.4.2, we also compare the performances of our algorithms against the state-of-the-art technique – FGH [15]. To further evaluate the performances of our algorithms, in Section 3.4.3, we evaluate them on networks with variable U_T and bundle sizes. Finally, in Section 3.4.4, we use our techniques on the Abilene network with its 288 traffic demands over 24 hours period, each of which corresponds the traffic demand recorded at every five minutes; these datasets are obtained from [99].

3.4.1 Methodology

Table 2.3 summarizes the different network topologies used in our simulations, which includes the three topologies used in [15]; *i.e.*, F_Abilene and two synthetic topologies - a two level hierarchical graph (Hier50), and the Waxman graph (Wax50). To further evaluate the performance of SSPF and FGH, we used GT-ITM [102] to generate three random graphs Geo10, Geo30, and Geo50, shown in Table 2.3, that represent small, medium and large topologies respectively; Geo50 contains 50 nodes and 434 links. To show the efficiency of the algorithm for the time period traffic, we evaluate the SSPF with Abilene, shown in Table 2.3. Further, we used the traffic matrices from [99], measured every five minutes over a 24 hours period. We used Abilene and their 288 different traffic demands in Section 3.4.4.

Each link in the F_Abilene and Wax50 have capacity c_{ij}=10000 and c_{ij}=1000 respectively, while the capacity of links in Hier50 is either 1000 or 200; we assume the same capacity unit (*e.g.*, megabytes per second) for both link capacity and traffic demand. Each link in Abilene has either c_{ij}=9920 or c_{ij}=2480. For the three random topologies, each link has capacity c_{ij}=1000 and Hier100 has capacity c_{ij}=10000.

We have implemented SSPF in *Java 6*. For FGH, we used its implementation provided by the authors of [15]. The authors of [15] ran FGH on a Window machine and used the AMPL/CPLEX to solve the LP in [15]. In our simulation, we replaced AMPL/CPLEX with a Linux-based GLPK [108]. Note that FGH's running times reported in [15] are significantly slower as compared to those generated in our simulations. For example, the authors of [15] reported that FGH for Wax50 required up to 50±20 minutes while in our simulation, the algorithm took only 5±2 minutes. However, the power savings reported in [15] for FGH on the three topologies (*i.e.*, Abilene, Wax50, and Hier50) are equivalent to our results; we used the results obtained in our simulations for the FGH's running time and power saving. Using Equation (2.1), we compute the power saving as the ratio between total powered-off cables and all cables in the network.

We ran all the algorithms on a Linux machine with *Fedora 10* (2.6.x kernel), 1024 MB memory and 28GB hard disk. For SSPF-1, SSPF-2 and SSPF-R, we set k=100, and SSPF-R uses the *argmax* function. We ran each algorithm five times, and calculated its average CPU time.

Table 3.1 Power Saving for Various Networks with w_{ij}=1, U_T=1.0

Topology	Power Saving (%)				Running Time (Second)			
	SSPF-1	SSPF-2	SSPF-R	FGH	SSPF-1	SSPF-2	SSPF-R	FGH
F_Abilene	50	46.3	51.2	46.3	0.385	0.392	2.036	79.6
Hier50	37.8	37.2	37.8	37.2	3.591	4.394	41.3	313.7
Wax50	56.8	60.4	63.3	53.3	2.969	5.723	163.28	359.3
Geo10	53.6	53.6	57.1	46.4	0.143	0.144	0.252	1.1
Geo30	72.1	72.8	73.5	69.9	0.564	1.310	18.225	85.7
Geo50	86.4	85.9	86.6	84.8	2.875	17.872	490.3	1395.9
Hier100	49.7	49.7	50.3	N/A	13.462	55.86	514.24	N/A

3.4.2 Performance Evaluation for U_T=1.0

3.4.2.1 Power Savings and Running Times for w$_{ij}$=1

Table 3.1 shows the power savings and running times for all versions of SSPF when each link has equal bundle size w_{ij}=1 and required link utilization U_T=1.0. We see that all algorithms are able to save power ranging from 37.2% to 86.6%. SSPF-1 and SSPF-2 produce almost equivalent power savings for all tested networks. However, SSPF-1 runs faster than SSPF-2 since the latter uses ***H-Select-e*()**, which has a time complexity of $O(|D|=n^2)$, in contrast to the *argmax* function in SSPF-1 which takes $O(m)$. As SSPF-2 may also produce better result for certain type of networks, *e.g.*, Wax50, we suggest running both alternative algorithms and use their best results. However, when faster running time is important, SSPF-1 is the better alternative. Table 3.1 shows that SSPF-R always produces better power savings than SSPF-1 and SSPF-2. Since SSPF-R, as described in Section 3.2.2, runs either SSPF-1 or SSPF-2 repeatedly and selects the best result from all possible outcomes, SSPF-R is guaranteed to always produce at least the power savings of SSPF-1 or SSPF-2. However, as a trade-off, the running time of SSPF-R is always slower than either

SSPF-1 or SSPF-2.

To further evaluate the efficiency and effectiveness of our SSPF, we have compared it with FGH [15]. FGH allows a demand to be routed through more than one path while SSPF restricts its routing through only a single path, allowing simpler routing protocol. As shown in Table 3.1, FGH, in most cases, produces inferior results compared to all versions of SSPF while using significantly more computational time. FGH produces power savings ranging from 1.9% to 25.3% worse than SSPF-1, and up to 28% worse than SSPF-2. Further, while producing better results, SSPF-1, SSPF-2, and SSPF-R take only, respectively, 0.21% to 13%, 0.49% to 13.09%, and 2.54% to 35.1% running time of FGH. Note that FGH failed to produce a result, denoted as ‘N/A’, for Hier100 after running for three hours.

To further evaluate the performances of our algorithms, we compare their results with the upper bound (UB) and lower bound (LB) of power savings. To generate the bounds, like in [15], we first consider the linear-programming version of our ILP problem stated in Equation (3.1) to (3.4), *i.e.*, replace $\sum_{(i,j)\in E} n_{ij}$ in Equation (3.1) with $\sum_{(i,j)\in E} f_{ij}$ to minimize the total flow over all links. Then, we use GLPK [108] to obtain the minimum flow of each link while satisfying the constraints in Equation (3.2) to (3.4). To obtain a UB on the power saving, like in [15], we “round down” the number of cables for each link needed to carry the traffic obtained by the solution. For example, for f_{08}=8.5 in Fig. 3.2, the upper bound of powered-off cables in e_{08} is $\lfloor f_{ij}/(c_{ij}/w_{ij}) \rfloor = \lfloor 8.5/(10/2) \rfloor = 1$. Note that as stated in [15], no flow assignment that satisfies all the demands can use fewer cables than those used in the upper bound. A lower bound is obtained similarly by “rounding up” the number of cables in each link, *i.e.*, for the example, the lower bound of powered-off cables for e_{08} is $\lceil f_{ij}/(c_{ij}/w_{ij}) \rceil = \lceil 8.5/(10/2) \rceil = 2$.

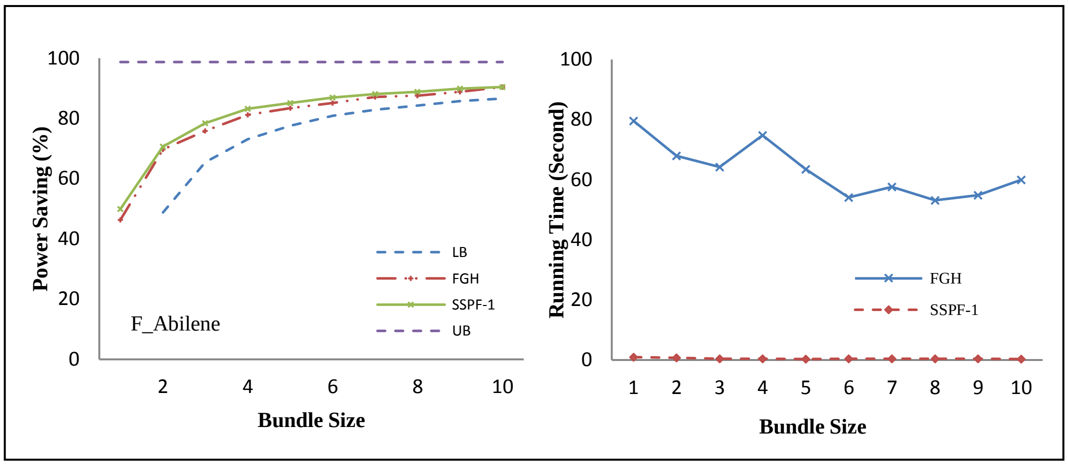


Figure 3.7 Power saving and running time of SSPF on F_Abilene

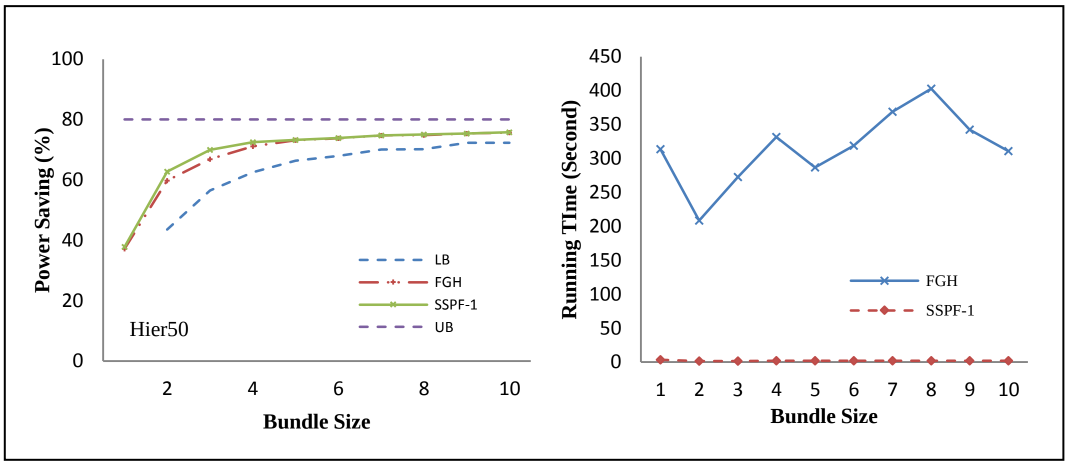


Figure 3.8 Power saving and running time of SSPF on Hier50

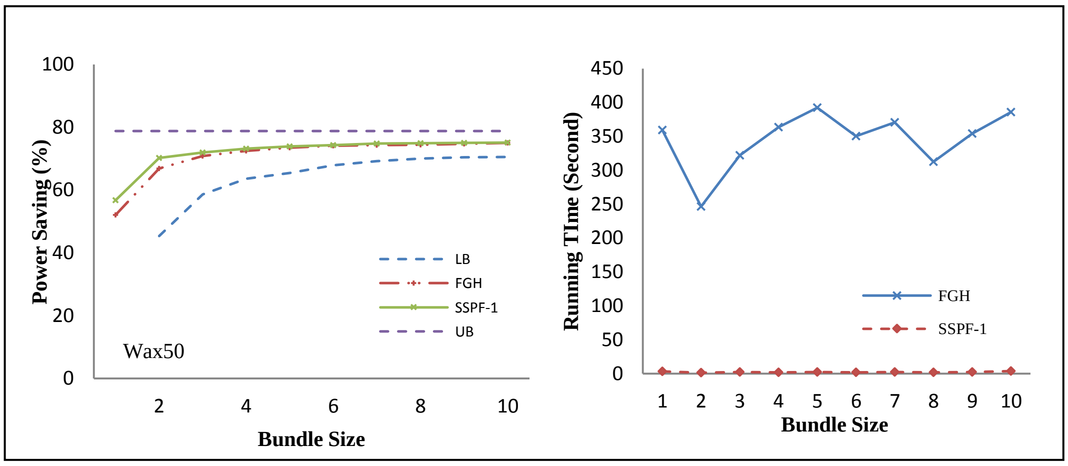


Figure 3.9 Power saving and running time of SSPF on Wax50

As shown in Fig. 3.7 to 3.9, the power savings produced by our SSPF algorithms and FGH are in between their LB and UB. In particular, the power savings produced by

SSPF-1 are between 3.7% at w_{ij}=10 and 20% at w_{ij}=1 off from the UB on Waxman network. Further, SSPF-1 could improve the power saving generated by LB between 4.6% at w_{ij}=10 and 42% at w_{ij}=2 on Waxman network; for w_{ij}=1, LB could not power-off any cable.

3.4.2.2 Power Savings and Running Times for Variable w_{ij}

We further evaluated SSPF-1 using the F_Abilene, Hier50 and Wax50 topologies when their bundle size, w_{ij}, increases from 1 to 10. For Abilene, as shown in Fig. 3.7, the power savings produced by the algorithm increases sharply when w_{ij} increases from one to three; *i.e.*, 50% to 82.1%. Similarly, for Hier50 and Wax50, increasing w_{ij} from 1 to 3 also significantly reduces the power consumption. As a comparison, the figure shows the results for FGH; as shown, SSPF-1 slightly outperforms FGH in term of power saving for all topologies. The result contradicts the intuition that the less restrictive problem (*i.e.*, to allow demand routed through one or more paths, and hence more path selection flexibility) would lead to a better power saving. We believe SSPF produces better results because it is able to switch off cables in more optimal order than using the LP solver in FGH. Further, consistent with the reported running time in Section 3.4.2 for w_{ij}=1, Fig. 3.7, 3.8 and 3.9 show that SSPF-1 requires significantly less CPU time than FGH for w_{ij}>1; *i.e.*, on average 0.512 seconds versus 62.99 seconds for Abilene, 2.068 versus 315.62 for Hier50, and 2.33 versus 345.59 for Wax50.

3.4.2.3 Effects on Link Utilization

This section analyses the effects of using fewer cables, thus saving power, on the average Link Utilization (LU), for U_T=1.0 and w_{ij}=1, calculated using the following Equation (3.5):

$$Ave(LU) = (\sum_{(i,j)\in E}((f_{ij}w_{ij})/(n_{ij}c_{ij})))/m' \qquad (3.5)$$

Note that m' is the total number of powered-on links, $((f_{ij}w_{ij})/(n_{ij}c_{ij}))$ is the link utilization of (*i*, *j*) that is ignored when n_{ij}=0 since the link is switched off when its cables are all off. Table 3.2 shows the MLU and average LU of the generated topology (after turning off cables) using SSPF and FGH; both approaches produce equivalent results. As a benchmark, we have compared the results with MLU and average LU, before turning off cables, when each traffic demand is routed through its SP. Table 3.2 shows that the SP routing using all cables in the F_Abilene, Hier50, and Wax50 results in MLU of 65.5%, 100%, and 92.9%, respectively; the MLU for the other networks is less than 30%. Further, the average LU using SP ranges from 0.9% to 24.1%; low average link utilization is expected during off-peak period, and in general, all algorithms achieve high percentage of power savings because the network has low link utilization. As shown in the table SSPF and FGH increase the MLU and average LU of the networks as compared to SP. The results are expected because when fewer cables are used to carry the same amount of flow, each cable carries more flow. However, we observe that it is possible to power-off higher percentage of cables in a network that has the higher average link utilization; see Wax50 and Geo10 with average LU of 24.1% and 0.9% (Table 3.2) but with power savings of 63.3% and 57.1% (Table 3.1), respectively.

Table 3.2 MLU (%) and Average Link Utilization (%) when w_{ij}=1 and U_T=1.0

Network	SSPF-1		SSPF-2		SSPF-R		FGH		SP	
	MLU	Ave	MLU	Ave	MLU	Ave	MLU	Ave	MLU	Ave
F_Abilene	80.35	16.33	83.25	16.32	98.13	16.77	80.48	15.22	65.5	4.9
Hier50	100	51.26	100	50.56	100	51.51	100	48.36	100	22.9
Wax50	99.95	72.9	100	61.8	100	71.48	100	77.26	92.9	24.1
Geo10	4.49	3.32	5.99	3.32	4.65	3.32	5.87	2.45	2.17	0.9
Geo30	61.17	14.98	68.87	19.63	69.25	21.32	53.34	16.24	14.86	1.9
Geo50	86.04	38.76	99.96	37.34	100	40.62	90.2	31.12	24.2	1.4
Hier100	58.84	14.82	67.15	15.18	59.96	16.83	N/A	N/A	26.25	2.9

To further evaluate the effects of shutting down cables on the remaining link's utilization, we show in Fig. 3.10 (Left), Fig. 3.11 (Left) and Fig. 3.12 (Left) the cumulative distribution function (CDF) of the link utilizations in F_Abilene, Wax50 and Hier100 networks, respectively using SSPF-1, SSPF-2, SSPF-R, FGH and SP routings. The results for other topologies are similar and thus not shown here. All four power-saving routing algorithms increase link utilization as compared to using SP, but to no more than 0.85 and 0.6 for F_Abilene and Hier100 respectively; FGH cannot obtain any results in a reasonable time for Hier100, and thus Fig. 3.12 omits FGH. Among four algorithms, SSPF-2 and SSPF-R perform the best for F_Abilene and Hier100, respectively; for Wax50, they produce similar results. SSPF-2 increases the number of links with utilization above 0.4 from 3% to 14% as a tradeoff for reducing power by about 37% on F_Abilene; see Fig. 3.10 and Table 3.1.

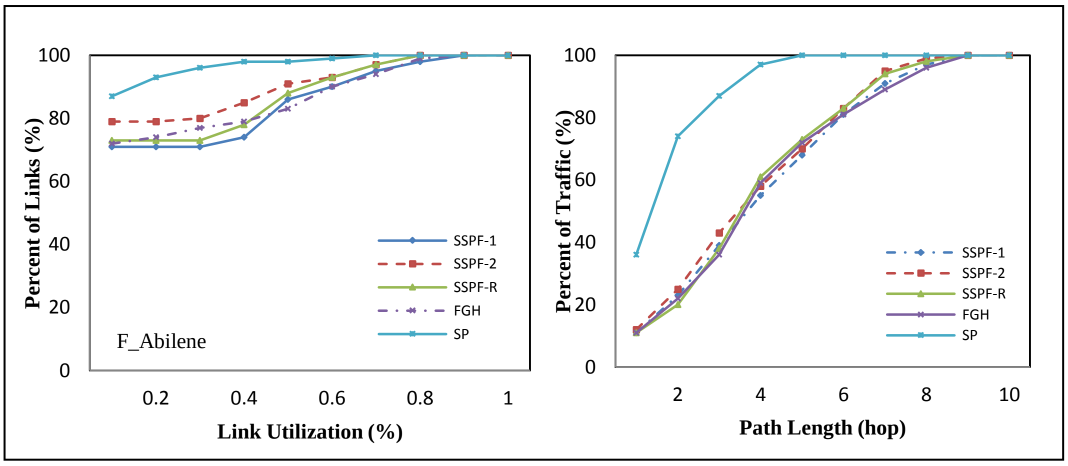


Figure 3.10 CDF of link utilization and path length on the F_Abilene

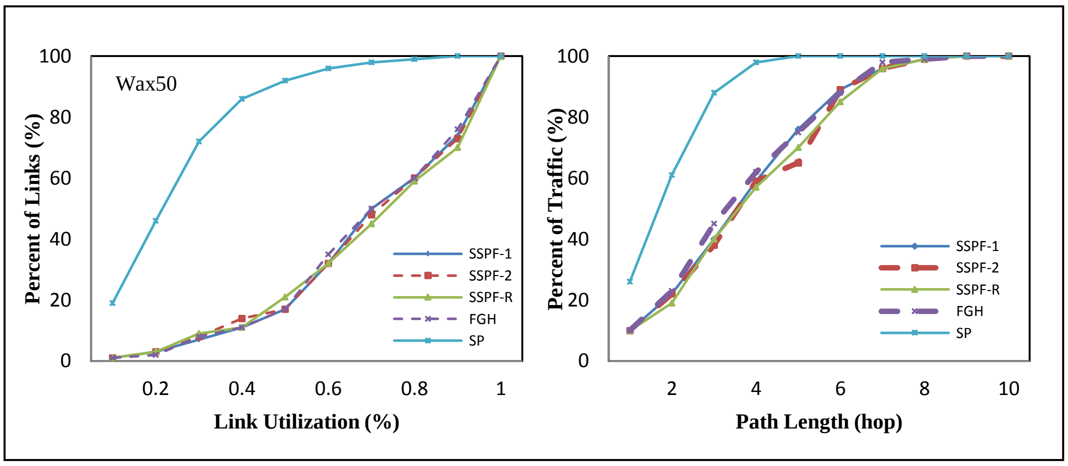


Figure 3.11 CDF of link utilization and path length on the Wax50

Similarly, for Hier100, SSPF-R increases the number of links with utilization above 0.1 from 3% to 23% while reducing power usage using SP by 53%; see Fig. 3.12 and Table 3.1. As expected, when the traffic demands are considerably larger than available link resources, *i.e.*, for Wax50 as shown in Fig. 3.11, all power-saving routing algorithms, while reducing power usage by close to 60% (see Table 3.1), result in larger ratio of link utilizations as compared to SP. However, as will be discussed in Section 3.4.3, our SSPF approach allows different MLU settings, *e.g.*, no more than 0.4 as the standard practice in ISP, while maximizing power savings.

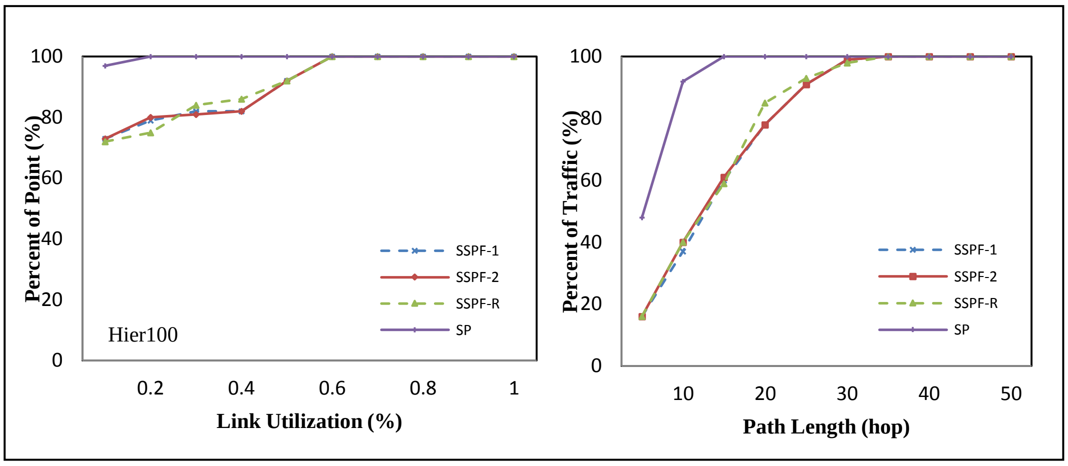


Figure 3.12 CDF of link utilization and path length on the Hier100

3.4.2.4 Effects on Path Length

Table 3.3 shows the effect of turning off cables, using SSPF and FGH, on the average path length $L(MP_d)$, for U_T=1.0 and w_{ij}=1, calculated as follows,

$$Ave(L(MP_d))=(\sum_{d \in D} L(MP_d))/|D| \qquad (3.6)$$

Note that $L(MP_d)$ is the length of the path used to route demand *d*, *i.e.*, its hop counts. For FGH, as demands may be routed through multiple paths, we used the maximum path length. As shown in Table 3.3 turning off cables, either using SSPF or FGH, have a significant impact on the average path length; SSPF and FGH produced similar results. The results are expected since turning off cables forces some part of the traffic to be routed through longer (non-shortest) paths. Notice, however, that our simulation considers only hop counts as the path lengths, which do not reflect the actual path lengths that include several other factors such as the queuing delays.

Fig. 3.10 (Right), Fig. 3.11(Right) and Fig. 3.12 (Right) show the CDF of the path lengths in F_Abilene, Wax50 and Hier100 networks using SSPF-1, SSPF-2, SSPF-R, FGH and SP routings, which evaluate the impact of power savings in path length. The results for other topologies are similar and thus not shown here. All power-saving routings produce similar results except FGH that cannot obtain the results in reasonable time for Hier100, and Fig. 3.12(Right) omits FGH. For F_Abilene, as shown in Fig. 3.10(Right), while decreasing power by 51.2%, the power saving routings, *e.g.*, SSPF-R, reduces the percentage of routes that have delay of one hop from 37% using SP to 11%, and those with two hops from 74% using SP to 25%. Notice that the longest path using SP is five hops, while that using SSPF-R is 8 hops, an increase of 60%. However, the longest path in SSPF-R is shorter than the network diameter of F_Abilene, *i.e.*, 9 hops. As shown in Fig. 3.11(Right) and Fig. 3.12(Right), the effects of shutting down cables on path length on Wax50 and Hier100 are similar to in F_Abilene in Fig. 3.10(Right). Further, similar to F_Abilene, the longest path of routes using SSPF-R for Wax50 and Hier100 is also shorter than the network diameter.

Table 3.3 Average Path Length (hop) when w_{ij}=1 and U_T=1.0

Network	SSPF-1	SSPF-2	SSPF-R	FGH	SP
Abilene	4.37	4.32	4.53	4.42	2.05
Hier50	4.83	4.71	4.9	4.92	3.17
Wax50	4.16	4.23	4.3	4.26	2.2
Geo10	1.92	1.92	1.97	1.89	1.18
Geo30	5.5	5.5	5.61	5.47	1.36
Geo50	7.4	7.36	7.4	7.35	1.0
Hier100	13.83	13.72	13.83	N/A	5.81

3.4.3 Power Savings for Different U_T

As discussed in the Section 3.4.2.3, SSPF affects link utilization as fewer links are used to carry traffic, In this section, we investigate the effect of using 10 different MLU bounds, *i.e.*, U_T between 0.1 and 1.0 – with 0.1 increments, on power savings achievable using our SSPF approach, for bundle size w_{ij} between 1 and 10 while running SSPF-R; we obtained similar results using SSPF-1 and SSPF-2. Table 3.4 and 3.5 shows results for the F_Abilene and Hier100, respectively where UT_X denotes MLU bound X; *e.g.*, UT_0.4 means U_T=0.4, and $ND = \sum_{(i,j)\in E}(w_{ij} - n_{ij})$ be the total number of switched-off cables.

Table 3.4 Power Saving on the F_Abilene using SSPF-R

Bundle Size	Total	UT_0.4		UT_0.5		UT_0.7		UT_0.9	
w_{ij}	Cables	*ND*	(%)	*ND*	(%)	*ND*	(%)	*ND*	(%)
1	82	31	37.8	36	43.9	39	47.6	42	51.2
2	164	103	62.8	111	67.7	115	70.1	116	70.7
3	246	164	66.7	189	76.8	192	78	193	78.4
4	328	244	74.3	253	77.1	270	82.3	275	83.2
5	410	323	78.8	342	83.4	347	84.6	350	85.1
6	492	395	80.3	409	83.5	425	86.3	429	87
7	574	474	82.6	480	83.6	503	87.6	508	88.1
8	656	556	84.8	557	84.9	580	88.4	584	88.9
9	738	629	85.2	634	85.9	658	89.1	663	90
10	820	709	86.5	711	86.7	716	87.3	740	90.5

Table 3.5 Power Saving of Hier100 using SSPF-R

Bundle Size	Total	UT_0.2		UT_0.3		UT_0.4		UT_0.7	
w_{ij}	Cables	*ND*	(%)	*ND*	(%)	*ND*	(%)	*ND*	(%)
1	286	117	40.9	124	43.4	133	46.5	143	50.3
2	572	307	53.7	394	68.9	408	71	420	50.3
3	858	589	68.7	593	69.1	686	79.8	701	73.4
4	1144	871	76.1	876	76.6	880	83.8	981	81.7
5	1430	1156	80.8	1156	80.8	1162	81.6	1260	85.8
6	1716	1440	83.9	1441	84	1446	84.7	1541	88.1
7	2002	1723	86	1722	86	1728	86.7	1827	89.8
8	2288	2002	87.5	2004	87.6	2009	87.7	2096	91.3
9	2574	2287	88.8	2292	89	2293	89	2378	91.6
10	2860	2571	89.9	2574	90	2582	90.2	2658	92.4

The results for other topologies have the similar trend when the bundle size increases from 1 to 10 under different U_T, and thus not shown here. For F_Abilene, SSPF-R fails to save power with $U_T \leq 0.3$; these results are also omitted. Further, for each algorithm, increasing U_T from 0.5 to 0.6, or from 0.7 to 0.8, or from 0.9 to 1.0 does not affect power savings, and thus, Table 3.5 only shows the results for UT_0.4, UT_0.5, UT_0.7 and UT_0.9. For Hier100 network, SSPF-R fails to save power with $U_T \leq 0.1$; thus we do not show its result in Table 3.5. Similar to the F_Abilene network, other than U_T=0.2 and U_T=0.3, increasing U_T from 0.4 to 0.6, or from 0.7 to 1.0 does not affect power savings. To this end, Table 3.5 only shows the results for UT_0.2, UT_0.3, UT_0.4 and UT_0.7.

As shown in Table 3.4 and 3.5, power savings increase, for each U_T, when the bundle size increases from 1 to 10 since there are more idle cables. Notice that there is a large increase in power saving, *i.e.*, by about 25% when the bundle size in F_Abilene is increased from 1 to 2. Similarly, a large increase in power saving also occurs in Hier100 when the bundle size increases from 1 to 4. As shown in Table 3.4, our SSPF approach is able to reduce power usage in Abilene between 37.8% and 86.5% when its bundle size is set between w_{ij}=1 and w_{ij}=10, respectively, even when each link utilization is set to no more than 40%, which is within the standard practice set by ISP [15]. Similarly, for MLU≤40%, SSPF-R is able to reduce the power expenditure of Hier100 by 46.5% to 90.2%, for bundle sizes between 1 and 10, respectively. For both F_Abilene and Hier100, relaxing the link utilization constraint to higher values allow our SSPF algorithm to find better alternative paths, and thus further reducing power usage.

Tables 3.4 and 3.5 also show that for each bundle size, there is negligible effect from using different U_T constraints on power saving. As an example, Table 3.5 shows that, for w_{ij}=1, using significantly more restrictive U_T=0.2, as compared to U_T=0.7, only slightly decreases power saving to 40.9% from 50.3%. This effect may be due to the low traffic levels. Recall that for Hier100, each traffic flow for demand d is computed as $b_d=10*rn_1*rn_2$, where rn_1 and rn_2 are two random numbers between 0 and 1; thus, $0 \leq b_d \leq 10$ is a small value as compared to the capacity of each link, *i.e.*, c_{ij}=10000.

To see the effect of traffic levels, *i.e.*, different flow size in traffic demands, we generated five other traffic levels for Hier100 when w_{ij}=1. Specifically, we multiplied each b_d used to generate Table 3.5 for w_{ij}=1 with five scaling factors: 0.5, 2, 3, 4, 5. Thus, the smallest scale, *i.e.*, 0.5, sets $0 \leq b_d \leq 5$, while the largest, *i.e.*, 5, generates $0 \leq b_d \leq 50$; the former simulates lower traffic level while the latter assumes a more congested network as compared to the traffic level used in Table 3.5 with $0 \leq b_d \leq 10$. As expected, Fig. 3.13 shows that, for each constraint U_T, SSPF-R produces higher power savings for lighter traffic flows, *e.g.*, 51.2% for 0.5*b_d versus 46.5% for b_d with U_T=0.6, and lower power savings in more congested networks, *e.g.*, only switching-off 39.16% links for 3*b_d. Further, consistent with our results in Table 3.4 and 3.5, the figure shows that for each traffic level, U_T does not significantly affect the power savings produced by SSPF-R, *e.g.*, for 0.5*b_d the savings increase only from 41.96% for U_T=0.1 to 51.2% for U_T=1.0. We observed that the different traffic levels only affect the feasibility of traffic routings. For example, there is no feasible routing for demands using $0 \leq b_d \leq 50$ ($0 \leq b_d \leq 20$) with $U_T \leq 0.8$ ($U_T \leq 0.3$), *i.e.*, the SP routing, described in Section 3.4.2.3, and SSPF-R fail to produce results due to insufficient link capacities for the given set of traffic demands.

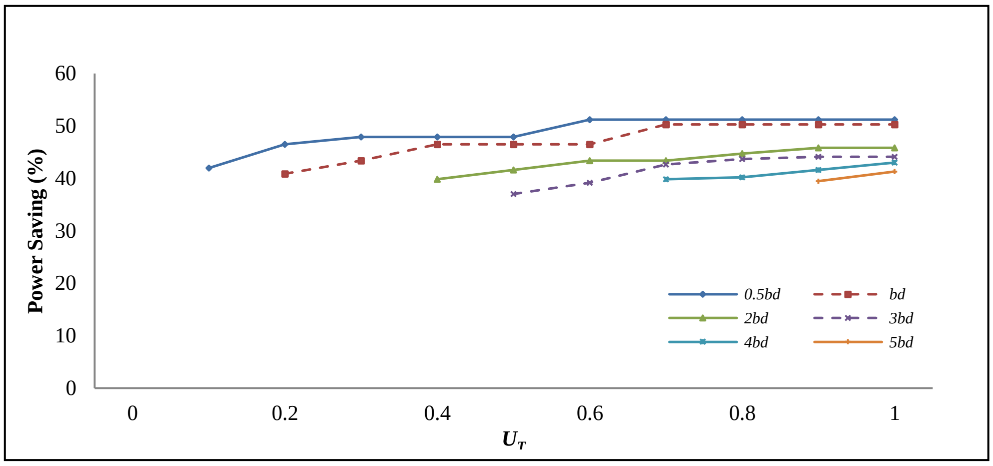


Figure 3.13 Impact of different traffic levels and U_T on power saving in Hier100

3.4.4 SSPF versus Optimal Solution

Ideally, we aim to evaluate the effectiveness of our algorithms against an optimal solution (SP-OS) using the ILP, shown in Equation (3.1) to Equation (3.4), for the SP-EAR problem in term of power savings and running times. Unfortunately, due to the exponential time complexity of the problem, as described in Section 3.2, we could not generate SP-OS for the medium and large networks. As an alternative, we only use SP-OS for the small network in Fig. 2.1. For fair comparison against FGH, we set U_T=100%, and w_{ij}=1 for each link (i, j); the results are shown in Table 3.6.

Table 3.6 SSPF and FGH versus SP-OS for the Network in Fig. 2.1

Mechanism	**PS (%)**	**Time (second)**
SP-OS	50	2
SSPF	40	0.034
FGH	40	0.4

From Table 3.6, we can find that SSPF performs the same as FGH in term of power saving, which is 20% lower than SP-OS. However, SP-OS runs 58.82 times slower than SSPF. Further, the 0.034 seconds running time of SSPF is only 8.5% of the time used by FGH.

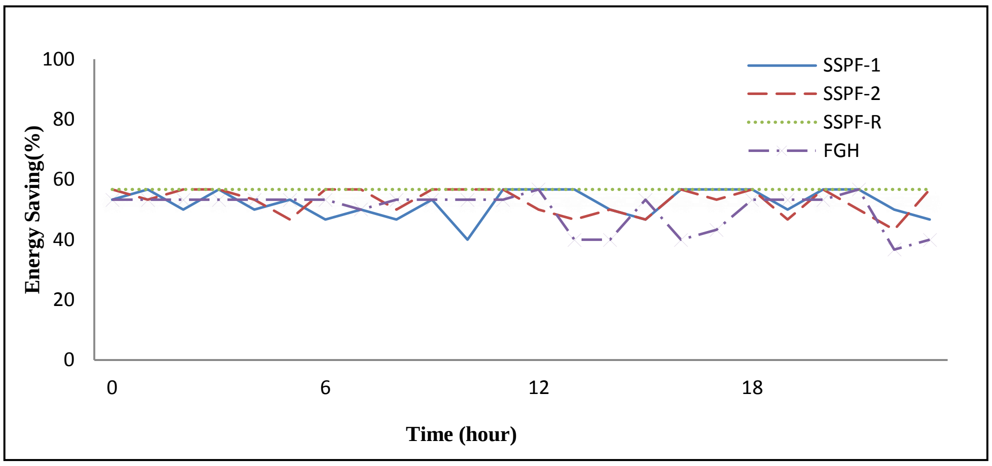


Figure 3.14 The potential power saving of SSPF on Abilene on Sept. 5, 2004

3.4.5 Potential Power Savings on Abilene

Fig. 3.14 shows the potential power savings on Abilene [99] using SSPF-1, SSPF-2, and SSPF-R. We ran each algorithm for 288 different traffic demands from [99]; each demand represents traffic traces recorded every five minutes over 24 hours on September 5, 2004. For this experiment, SSPF-1, SSPF-2, and SSPF-R each required on average 0.21, 0.24 and 0.547 CPU seconds, respectively, to produce results for each five minutes of traffic demand. As shown in Fig. 3.14, SSPF-R consistently produces the best power savings compared to SSPF-1 and SSPF-2. It is interesting to observe that SSPF-2 outperforms SSPF-1 from 0:00 through 11:00 hours and from about 22:30 through 24:00; at other times, SSPF-1 in most cases produced better results than SSPF-2. Further, power savings produced by SSPF-R are always the same for each of the 288 different traffic demands. As another comparison, the figure shows that FGH in all (most) cases perform worse than SSPF-R (either SSPF-1 or SSPF-2).

3.5 Summary

In this chapter, we have proposed an efficient and effective heuristic approach, SSPF, to minimize network power usage while satisfying all network traffic demands subject to a given MLU constraint. Our approach aims to switch off redundant cables in core routers using bundled links. However, unlike [15], our approach routes each traffic demand onto one single path, which simplifies routing. We have used the F_Abilene topology - with both real and synthetic traffic matrices and several larger randomly generated topologies - with synthetic traffic matrices to evaluate its performances. The simulation shows that our approach could potentially save up to 56.7% of the power expenditure incurred by the Abilene topology, as per the 24 hours traffic demands, measured every five minutes, obtained from [99]. Further, our heuristic significantly outperforms the approaches in [15], both in terms of their running times and power savings. We will investigate multiple paths green routing in Chapter 4, and 2DP green routing in Chapter 5.

Chapter 4

Multi-path Green Routing

In this chapter, we propose an efficient approach - MSPF to reduce the overall power consumption of the backbone network of ISPs. Specifically, MSPF guarantees that each link, considering only its powered on cables, has a pre-configured MLU threshold $0 \le U_T \le 1.0$. Further, MSPF allows each traffic demand to be routed through one or more paths but with path length no longer than a given constraint; *i.e.*, either the network's diameter or λ times the hop count of its original shortest path, for a given delay multiplier $1.0 \le \lambda \le 2.0$. In addition, MSPF considers the possibility of turning-off routers to further reduce power consumption. In other words, our approach aims to find the minimum set of operational devices, *i.e.*, routers and cables, which can be used to route a given set of traffic demands while satisfying users' MLU and path length constraints. Note that the preliminary version of MSPF that only considers switching off cables has been published in [45], and the extended version that considers switching off both nodes and cables has been published in [120].

The problem in this chapter generalizes the NP-complete problem in [15] that excludes MLU and path length constraints. Our problem also extends the problem in [16] that sets each link to contain only one cable, *i.e.*, w_{ij}=1. Unlike our MSPF, both solutions in [15] and [16] ignore the possibility of turning off routers to further reduce network power consumption. As shown in [88], routers/switches consume orders of magnitude more power. In this respect, switching off routers/switches is expected to further reduce network's power consumption. Thus, considering all powered-on routers, for w_{ij}=1 and λ=2.0, our solution reduces to that in [16], and it becomes that in [15] if we ignore these two constraints for $w_{ij} \ge 1$. Simulations in Section 4.4 show that MSPF is more efficient and as effective, if not more, as compared to the solutions in [15] and [16] while capable of solving more general problems. Specifically, considering only switched off links, as compared to GreenTE [16], MSPF runs on average 99% faster while improving its power savings by 5% on tested topologies and traffic demands. In addition, MSPF requires only 0.35% of the run time of FGH, the fastest approach in [15], while yielding equivalent power savings.

The layout of this chapter is as follows. Section 4.1 gives an overview of the problem and its linear programming formulation. Section 4.2 describes two version of MSPF: MSPF-LF and MSPF-NF, and Section 4.3 analyzes the running time complexity of MSPF. Section 4.4 evaluates the performance of MSPF using both real and synthetic topologies and traffic matrices. It also provides performance comparisons against GreenTE [16] and FGH [15]. Finally, Section 4.5 concludes the chapter.

4.1 Problem Statement

Given a network $G(V, E)$ and a traffic demand set D, the Multiple Paths Energy-Aware Routing (MP-EAR) problem is to generate (i) the minimum number of powered on (or active) nodes and cables, and (ii) a set of multiple paths $P_D=\{MP_d| d=1, 2, \ldots, |D|\}$ that can be used to route traffic of each demand $d \in D$ while using only powered-on nodes and cables, subject to two constraints: (C1) the utilization of each link (i, j) is no larger than a given U_T, *i.e.*, $u_{ij} \leq U_T$, and (C2) the length of each path in MP_d for demand d is no longer than a given constraint L_d, *i.e.*, $L(sp_{dq}) \leq L_d$, $sp_{dq} \in MP_d$. In other words, the problem is to find as many nodes and cables that can be switched off while satisfying all traffic demands in D under constraints (C1) and (C2).

The two constraints are used to ensure the solution, while minimizing power usage, does not affect the QoS requirements of customers. In particular, we note that the average link utilization in backbone networks of large ISPs is deliberately set to around 30-40% in order to guarantee QoS requirement [15]. For path length, we consider two path length constraints when routing each demand d with powered-off nodes and cables: (C2.1) each d must be routed through one or more paths with a bounded delay $L_d \leq \text{ND}$, where ND is the diameter of original network (*i.e.*, network with no switched-off cable), or (C2.2) each d must be routed through one of more paths with threshold delay $L_d \leq l_d \lambda$, where l_d is the length of the shortest path to route demand d in the original network, and $1.0 \leq \lambda \leq 2.0$ is a given delay multiplier.

Let $x_{dp} \leq b_d$ denotes a continuous non-negative traffic allocated to path sp_{dq}, and δ_{ij}^{dq} be a binary variable that is set to 1 if link $(i, j) \in sp_{dq}$. Let f_{ij} be the total flow on link (i, j). For a switched-on node v and its every switched-on outgoing link (v, j), we compute its out-going flow and total capacity as $f_v = \sum_{(v,j)\in E} f_{vj}$ and $c_v = \sum_{(v,j)\in E} c_{vj}$ respectively. The MP-EAR problem can be formulated as follows.

Minimize:

$$\sum\nolimits_{(i,j)\in E} p_{ij} n_{ij} + \sum\nolimits_{v\in V} p_v ON_v \tag{4.1}$$

Subject to

$$\sum_{j=1}^{n}\sum_{q=1}^{|SP_d|}\delta_{ij}^{dq}x_{dq} - \sum_{j=1}^{n}\sum_{q=1}^{|SP_d|}\delta_{ji}^{dq}x_{dq} = \begin{cases} b_d, i = s_d \\ -b_d, i = t_d \\ 0, i \neq s_d, t_d \end{cases}, \forall d \in D \tag{4.2}$$

$$f_{ij} = \sum\nolimits_{d\in D}\sum\nolimits_{q=1}^{|SP_d|}\delta_{ij}^{dq}x_{dq} \leq U_T(n_{ij}/w_{ij})c_{ij}, \forall (i,j)\in E \tag{4.3}$$

$$f_v = \sum\nolimits_{(v,j)\in E}\sum\nolimits_{d\in D}\sum\nolimits_{q=1}^{|SP_d|}\delta_{vj}^{dq}x_{dq} \leq c_v ON_v, \forall v \in V \tag{4.4}$$

$$\sum\nolimits_{(i,v)\in E}\left\lceil n_{iv}/w_{iv}\right\rceil + \sum\nolimits_{(v,j)\in E}\left\lceil n_{vj}/w_{vj}\right\rceil \leq 2nON_v, \forall v \in V \tag{4.5}$$

$$0 < \sum\nolimits_{(i,j)\in sp_{dq}} \delta_{ij}^{dq} \leq L_d, \forall d \in D \quad q = 1, \ldots, |SP_d| \tag{4.6}$$

Equation (4.1) quantifies the total power consumed by all active cables and routers. Equation (4.2) is the standard flow conservation constraint that ensures no flow is lost, and ensures that the sum of the flows leaving the source, or entering the destination of demand d sums to b_d. Equation (4.3) computes the total flow traversing each link while restricting it to within the link's allowable capacity utilization, *i.e.*, $U_T(n_{ij}/w_{ij})c_{ij}$. Note that, when n_{ij}=0, *i.e.*, no cable in (i, j) is on, the flow through the link f_{ij}=0. Equation (4.4) limits the total traffic throughput of a node v to no larger than c_v. With regards to constraint (4.5), node v can be shut down only when all cables of the node's incident links are powered off, *i.e.*, n_{ij}=0 for each link (i, j) connected to the node v. Lastly, constraint (4.6) ensures that the path length of routing path sp_{dq} for demand d, *i.e.*, $L(sp_{dq}) = \sum_{(i,j)\in sp_{dq}} \delta_{ij}^{dq}$, is no longer than a given threshold L_d. The resulting formulation is the Multiple Constraints Path (MCP) problem that is NP-complete [110].

4.2 Multiple Paths by Shortest Path First

This section describes two versions of our heuristic, called MSPF, to solve the aforementioned NP-complete problem. The first version, MSPF-Link First (MSPF-LF), switches off as many cables as possible with the remaining cables used to route all traffic demands. Moreover, any nodes with their incident links switched off are also powered down. In contrast, the second version, MSPF-Node First (MSPF-NF), aims to switch off as many nodes as possible by deleting their incident links, and hence cables. In other words, the NF version prioritizes switching off nodes over links since the power consumption of a node is significantly larger than a link.

4.2.1 MSPF-LF

Fig. 4.1 describes the main steps of MSPF-LF; for a demand set D, let B_D be a sequence of switched-off cables, V_D be the set of switched-off nodes, and P_D be the set of routing paths through the remaining routers and cables in the network. In **Step 1**, *Yen*'s algorithm [30] is used to generate k shortest paths, SP_d, for each demand d; each of which has delay $L(sp_{dq}) \leq L_d$. In **Step 2**, we use function ***Distribute-Flow*()** to distribute the traffic flow in each demand d through one or more candidate paths in SP_d. The function distributes the flow starting from the shortest candidate path. If the total traffic flow of d cannot be routed through the path, it uses the next shortest path in SP_d to carry the remaining flow, and so forth, until the total flow is routed. The function returns false if the traffic volume of d cannot be routed through the candidate paths in SP_d. Otherwise, it returns true and creates a set $MP_d \subseteq SP_d$ that contains all paths used to route demand d, and inserts the set in P_D. Notice that in Step 2 the function would always return true since we assume that the original network has sufficient capacity to meet the given demands. In **Step 3**, we calculate the total flow f_{ij} for each link (i, j), and compute the remaining link capacity r_{ij}, which in turn is used to calculate the maximum number of redundant cables $\lfloor r_{ij} \rfloor$ to shut down; r_{ij} is calculated using the following equation:

$$r_{ij} = U_T (n_{ij} / w_{ij}) c_{ij} - f_{ij}, \forall (i, j) \in E \tag{4.7}$$

Step 4 consists of two main sub-steps. In Step 4a), we repeatedly select a candidate cable in (i, j) to switch off; we target the cable whose link (i, j) has the largest remaining capacity r_{ij}. In Step 4b), we use the function ***Reroute-Demand*()**, described later, to reroute the flow of paths in MP_d for each demand d that is affected by the removal of a cable in (i, j). If rerouting is possible, we delete the cable and put it in the set of powered off cables B_D. Otherwise, we know that the cable must be switched on to ensure the feasibility of satisfying all demand flows; therefore, we set $fix((i, j))=true$. Steps 4a) and 4b) are repeated until each $fix((i, j))$ is $true$.

```
MSPF-LF(G(V,E), D)
OUTPUT: B_D = (b_ij | all powered off cables in ∀(i, j)∈E), V_D={v | all powered off nodes in ∀v∈V} and P_D={MP_d, ∀d∈D}
Begin
1) For each demand $d \in D$ do generate $SP_d$;
2) For each demand $d \in D$ do Call Distribute-Flow($b_d$, $SP_d$)
      If feasible then create a set $MP_d$ that contains all paths used to route demand $d$,
      and inserts the set into $P_D$.
3) For each link $(i, j)$ do calculate $r_{ij}$, remove the maximum cables such that all flows
   are still satisfied and set $fix((i, j)) \leftarrow false$;
4) Repeat
   a) Find a candidate edge $(i, j) \in E$, denoted as del_e, that has the largest remaining
      capacity per Equation (4.7), remove a cable in $(i, j)$, i.e., $n_{ij}=n_{ij}-1$, and put the
      cable in $B_D$.
   b) Call Reroute-Demand(del_e)
      (i)  If feasible then go to Step 3.
      (ii) If not feasible then retain the deleted cable, remove it from $B_D$, and set $fix((i, j)) \leftarrow true$.
      Until $fix((i, j))=true$ for every $(i, j) \in E$.
5) Remove each node $v$ that is not connected to any powered-on cable from $V$ and store
   it into $V_D$.
End
```

Figure 4.1 Algorithm MSPF-LF

Finally, **Step 5** searches for nodes that are not connected to any powered-on cables. Each such node is turned off to reduce power, and inserted into V_D.

```
Reroute-Demand(del_e)
OUTPUT:   If it has feasible solution, return true, else return false.
Begin
    For each $d \in D$ do
        $TP_d = \varnothing$;
    1)  Move all paths in $MP_d$ that contain del_e into $TP_d$;
        If all cables in link del_e are switched-off then
            $SP_d = SP_d - TP_d$;
    2)  Set $f_{ij}=f_{ij}-x_{dq}$ and $c_{ij}=c_{ij}+x_{dq}$, for each link $(i, j) \in sp_{dq}$ and each path $sp_{dq} \in TP_d$
    3)  Call Distribute-Flow( $\sum_{sp_{dq} \in TP_d} x_{dq}$ , $SP_d$);
    4)  If the flow distribution is feasible then
                Update $MP_d$ to include all paths used to route flow $\sum_{sp_{dq} \in TP_d} x_{dq}$ ;
        Else
                For each $d \in D$ do
                    Retain the previous contents of $SP_d$ and $MP_d$;
                    Retain the original value of each $f_{ij}$ and $c_{ij}$ on each path $sp_{dq} \in TP_d$;
            Return false;
    End-For
    Return true;
End
```

Figure 4.2 Function Reroute-Demand ()

As shown in Figure 4.2, Step 1 of ***Reroute-Demand*()** finds all paths in MP_d for each d that contains the candidate link *del_e*, and put the affected paths in the temporary set TP_d. If the bundle size of *del_e* is zero, each path $sp_{dq} \in TP_d$ that contains *del_e* is removed from SP_d, *i.e.*, $SP_d=SP_d-TP_d$ since the path is disconnected. Then, Step 2 reverts previously allocated flow x_{dq} to each $sp_{dq} \in TP_d$ by setting $f_{ij}=f_{ij}-x_{dq}$ and $c_{ij}=c_{ij}+x_{dq}$ for each $(i, j) \in sp_{dq}$. This step is needed because the function aims to redistribute the flow of each path in TP_d. In Step 3, we use function ***Distribute-Flow*()** to distribute all the reverted traffic flows for demand d, *i.e.*, $\sum_{sp_{dq} \in TP_d} x_{dq}$, through the remaining candidate paths in SP_d. If this is not possible for any affected demand is not feasible, *i.e.*, ***Distribute-Flow*()** returns false for any TP_d, Step 4 retains the original contents of sets SP_d and MP_d, and the flow and capacity of each link in the sets, and returns *false*. Otherwise, the step updates MP_d to include all paths that are used to route the reverted flows.

4.2.2 MSPF-NF

We propose two versions of MSPF-NF algorithm: version 1 (MSPF-NF1) and its faster version 2 (MSPF-NF2). MSPF-NF1 converts the NP-complete ILP objective in (4.1) into its equivalent LP formulation, described later. On the other hand, MSPF-NF2 uses MSPF-LF; see Fig. 4.1.

4.2.2.1 MSPF-NF1

Using a similar idea outlined in [15], and retaining the same constraints in the original problem, *i.e.*, (4.2), (4.3), (4.4), (4.5), and (4.6), we convert the MIP objective in (4.1) into a LP objective as follows:

$$\text{Minimize} \quad \sum\nolimits_{(i,j)\in E} f_{ij} \tag{4.8}$$

The new problem formulation, *i.e.*, (4.8) is useful since the LP can be used to find a feasible distribution of flows. Specifically, we use the LP formulation to solve our problem as follows. First, given input D and $G(V, E)$, we use an LP solver, *e.g.*, CPLEX [97], to obtain the flow on each link and total flow through each node. Note that the solver will find a feasible solution because the original network has sufficient resources to distribute the flows in D. Second, after removing idle nodes and links from G, we find a candidate node v with the least flow f_v from the remaining nodes in G. Next, we remove v and its incident links from G, and run the LP to check if the revised G has sufficient resources to distribute all flows in D. If the LP solver returns with a feasible solution, we remove the node and its incident links from G permanently. Otherwise, we restore v and its links and flag v as *final*. We repeat the third step until all nodes in G is marked *final*. Finally, we search for additional cables that can be turned off from the remaining links in G. For this last step, one can use

FGH to solve the LP in (4.8) but ignoring the nodes and using (4.7) to compute each link's spare capacity.

4.2.2.2 MSPF-NF2

MSPF-NF2 uses exactly the same algorithm as MSPF-LF, shown in Fig. 4.1, except for steps 3, 4 and 5. Step 3 of MSPF-NF2 calculates the total flow f_v through the node $v \in V$, and set the status of all nodes, *i.e.*, *fix*(*v*)=*false*. In contrast to finding a candidate link in MSPF-LF, in Step 4a), MSPF-NF2 chooses a candidate node that meets, in order, one or more of the following properties: (i) a node that is used least frequently in the initial SP routing to minimize the number of routing disruptions; (ii) a node with minimum degree so that switching off the node will affect fewer traffic demands, and (iii) a node with the least flow so that its flow can be redistributed easily. Note that deleting a node includes removing its entire incident links, and hence all cables in the links. Step 4b) uses a slightly modified ***Reroute-Demand*()** function to check if it can produce a feasible solution using the network's remaining nodes and cables. Specifically, the variable *del_e* in Step 1 is a set of incident links to the candidate node, which is used by the function in Step 4b) of Fig. 4.1. Thus, in this step, TP_d contains every path in MP_d that contains at least one link in *del_e*. If the function returns true, then the selected node and cables are removed permanently. Otherwise, it retains the selected nodes and their respective links. This process is repeated until all nodes are flagged true. While Step 5) in MSPF-LF checks for nodes, the step for MSPF-NF2 searches for links/cables that can be switched off; we use Step 4 of MSPF-LF for Step 5 of MSPF-NF2. Since MSPF-NF2 calculates the *k*-shortest path only once, and routes traffic demands using simple shortest path routing, this version 2 runs faster than version 1 that uses LP solver multiple times.

4.3 Time Complexity

4.3.1 Running Time of MSPF-LF

For MSPF-LF, *Yen*’s algorithm [30], see Step 1, incurs $O(kn(m+nlogn))$ time. Amongst the k paths, we select only a path that meets the corresponding delay constraint L_d; this sub step requires $O(m)$. Note that n and m are the total number of nodes and edges in G respectively. For each demand, function ***Distribute-Flow*(),** called in Step 2**,** takes $O(km)$ time. Therefore, the time complexity of Step 2 is $O(km|D|)= O(kmn^2)$ because $|D|\leq n^2$; $|D|$ is the total number of traffic demands. Step 3 requires searching all links in G and therefore has a time complexity of $O(m)$. For Step 4a), MSPF-LF uses Equation (4.7); it takes $O(m)$ to select each candidate. Thus, for m links, this step has complexity $O(m^2)$. Step 4b) uses function ***Reroute-Demand*()** that incurs a bound of $O(kmn^2)$, described later. Repeating the step m times, in total MSPF-LF has a time complexity of $O(kn(m+nlogn)+km^2n^2)$ = $O(km^2n^2)$.

Each MP_d contains at most k paths, and checking for *del_e* in Step 1 of ***Reroute-Demand*()** requires $O(m)$. Therefore, Step 1 of the function takes $O(km)$ time. Similarly, Step 2 also takes $O(km)$ time since in the worst case $|TP_d|=k$ and each $sp_{dq}\in TP_d$ contains at most m links. Step 3 is repeated at most $|D|$ times, and thus the step requires $O(kmn^2)$ time, while Step 4 uses $O(m)$ at maximum. Step 5 uses $O(n)$ at maximum to check all the nodes. Therefore, function ***Reroute-Demand*()** has a time complexity of $O(kmn^2)$.

4.3.2 Running Time of MSPF-NF

MSPF-NF1 needs to solve the LP in Equation (4.8) up to $O(n)$ times to check for the feasibility of deleting each candidate node and its incident links. Further, the last step of the approach uses a method similar to FGH that runs the LP $O(m^2)$ times to switched off all possible cables from the remaining links. Thus, MSPF-NF1 runs LP solver $O(n+m^2)$ times. For MSPF-NF2, its time complexity analysis is close to MSPF-LF, except including the steps for checking all nodes. The new Step 4a) has time complexity of $O(n^2)$, and new Step 4b) costs $O(kmn^3)$. Since the last step is the same as Step 4 of MSPF-LF, which requires $O(km^2n^2)$, the total complexity of MSPF-NF2 is $O(kn(m+nlogn)+n^2+2km^2n^2+kmn^3)=O(2km^2n^2+kmn^3)=O(km^2n^2+kmn^3)=O(km^2n^2)$, since $m\leq n^2$.

4.4 Evaluation

In this section, we evaluate the performance of both MSPF-LF and MSPF-NF on real and synthetic network topologies and traffic demands under various delay and link utilization constraints. To compare their performance against GreenTE [16], we set each link's bundle size to $w_{ij}=1$. Similarly, we set $\lambda=\infty$, and $U_T=100\%$ to compare our algorithms against FGH [15]. Note that the power savings calculated in both GreenTE and FGH, as defined in [16] and [15] respectively, do not include the savings from turning-off unused nodes, *i.e.*, nodes whose connected links/cables are powered-off. For fair comparisons, we have added a function in both GreenTE and FGH that finds all unused nodes and calculates additional power saving from switching-off nodes.

4.4.1 Experiment Setup

We investigate the average power saving when using MSPF over different network topologies and traffic matrices described in Section 2.5. As shown in Table 2.2 and Table 2.5, we consider four real topologies, *i.e.*, Abilene [99], GÉANT [100] and two topologies from Rocketfuel [101], Sprint and AT&T, and three synthetic topologies, *i.e.*, TS8_56, TS23_161 and TS40_280. For each real topology, we consider link (i, j) with bundle size w_{ij} ranging from 1 to 10 and the MLU $U_T\leq 50\%$. For each synthetic topology, we use a larger range of bundle sizes, with w_{ij} ranging from 1 to 30, but with the same MLU $U_T\leq 50\%$; the length of any routing path must be no longer than the diameter of the network. Our simulations were performed on a Linux PC with 3.07GHz CPU and 8GB RAM. For GreenTE and FGH, we ran the source code provided by their respective authors, and the CPLEX [97] LP solver. We set k=100 candidate shortest paths for both GreenTE and MSPF.

4.4.2 Power Saving from Cables Only

The experiments in this section use topologies in Table 2.2, each of which contains only access nodes that cannot be switched off [16]. Therefore, we use MSPF-LF, shown in Fig. 4.1, to only switch off cables. As we use the same power consumption, p_{ij}, for each cable in each link (i, j), a network's power saving can be calculated using Equation (2.1).

Let M_{ND} and M_{λ} represent the power savings generated by MSPF when the delay constraints in (C2.1) and (C.2.2) are set to $L_d \leq$ND and $L_d \leq l_d\lambda$, respectively. Let M_{∞} denote the power saving when the delay constraint is set to infinity; thus $M_{\infty} \geq M_{ND}$ and $M_{\infty} \geq M_{\lambda}$. For each network, we used the LP solution in [13] to find the minimum delay multiplier λ that yields a feasible solution; Abilene, GÉANT, Sprint and AT&T require a minimum λ of 1.5, 1.4, 1.5 and 1.5 respectively. We used the pre-computed λ in MSPF to produce the power saving of a network. In other words, $M_{1.5}$, $M_{1.4}$, $M_{1.5}$, $M_{1.5}$ shown in Fig. 4.3, 4.4, 4.5 and 4.6 are the lower bound of power saving on the respective network produced by MSPF.

4.4.2.1 Research Network – Abilene and GÉANT

Fig. 4.3 shows the average power savings computed by (4.9) for the Abilene network over the 288 traffic matrices for w_{ij}= 1, 2, …, 10. For w_{ij}=1, M_{ND}=27% is better than $M_{2.0}$=15% because, for each d, there are more paths with $L_d \leq$ND than $L_d \leq 2.0 l_d$; thus MSPF can use more candidate paths for M_{ND} than for $M_{2.0}$. The figure also shows that the average power savings increases sharply when the bundle size is incremented from 1 to 2, and 2 to 3 for both M_{ND} and $M_{2.0}$. Notice that MSPF produces the best power saving M_{ND}=$M_{2.0}$=84% for w_{ij}=10. The result is expected since more cables in one link give the algorithm more flexibility in switching-off cables in the link. Therefore, although link capacity remains the same, increasing its bundle size leads to larger power saving. The power savings M_{∞} and $M_{1.5}$ have the same trend as M_{ND} and $M_{2.0}$ when the bundle size increases from 1 to 10. The figure shows that the power saving when there is no constraint on path length, M_{∞}=46% is significantly larger than $M_{2.0}$=8% for w_{ij}=1, but gradually decreases in significance as w_{ij} increases; eventually the power savings in two scenarios are similar when w_{ij}=10, *i.e.*, at 86% and 84% respectively.

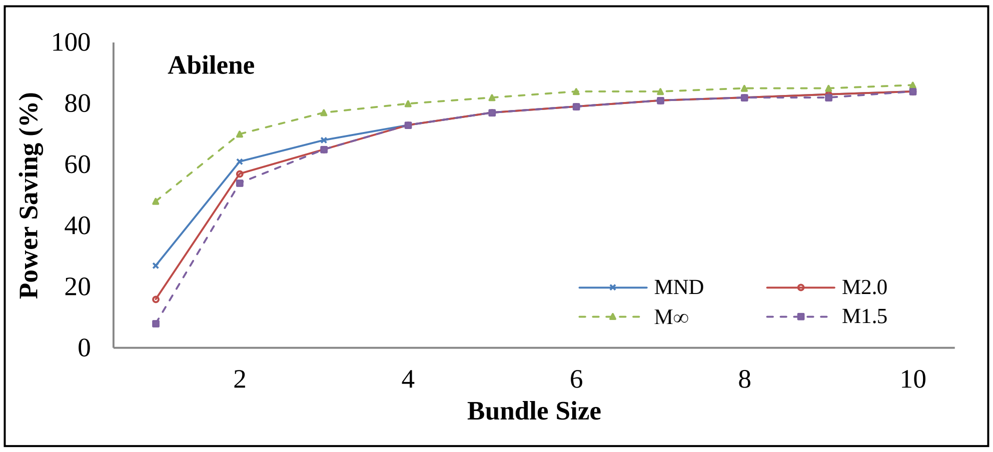


Figure 4.3 Power saving of MSPF on Abilene

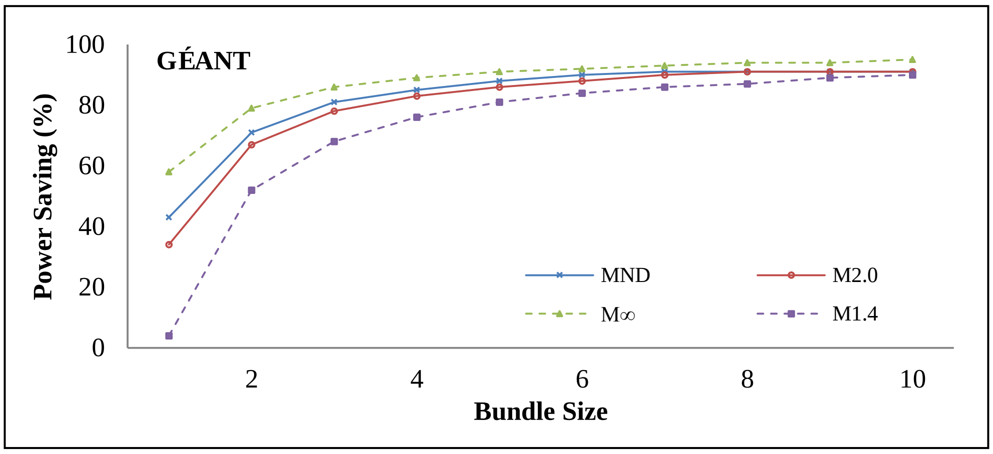


Figure 4.4 Power saving of MSPF on GÉANT

Fig. 4.4 presents the power saving of the GÉANT network averaged over the 96 traffic matrices for w_{ij}= 1, 2, …, 10. For w_{ij}=1, like in the Abilene network, $M_{2.0}$=34% is lower than M_{ND}=43% because the network contains fewer paths that has length $L_d \leq 2.0 l_d$ than $L_d \leq$ND; thus MSPF has a smaller search space on the former than the latter constraint. Notice the significant jump in power saving, *i.e.*, M_{ND}=71% and $M_{2.0}$=67%, when the bundle size is incremented to w_{ij}=2. Both M_{ND} and $M_{2.0}$ reach their peak at 91% when the bundle size is w_{ij}=10. The gap between M_∞ and $M_{1.5}$ is very large; in fact, it exceeds 50% for w_{ij}=1 but less than 5% for w_{ij}=10.

4.4.2.2 Commercial Network – Sprint and AT&T

Fig. 4.5 and Fig. 4.6 show the power savings of Sprint and AT&T for bundle sizes w_{ij}=1 to w_{ij}=10. For Sprint, MSPF considers the first 100 shortest paths to reroute each demand, *i.e.*, *k*=100; we set *k*=20 for AT&T. As shown in Fig. 4.5 and Fig. 4.6, the power savings for Sprint and AT&T increase sharply as we increase the bundle size from 1 to 2; their peak occurs at w_{ij}=10. The explanation for these results is similar to the Abilene and GÉANT, *i.e.*, more cables in one link lead to more feasibility in switching-off cables. For Sprint network, the upper bound M_∞=42% is more than twice that of the lower bound $M_{1.5}$=19%. For the AT&T network, the upper bound is very close to the lower bound, *i.e.*, M_∞=22% versus $M_{1.5}$=19%.

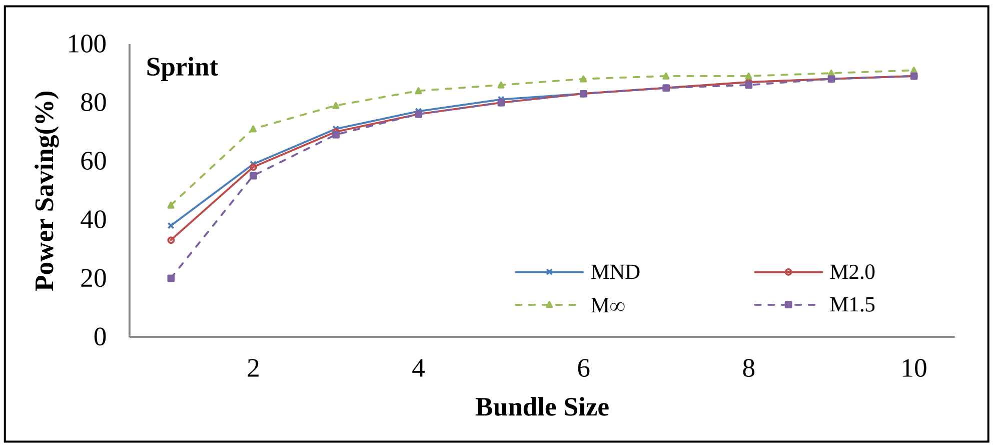


Figure 4.5 Power saving of MSPF on Sprint

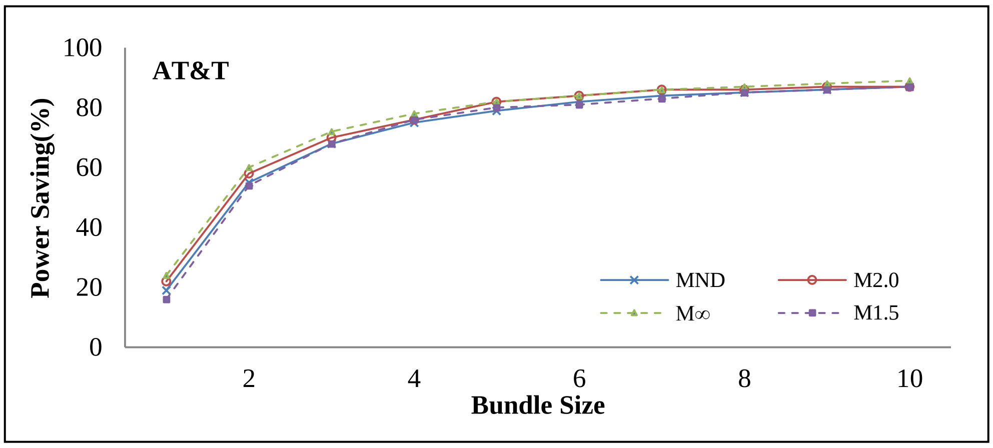


Figure 4.6 Power saving of MSPF on AT&T

4.4.3 MSPF versus FGH and SSPF

FGH aims to turn-off as many cables as possible to maximize power saving by running LP, while SSPF aims to switch off as many cables as possible to maximize power saving while restricting each demand to be routed via a single path with bounded link utilization. As mentioned in Section 4.2, while FGH guarantees sufficient powered on cables to support the given traffic demands, it does not guarantee an upper bound on the delay of each rerouted flow. Further, FGH may increase each link's utilization above a threshold, which consequently may affect a network's resilience against failures during peak hours. To compare the performance of MSPF against FGH in terms of power savings and running time, we consider w_{ij}=3, and set MSPF with U_T=100% and $\lambda=\infty$, *i.e.*, a scenario where there is no upper limit on link utilization nor traffic delay; the results are outlined in Table 4.1. For MSPF against SSPF, we set w_{ij}=3, U_T=50%, with $\lambda=\infty$ for MSPF since SSPF does not support bounded delay; the results are shown in Table 4.2.

Table 4.1 Average Power Saving (%) & Running Time (Seconds)
for U_T=100% and $\lambda=\infty$

	Abilene		**GÉANT**		**Sprint**		**AT&T**	
MSPF	80	0.037	83.3	1.7	82.3	3.5	74.43	10.3
FGH	78.9	1.29	86	63.9	82.1	1184.3	75	5319.7

Table 4.2 Average Power Saving (%) & Running Time (Seconds)
for U_T=50% and $\lambda=\infty$

	Abilene		**GÉANT**		**Sprint**		**AT&T**	
MSPF	73.3	0.035	82.9	1.7	79.2	3.6	74.5	10.4
SSPF	73.3	0.032	85.1	1.2	81.3	2.8	71.3	7.6

From Table 4.1 and 4.2, we see that MSPF runs significantly faster than FGH while producing very competitive power savings on each of the four evaluated network topologies. Our MSPF requires only 0.73%, 2.68%, 0.3%, and 0.35% of the computation time of FGH for the Abilene, GÉANT, Sprint, and AT&T networks respectively. Notice that MSPF produces equivalent or better power savings for Abilene and Sprint than FGH. As compared to SSPF at U_T=50%, MSPF produces better power savings only on the largest topologies, *i.e.*, AT&T, while obtaining slightly worse power savings on GÉANT and Sprint. Further, MSPF also requires slightly higher time complexity as compared to SSPF. However, as discussed in Section 4.4.5 later, by restricting each demand to only a single path, SSPF performs the worst, in terms of link utilization and path length, as compared to MSPF, GreenTE and FGH.

4.4.4 MSPF versus GreenTE

For GreenTE, the authors of [16] assume a hierarchical topology, which is typical of a Wide Area Network (WAN), where all links are bidirectional. This means each pair of directional links connecting nodes *i* and *j*, *i.e.*, link (*i, j*) and (*j, i*), must be turned on or off together. Using this model, GreenTE aims to maximally switch off paired, directional links. Further, their model considers $L_d \leq$ND or $L_d \leq 2.0 l_d$ with w_{ij}=1; *i.e.*, the model does not consider links with bundled cables. For fair comparison, we set the same values of U_T, L_d, and w_{ij} for both GreenTE and MSPF. Let G_{ND} and $G_{2.0}$ represent the power saving produced by GreenTE when its delay constraint is set to the network diameter and twice of the shortest path, respectively. The results are shown in Fig. 4.7 to 4.10.

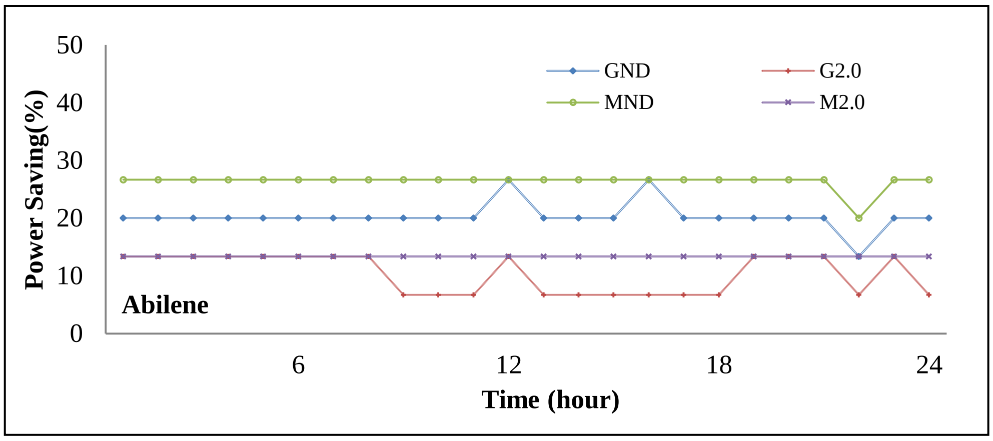


Figure 4.7 Comparison of power saving on Abilene using MSPF and GreenTE

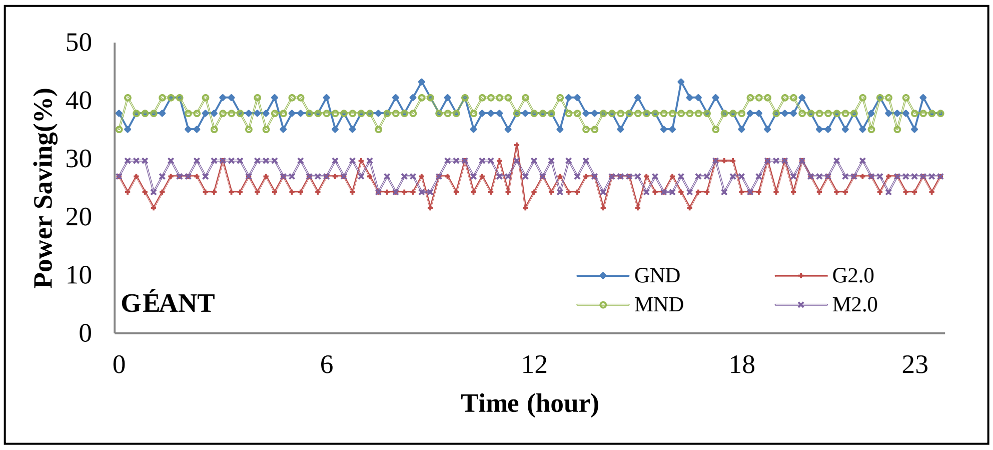


Figure 4.8 Comparison of power saving on GÉANT using MSPF and GreenTE

Fig. 4.7 and 4.8 show a comparison of power saving with U_T≤50% on Abilene and GÉANT over 24 hours. As shown in Fig. 4.7, for the Abilene network, MSPF is able to shut down more cables than GreenTE, resulting in power saving of almost M_{ND}=27%, which is a 7% improvement over GreenTE that has only G_{ND}=20%. For delay constraint (C2.2), MSPF consistently obtained $M_{2.0}$=13.33%, better than GreenTE whose $G_{2.0}$ ranges between 8% and 13.33%. In Fig. 4.8, for GÉANT, the average power saving of running MSPF is always larger than GreenTE with λ=2.0 ($G_{2.0}$≤$M_{2.0}$), *i.e.*, around 25%. In terms of running time, MSPF requires only about 2-3 CPU seconds to produce its results, significantly faster than GreenTE, which required 300 CPU seconds while producing results that incur higher power expenditure.

Fig. 4.9 and 4.10 further compare the performance of MSPF against GreenTE for two large Rocketfuel topologies, Sprint and AT&T, using the traffic matrices generated as described in Section 2.6. As shown in Fig. 4.9, for U_T≤70%, MSPF outperforms GreenTE, on average about 5% in terms of power saving for delay constraint (C2.1); see M_{ND} versus G_{ND}. Similarly, MSPF achieves power saving $M_{2.0}$ on average 3% better than $G_{2.0}$ generated by GreenTE. Notice that GreenTE produces the results for these large topologies in 300 seconds; CPLEX [97], as used in GreenTE, was unable to produce the optimal solution, and therefore, as suggested by the authors [16], we stopped CPLEX after it ran for 300 seconds. In contrast, our MSPF requires only approximately 10 seconds while producing better power savings for Sprint and AT&T's networks. Fig. 4.9 and 4.10 also show that MSPF is more effective in reducing power as compared to GreenTE when MLU increases. Specifically, for Sprint with network diameter constraint, MSPF maintains power saving at 40% as compared to GreenTE whose power saving drops from 36% to only 5% when MLU reaches 70%. A similar situation occurs for AT&T (Figure 4.10) when MLU reaches 45% (see $M_{2.0}$ versus $G_{2.0}$) and 70% (M_{ND} versus G_{ND}). However both algorithms fail to save power for Sprint network with MLU>80%; for AT&T, both fail when MLU>90% for ND and when MLU reaches 100% for the other constraint.

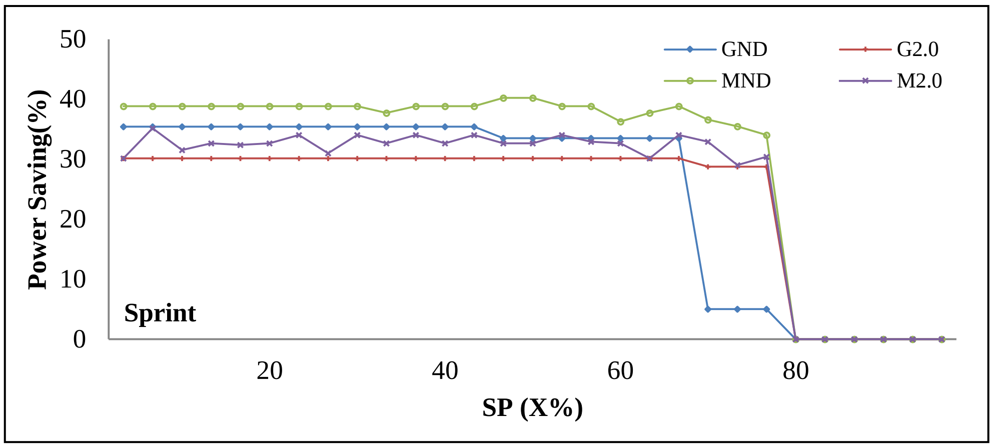


Figure 4.9 Comparison of power saving on Sprint using MSPF and GreenTE

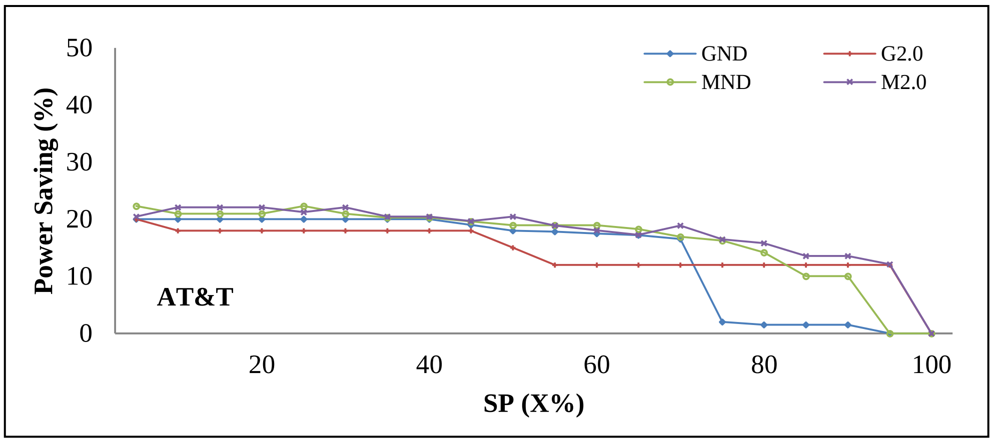


Figure 4.10 Comparison of power saving on AT&T using MSPF and GreenTE

4.4.5 MSPF versus Optimal Solution

This section aims to evaluate the effectiveness of MSPF against the optimal solution (MP-OS) using the ILP in Equation (4.1) to Equation (4.6) for the MP-EAR problem in term of power savings and running times. Unfortunately, due to the exponential time complexity of the problem, as described in Section 3.2, we could not generate the optimal result for the medium and large networks. As an alternative, we use the ILP only on the small network in Fig. 2.1. For this case, we assume each demand d can be routed through its *multiple paths*, when such route exists in its original network, *i.e.*, all links are switched-on. Further, we only consider switching off cables, *i.e.*, all nodes are always on. Finally, for fair comparison with GreenTE, we set U_T=50%, $L_d \leq$ND for each demand d, and w_{ij}=1 for each link (i, j); Table 4.3 shows the results.

Table 4.3 MSPF and GreenTE versus MP-OS on the Network in Fig. 2.1

Mechanism	PS (%)	Time (second)
MP-OS	30	2.43
MSPF	30	0.025
GreenTE	30	0.57

From Table 4.3, we can find that MSPF and GreenTE obtain the same results as MS-OS in term of power savings. However, MSPF performs best on the running time, *i.e.*, 0.025, which is only 4% and 1% of the running time of GreenTE and MP-OS, respectively.

4.4.6 Effects on Link Utilization and Path Length

Intuitively, MSPF, FGH, GreenTE and SSPF would affect the utilization of links as fewer links are used to carry traffic. Since part of the traffic is routed through non-shortest paths, all four algorithms may also increase the routing paths' length as compared to SP routing. In this subsection, we evaluate the impact of the four green routing algorithms on link utilization and path length. In the evaluation, we set U_T=100% and $\lambda=\infty$ in which MSPF, GreenTE and SSPF are expected to produce the largest power savings. Further, FGH can be used only using such settings. In this simulation, we consider GÉANT topology with its real traffic demands, and Sprint topology with synthetic traffic demands as described in Section 2.6. Note that the traffic demands in GÉANT are lighter than in Sprint. Since the results of the other two topologies are similar to these two topologies, they are not shown here.

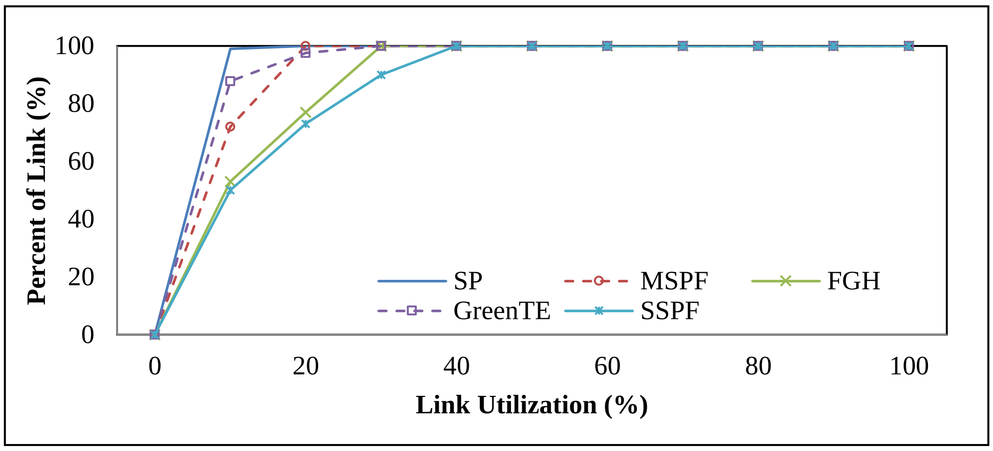


Figure 4.11 CDF of link utilization on GÉANT

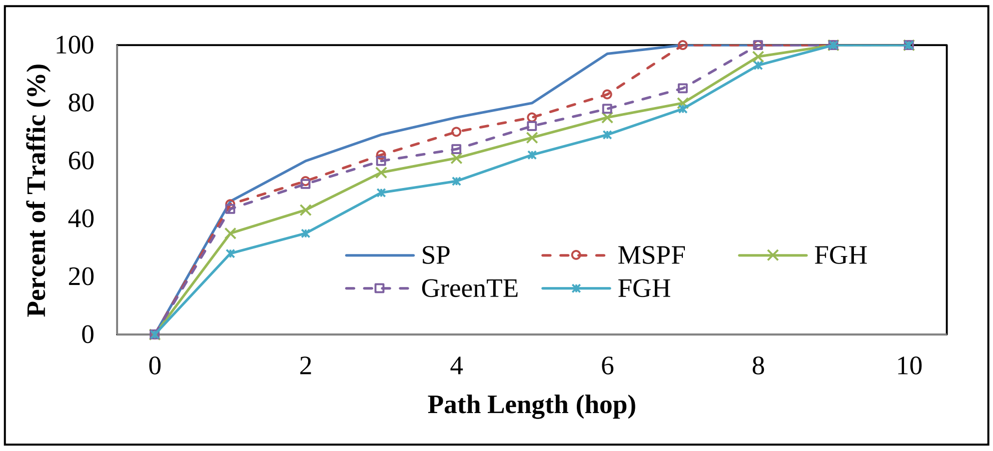


Figure 4.12 CDF of path length on GÉANT

Fig. 4.11 and Fig. 4.12 show the CDF of link utilization and path length on GÉANT using MSPF, FGH, SSPF, GreenTE and SP routing; we use the results obtained using SP routing as the benchmark. As compared to GreenTE, MSPF has fewer links with link utilization below 10% in order to produce better power savings. However, each link's utilization using GreenTE may reach 30%, larger than that using MSPF with a maximum of 20%. The MLU of MSPF is the same as SP routing; however MSPF only has 72% of links with $u_{ij}\leqslant$10% as compared to 99% using SP routing. Among the four algorithms, FGH performs the worst; it only has 53% links with $u_{ij}\leqslant$10%. Among the five algorithms, in term of link utilization, SSPF performs the worst; it has only 50% links with $u_{ij}\leqslant$10% due to its single path routing requirement. In terms of path length, Fig. 4.12 shows that MSPF performs better than FGH and GreenTE because it produces routing with more paths with delay closer to the shortest path. We can see that the maximum path length using either MSPF or SP is 7 hops, versus 8 hops for GreenTE and 9 hops for SSPF and FGH, respectively.

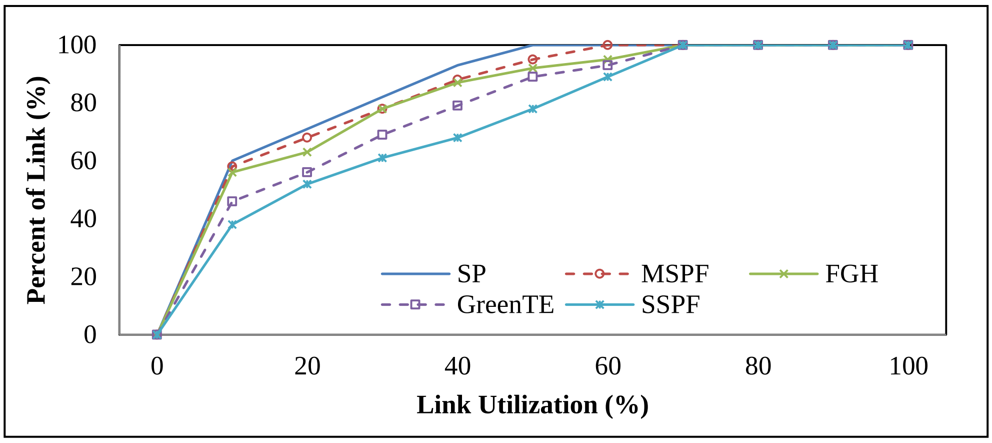


Figure 4.13 CDF of link utilization on Sprint

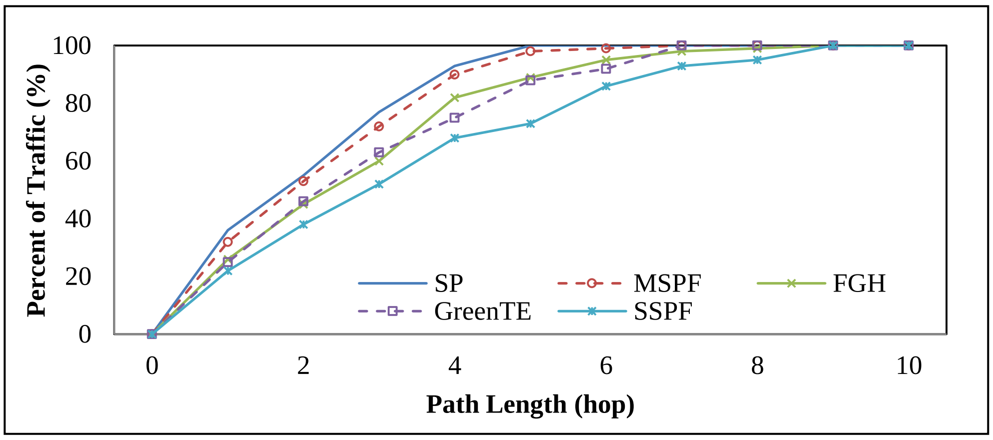


Figure 4.14 CDF of path length on Sprint

In Fig. 4.13 and Fig. 4.14, we plot, respectively, the CDF of link utilization and path length for the Sprint topology. We also use the results obtained using SP routing as the benchmark. Fig. 4.13 shows that each link’s utilization using SP routing is less than 50%. Further, it shows that the utilization of each link using FGH and GreenTE never exceeds 70%, but they are worse than MSPF, which obtained u_{ij}≤60%. Notice that SSPF performs the worst since 32% of the links have utilization larger than 40% while SP, MSPF, FGH and GreenTE only recorded 7%, 12%, 13% and 21% respectively. Fig. 4.14 shows the comparison of path length among the four algorithms. SP achieves the best delay, with path length no longer than five hops, because it always selects the shortest path to route each demand. Among the other three schemes, MSPF and GreenTE are the closest to SP, with a maximum of seven hops, because they also aim to use the shortest path to route each demand, but they allow multiple paths when needed to reduce the total number of powered on links/cables. Using k=100 candidate shortest paths, both MSPF and GreenTE have sufficient number of alternative paths to route each demand.

However, as shown in Fig. 4.14, 12% of the traffics in GreenTE have a path length longer than five hops as compared to only 2% in MSPF, which means more traffic demands are routed through longer paths in GreenTE. This is due to the use of the LP solver, which does not preferentially route traffic over shortest paths. FGH also uses LP to generate its routes, and thus cannot direct flows through possible shortest paths as is done in MSPF. SSPF distributes traffic via single paths, which results in longer paths. Consequently, the route lengths in FGH and SSPF reach up to nine hops, which is longer than that of SP and MSPF.

4.4.7 Power Savings from Routers and Cables

In this subsection, we consider two different node types: access and backbone. We evaluate MSPF_LF, MSPF_NF1 and MSPF_NF2 with three synthetic topologies in Table 2.5 for U_T=50% and $L_d \leq$ND. Similar to [88], we assume each cable in link (i, j) has the same power consumption p_{ij}=0.6kw and each node v consumes the same power p_v=3kw. Thus, the power saving of the network is calculated as Equation (2.2).

Fig. 4.15 and 4.16 show the power saving and the running times required to obtain the solution respectively. Since topology TS8_56 only has 8 backbone nodes out of 56 total nodes, only a few of these nodes can be switched-off. Consequently, switching-off cables is more significant in reducing power than nodes, especially for w_{ij}=30. As a result, given that no more than one node can be switched off, running MSPF-NF1, MSPF-NF2 and MSPF-LF on the TS8_56 produce similar results. Thus, the three algorithms produce almost the same power savings, reaching 80% when

w_{ij}=30; see Fig. 4.15. However, as shown in Fig. 4.16, the running time for MSPF-NF1 is significantly larger as compared to either MSPF-NF2 or MSPF-LF. MSPF-NF2 and MSPF-LF use less than one CPU second while MSPF-NF1 consumes 145 seconds. This is attributed to the LP solver, which it uses to generate routes for traffic whenever it considers switching-off a node/cable.

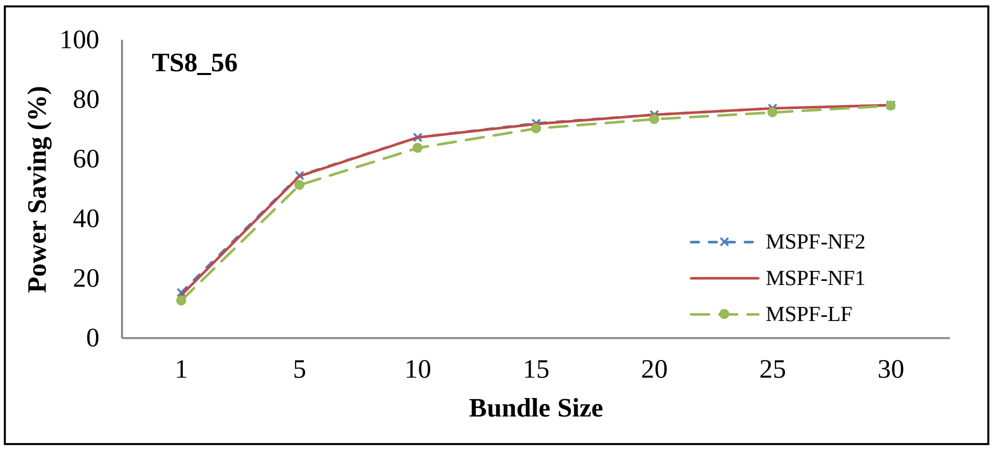


Figure 4.15 Power saving of MSPF on TS8_56

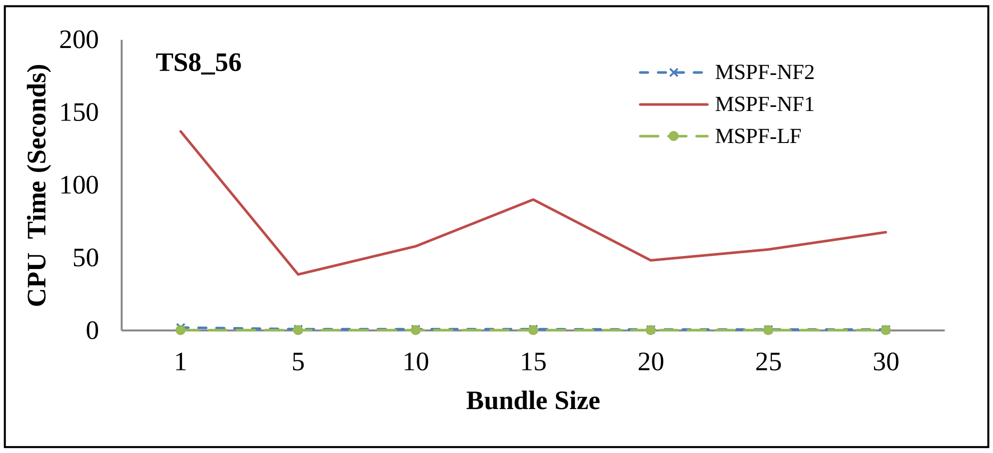


Figure 4.16 Running time of MSPF on TS8_56

Fig. 4.17 and Fig. 4.18 show the power saving and the CPU time, respectively, of MSPF-NF1, MSPF-NF2 and MSPF-LF, on TS23_161 topology that has 23 backbone nodes of 161 total nodes. Since more backbone nodes can be shut down comparing with TS8_56 and both MSPF-NF2 and MSPF-NF1 aim to switch-off all possible nodes first, the node-first approaches perform better than the link-first method, MSPF-LF; see Fig. 4.17. The three algorithms can save up to 80.2% of the power when w_{ij}=30. In terms of running time, as shown in Fig. 4.18, MSPF-NF2 runs almost as fast as MSPF-LF, faster than MSPF-NF1; the first two algorithms use less than one CPU minute while MSPF-NF1 requires up to 400 minutes (6 hours). For the largest topology, TS40_280, MSPF-NF1 could not generate results after running for more than 10 hours, and therefore we use only MSPF-NF2 and MSPF-LF.

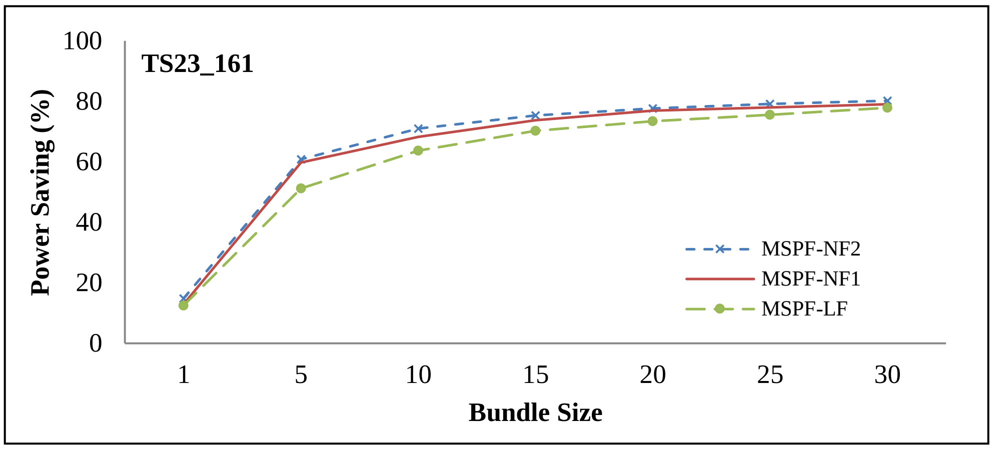


Figure 4.17 Power saving of MSPF on TS23_161

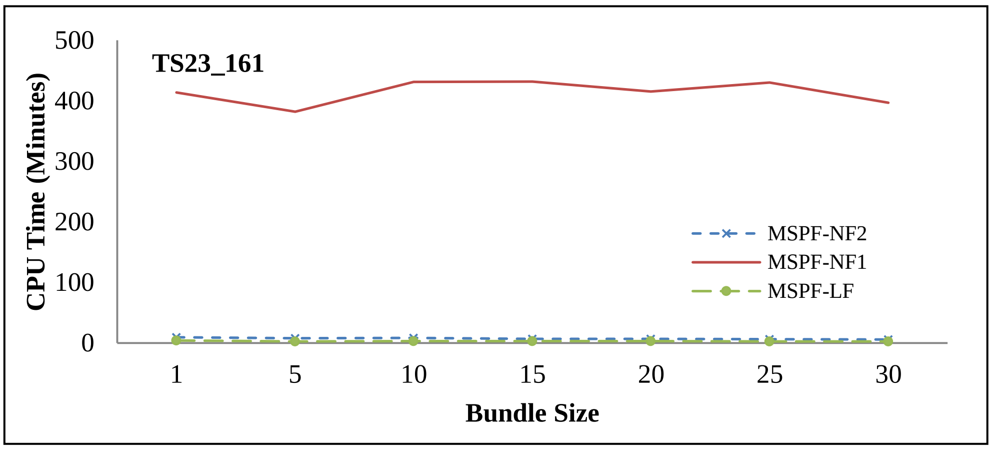


Figure 4.18 Running time of MSPF on TS23_161

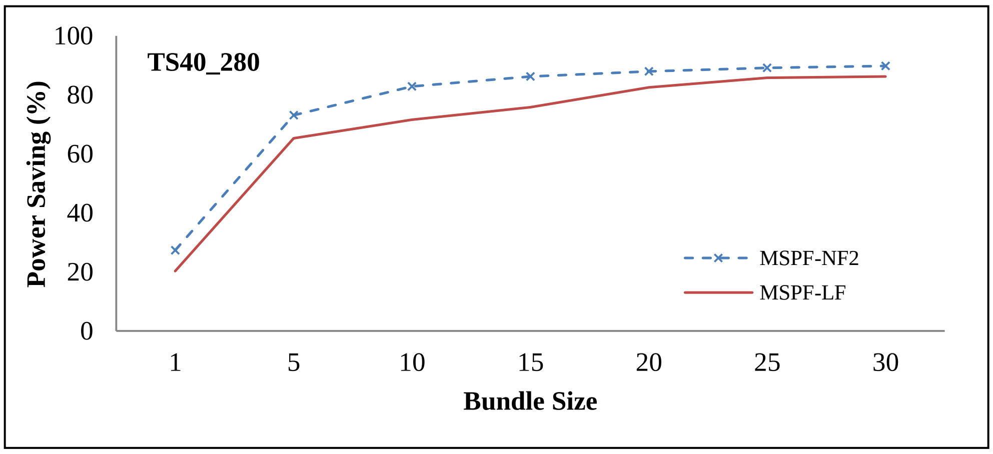


Figure 4.19 Power saving of MSPF on TS40_280

Fig. 4.19 shows that MSPF-NF2 saves more power as compared to MSPF-LF, increasing from 27.33% at w_{ij}=1 to 89.81% at w_{ij}=30 and from 20.35% at w_{ij}=1 to 86.3% at w_{ij}=30 respectively in terms of computation time, both algorithms require about 35 minutes.

4.5 Summary

We have described an optimization problem to reduce the power usage of networks comprising of nodes and links with bundled cables. To reduce power consumption, our NP hard problem aims to maximally switch off unnecessary devices, *i.e.*, nodes and cables, during off-peak periods such that the remaining powered on devices can route the given traffic demands. Further, each demand is only re-routed through one or more paths with length no longer than its given constraint, and each link's utilization does not exceed a given threshold. To this end, we have proposed three efficient and effective heuristic techniques. Through extensive simulations on both real and synthetic network topologies and traffic demands, we have shown their benefits in reducing power consumption, and their superiority against two state-of-the-art techniques, GreenTE [16] and FGH [15]. While our power saving solutions meet user requirements on path length and MLU, shutting down network cables and nodes may reduce reliability, which may affect some critical applications. In chapter 5, we will explore the impact of power saving on network reliability.

Chapter 5

Two-Disjoint-Path Green Routing

In this chapter, we address two versions of an optimization problem, called Energy-Aware Two Disjoint Paths Routing (EAR-2DP). The problem is important in reducing power usage in networks that use either 2DP-L or 2DP-N routing to improve fault tolerance and/or bandwidth/throughput. EAR-2DP aims to maximize powering off nodes and/or links while ensuring (i) the network maintains at least $0 \leq Q_T \leq Q_{max}$ fraction of all possible (s_d, t_d) 2DPs and (ii) MLU is at most $0 \leq U_T \leq 100\%$. Note that Q_{max} is the fraction of the total number of (s_d, t_d) paths that have at least one 2DP.

The first version of EAR-2DP, called EAR-2DP-1, considers switching-off only network links. We formally prove that EAR-2DP-1 is NP-complete, and propose a novel algorithm - Two Disjoint Paths by Shortest Path version 1 (2DP-SP-1) to solve EAR-2DP-1. Our extensive experiments confirm the efficiency of 2DP-SP-1 and its impact on network performance, *i.e.*, reliability and path length. Note that the preliminary version of 2DP-SP-1 has been published in [106]. Further, we have shown in [107] that switching-off redundant links/cables, to save energy, using several state-of-the-art green routing algorithms has significant negative effects on the network's reliability.

The second version of EAR-2DP, called EAR-2DP-2 aims to minimize the power consumption of network resources by switching off both unused nodes and links. We propose an approach, called Two Disjoint Paths by Shortest Path version 2 (2DP-SP-2), to solve EAR-2DP-2. 2DP-SP-2 prioritizes switching off nodes to links since nodes consume an order of magnitude more power [88]. Our extensive simulation results show the advantage of using 2DP-SP-2. Note that the preliminary version of 2DP-SP-2 has been published in [111].

The layout of this chapter is as follows. Section 5.1 shows the problem statement of EAR-2DP-1 and EAR-2DP-2, and presents the proof of NP-completeness for EAR-2DP-1. Section 5.2 describes 2DP-SP-1 to solve the EAR-2DP-1 problem, while Section 5.3 proposes 2DP-SP-2 that switches off both nodes and links to solve EAR-2DP-2 problem. Section 5.4 uses simulation to evaluate the two proposed algorithms, *i.e.*, 2DP-SP-1 and 2DP-SP-2. Finally, Section 5.5 summarizes this chapter.

5.1 Problem Statement

In this section, we describe some notations specific to 2DP routing. Note that more general notations are described in Section 2.1.

5.1.1 Notations

Let R^β be a possible set β that contains $|D|$ number of (s_d, t_d) single or multiple paths with sufficient capacity to route all demands in D, *i.e.*, $R^\beta=\{R_d^\beta \mid R_d^\beta \in SP_d$ and/or $R_d^\beta \in DP_d$ that can be used to route, $\forall d \in D\}$. The set of all possible solutions to route all demands in D is denoted as $R=\{R^\beta \mid \beta=1, 2, \ldots, |R|\}$. Let $TP^\beta \subseteq R^\beta$ be the set of all $R_d^\beta \in R^\beta$ that include at least one $dp_{dl} \in DP_d$ and $\mathrm{M}(R^\beta)$ be the fraction of (s_d, t_d) pairs in R^β that are routed over 2DP, *i.e.*, $\mathrm{M}(R^\beta)=|TP^\beta|/|R^\beta|$. Finally, let $\mathrm{S}(R^\beta)$ be the total number of links used in set R^β and $\mathrm{U}(R^\beta)=\max\{f_{ij}/c_{ij} \mid \forall(i, j) \in R^\beta\}$ be the MLU of R^β.

To illustrate the notations, consider the network in Fig. 2.1. The figure shows one possible set of flow, *e.g.*, β=1, for demands in D using 2DP routing without power saving. Specifically, $R^1=\{R_1^1=\{(0,1)\}, R_2^1=\{(0,1,3), (0,2,3)\}, R_3^1=\{(0,1,3,5), (0,2,4,5)\}$, $R_4^1=\{(2,3)\}$, $R_5^1=\{(2,4)\}$, $R_6^1=\{(2,4)\}$, $R_7^1=\{(3,5), (3,4,5)\}$. Note that demands d=3 and d=7 are the only other demands that can be routed through 2DP, and thus $TP^1=\{R_2^1, R_3^1, R_7^1\}$ and $\mathrm{M}(R^1)=|TP^1|/|R^1|=3/7=0.427$. Without power saving, *i.e.*, all links are switched-on, $\mathrm{S}(R^1)$=10; however, links (4,1) and (1,2) are not used in R^1 and therefore can be switched-off to save energy. Finally, U(R^1)=0.5 since R^1 uses link (4,5) that has the maximum utilization of 0.5. The power saving PS=0 since SP routing without switching off any nodes and links

5.1.2 EAR-2DP-1 Problem

Consider a tuple (G, D, Q_T, U_T), where G (V, E) is a network topology, D is a set of traffic demands, Q_T is a threshold that satisfies $0 \leq Q_T \leq Q_{max}$, and U_T is the MLU threshold. Here Q_{max} satisfies $0 \leq Q_{max} \leq 1.0$ and is the fraction of (s_d, t_d) pairs that have at least one 2DP-L, *i.e.*, $Q_{max}=max(|TP^\beta|)/|R^\beta|=max\{\mathrm{M}(R^\beta)\}$. Note that demands d=6 and d=10 are the only other demands that can be routed through 2DP, and thus Q_{max}=3/10=0.3 in Fig. 2.1. Our Energy Aware Two Disjoint Paths Routing version 1 (EAR-2DP-1) problem is defined as follows.

EAR-2DP-1: Find a set of paths $R^{min} \in R$ that can be used to route all demands in D such that

$$\mathrm{S}(R^{min})=min\{\mathrm{S}(R^1), \mathrm{S}(R^2), \ldots, \mathrm{S}(R^\beta), \ldots, \mathrm{S}(R^{|R|})\}, R^\beta \in R \qquad (5.1)$$

$$\mathrm{M}(R^{min}) \geq Q_T \qquad (5.2)$$

$$\mathrm{U}(R^{min}) \leq U_T \qquad (5.3)$$

Equation (5.1) computes the solution, *i.e.*, to find R^{min} that contains the minimum number of total power-on links. Equation (5.2) states that the ratio of the total number of (s_d, t_d) pairs that use 2DP for routing in R^{min} must be no less than a given threshold Q_T. Equation (5.3) ensures the MLU of links must be no greater than U_T.

5.1.3 NP-Completeness Proof of EAR-2DP-1

To prove the NP-completeness of EAR-2DP-1, we first convert the problem into its equivalent decision problem, EAR-2DP-1'.

Then, we show that the latter problem is NP-complete by reducing from the NP complete SUBSET-SUM problem [112].

EAR-2DP-1': Given (G, D, Q_T, U_T, NL), where NL is a positive integer that denotes the total number of links, is there a set of paths R^β that satisfies the following three conditions?

$$S(R^\beta) = NL \tag{5.4}$$

$$M(R^\beta) \geq Q_T \tag{5.5}$$

$$U(R^\beta) \leq U_T \tag{5.6}$$

Note, functions S(.), M(.) and U(.) in Equation (5.4), (5.5) and (5.6) are the same functions as those used in Equation (5.1), (5.2) and (5.3), respectively. One can obtain the solution for EAR-2DP-1 from that of EAR-2DP-1' in polynomial time using the pseudo code, as shown in Fig. 5.1.

```
EAR-2DP-1 ($G(V, E)$,$D$, $Q_T$, $U_T$)
Begin
    Set $NL$=|$E$|;
    While (EAR-2DP-1' ($G(V,E)$,$D$, $Q_T$, $U_T$, $NL$) == YES)
            $R^\beta$ = EAR-2DP-1' ($G(V,E)$,$D$, $Q_T$, $U_T$, $NL$);
            $NL = NL - 1$;
    End-While
    Return $R^\beta$
End
```

Figure 5.1 Algorithm to derive EAR-2DP-1 from EAR-2DP-1'

Starting with $NL=|E|$, the solution to EAR-2DP-1, *i.e.*, $R^{min}=R^\beta$, is obtained by solving EAR-2DP-1' repeatedly, for at most $O(|E|)$ times; thus the conversion can be performed in polynomial time. Therefore, EAR-2DP-1 is at least as hard as EAR-2DP-1', *i.e.*, EAR-2DP-1$\leq_p$EAR-2DP-1'. On the other hand, if we can find the optimal solution for EAR-2DP-1, *i.e.*, we obtain the minimum number of edges $S(R^{min})$ in Equation (5.1), we can produce the solution for EAR-2DP-1'; we consider two possible cases: (i) $NL \geq S(R^{min})$, and (ii) $NL < S(R^{min})$. For case (i), when $NL=S(R^{min})$, EAR-2DP-1' returns "YES". On the other hand, although $NL>S(R^{min})$ does not meet the condition in Equation (5.4), one can always include ΔS additional links $(i, j)\in E$ and $(i, j)\notin R^{min}$ such that $NL=S(R^{min})+\Delta S$. Note that the additional links do not affect traffic distribution in EAR-2DP-1. Thus, for case (i), EAR-2DP-1' returns "YES". For case (ii), EAR-2DP-1' will output "NO" since S(R^{min}) is the minimum number of edges for the constraint in Equation (5.4); one cannot delete ΔS links from R^{min} to make NL=S(R^{min})-ΔS while satisfying Equation (5.2), and thus Equation (5.5). Therefore, if EAR-2DP-1' is NP-complete, EAR-2DP-1 is also NP-complete. Specifically, one can show the NP-completeness of EAR-2DP-1 by proving that EAR-2DP-1' is NP-complete. We first show that EAR-2DP-1' is in NP. Suppose we are given a graph $G(V,E)$, Q_T, U_T and NL. The certificate we choose is the routing $R^{\beta}\in R$. The verification algorithm affirms that M(R^{β})$\geq Q_T$ and U(R^{β})$\geq U_T$, and checks if the total number of edges that are used in R^{β}, *i.e.*, S(R^{β}), is equal to NL. This verification can be performed in polynomial time.

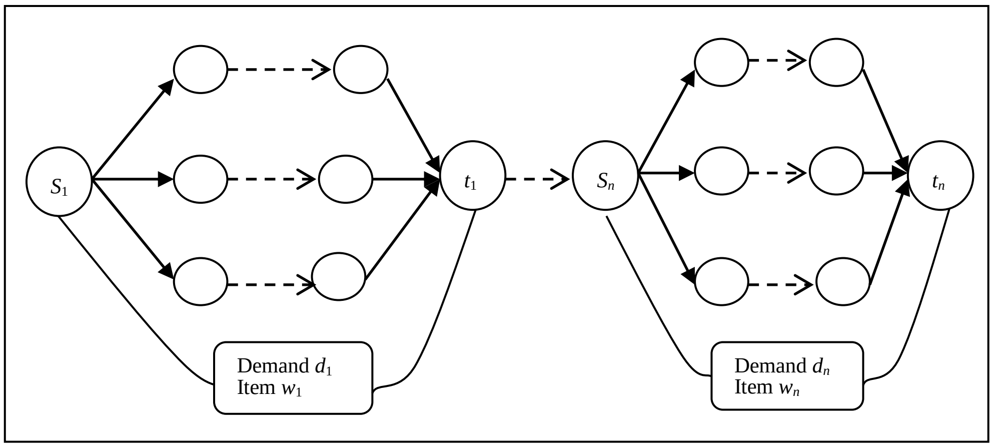


Figure 5.2 An instance of EAR-2DP-1'

We prove that EAR-2DP-1' is NP-complete by showing that SUBSET-SUM$\leq_p$ EAR-2DP-1'. Given a set of n integers W={w_1,w_2,…,w_n} and a positive integer t, the SUBSET-SUM, an NP-complete problem [112], is stated as follows:

$$\text{SUBSET-SUM}=\{<W, t>\text{: Is there a subset } W'\subseteq W \text{ such that } t=\sum_{w_i\in W'} w_i \}.$$

For the proof, we first construct an instance of EAR-2DP-1' which, in turn, is used to reduce to the SUBSET-SUM problem. Then, we provide two lemmas to show that EAR-2DP-1'$\leq_p$SUBSET-SUM and SUBSET-SUM$\leq_p$EAR-2DP-1'.

Fig. 5.2 shows an instance of EAR-2DP-1' for a topology G (V, E). In this instance, the graph G is separated into n sub-graph G_d (V_d, E_d), for each demand d and (s_d, t_d) node pair, for d=1, …, n. In this instance, each G_d is assumed to have sufficient resource to route the flow of each demand d, U_T=1.0 and Q_T=Q_{max}=1.0 (*i.e.*, there are at least 2DP between each (s_d, t_d). Let E_L be the set of links connecting all adjacent sub-graphs G_d and G_{d+1}. We thus have $E=E_1\cup E_2\cup\ldots\cup E_n\cup E_L$ and $E_1\cap E_2\cap\ldots\cap E_n\cap E_L=\varnothing$, and $V=V_1\cup V_2\cup\ldots\cup V_n$. Each $dp_{dl}=\{sp_{dx}, sp_{dy}\}\in DP_d$, for d=1, …, n, has $L(sp_{dx})+L(sp_{dy})$ edges. We set $y_d=L(sp_{dx})+L(sp_{dy})$ if dp_{dl} is used to route demand d; let $Y=\{y_1, y_2, \ldots, y_n\}$ be a set whose elements indicate the number of edges used in each sub-graph G_d (V_d, E_d), for d=1, 2, …, n. Finally, we set $NL=\sum_{d=1}^{n} y_d$.

We reduce SUBSET-SUM into EAR-2DP-1' as follows.

$$w_1 = y_1 \tag{5.7}$$

$$w_d = y_d + y_{d-1},\ (d>1) \tag{5.8}$$

$$NL=\sum_{d=1}^{n} y_d \tag{5.9}$$

Lemma 1. If the solution of the SUBSET-SUM problem exists, then the solution of EAR-2DP-1' instance also exists.

Proof. If the solution to the SUBSET-SUM problem exists, a subset $W'\subseteq W$ is derived from two possible options:

1) When n is odd*:*

$$W' = \{w_1, w_3, \ldots, w_n\}$$

$$\sum_{d=1}^{(n+1)/2} w_{2d-1} = \sum_{d=1}^{n} y_d = NL$$

2) When n is even*:*

$$W' = \{w_2, w_4, \ldots, w_n\}$$

$$\sum_{d=1}^{n/2} w_{2d} = \sum_{d=1}^{n} y_d = NL$$

Consequently, the solution of the EAR-2DP-1' instance exists.

Q.E.D.

Lemma 2. If the solution of the EAR-2DP-1' instance exists, then the solution of SUBSET-SUM problem also exists.

Proof. If we can find the solution $Y=\{y_1, \ldots, y_n\}$ for an EAR-2DP-1' instance, *i.e.*, given an integer number NL, according to Equation (5.9),

$$NL = \sum_{d=1}^{n} y_d \tag{5.10}$$

From Equation (5.7) and (5.8), we get

$$y_1 = w_1 \tag{5.11}$$

$$y_d = w_d - y_{d-1} \ (d>1) \tag{5.12}$$

Using Equation (5.10), (5.11) and (5.12), we have

1) When n is odd*:*

$$\exists W' = \{w_1, w_3, \ldots, w_n\} \subseteq W$$

$$NL = \sum_{d=1}^{n} y_d = \sum_{d=1}^{(n+1)/2} w_{2d-1}$$

2) When n is even:

$$\exists W' = \{w_2, w_4, \ldots, w_n\} \subseteq W$$

$$NL = \sum_{d=1}^{n} y_d = \sum_{d=1}^{n/2} w_{2d}$$

Thus, the solution of SUBSET-SUM exists.

Q.E.D.

***Theorem* 1**. EAR-2DP-L problem is NP-complete.

Proof. *Lemma* 1 and 2 show that EAR-2DP-1’$\leq_p$SUBSET-SUM and SUBSET-SUM$\leq_p$EAR-2DP-1’, respectively. Since EAR-TDP-1$\leq_p$EAR-2DP-1’, as shown in Fig. 5.1, EAR-2DP-1 is also NP-complete.

Q.E.D.

5.1.4 EAR-2DP-2 Problem

The EAR-2DP-1 is extended to EAR-2DP version 2 (EAR-2DP-2) problem that switch-off both nodes and links to further minimize the power consumption. EAR-2DP-2 is defined as follows.

EAR-2DP-2: Find a set $R_{min} \in R$ that can be used to route all demands in D such that

$$\mathrm{S}(R_{min}) = min\{p_v \mathrm{V}(R^{\beta}) + p_{ij} \mathrm{E}(R^{\beta}) | R^{\beta} \in R\} \tag{5.13}$$

$$\mathrm{M}(R_{min}) \geq Q_T \tag{5.14}$$

$$\mathrm{U}(R_{min}) \leq U_T \tag{5.15}$$

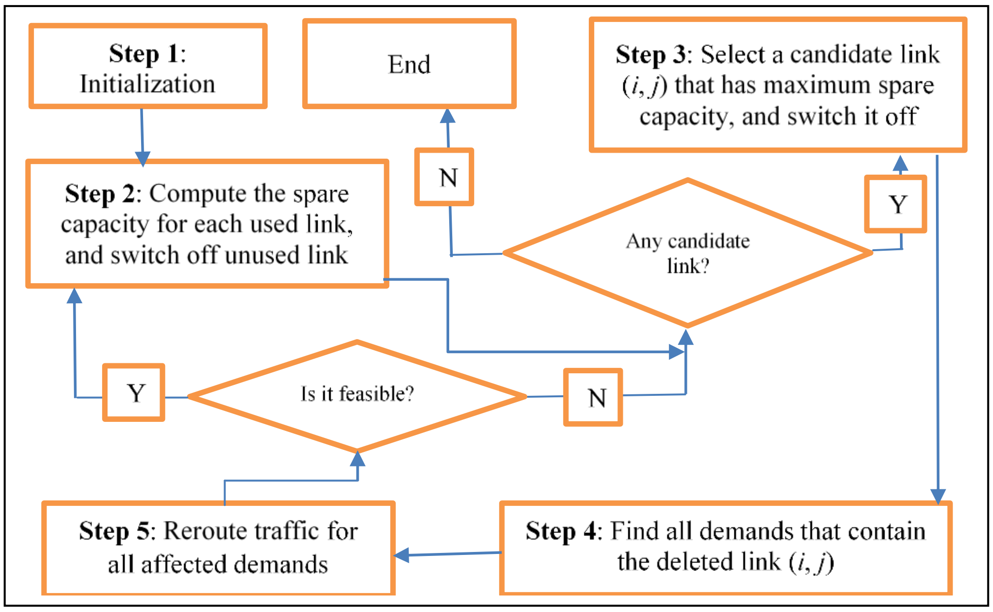
Note that $\mathrm{V}(R^{\beta})$ and $\mathrm{E}(R^{\beta})$ are the number of power-on nodes and links in each $R^{\beta} \in R$ respectively. The term $\mathrm{S}(R_{min})$ represents the power consumption of all nodes and links in R_{min} subject to constraints (5.14) and (5.15). Constraint (5.14) requires there to be at least Q_T fraction of routes in R_{min} that include at least one 2DP-L, while constraint (5.15) ensures the link utilization of each link used in R_{min} be no larger than U_T. The EAR-2DP-2 problem is a variant of the multi-constrained path (MCP) problem since it aims to generate an optimal set of feasible routes R_{min} subject to two constraints, *i.e.*, (5.14) and (5.15). Since MCP with more than one constraint is known to be NP-complete [12], we conclude that EAR-2DP-2 is NP-complete.

5.2 Switching off Links Only

In Section 5.2.1, we present an overview of our approach, called Two Disjoint Paths by Shortest Path version 1 (2DP-SP-1), to heuristically solve the EAR-2DP-1 problem. Then, in Section 5.2.2, we describe 2DP-SP-1 for 2DP-L. We later show

how to use the algorithm for 2DP-N in Section 5.2.3, and analyze the time complexity of 2DP-SP-1 in Section 5.2.4. The latter section also describes an alternative algorithm for 2DP-SP-1, called 2DP-SP-A, to reduce the time complexity of 2DP-SP-1.

5.2.1 2DP-SP-1 Algorithm



Step 1: Initialization
End
Step 3: Select a candidate link (i, j) that has maximum spare capacity, and switch it off
Step 2: Compute the spare capacity for each used link, and switch off unused link
N
Any candidate link?
Y
Y
Is it feasible?
N
Step 5: Reroute traffic for all affected demands
Step 4: Find all demands that contain the deleted link (i, j)

Figure 5.3 A flow chart for the 2DP-SP-1 algorithm

As shown in Fig. 5.3, 2DP-SP-1 algorithm is composed of five main steps. For each demand *d*, Step 1 comprises of two stages: i) Generate *k* shortest paths, *i.e.*, SP_d, and generate all possible 2DPs, and ii) Route its flow via paths in DP_d and/or SP_d. If demand *d* has at least one 2DP (*i.e.*, $|DP_d|>0$), its flow is routed via any one 2DP$\in DP_d$; otherwise if there is remaining flow or if there is no 2DP for the demand, route the remaining flow via one or more paths in SP_d. Note that the routing in stage ii) is always feasible since the original network (before switching-off links for power saving) has sufficient capacity; Q_{max} is computed after the routing. In Step 2, 2DP-SP-1 updates each link's current flow and remaining capacity. Each link with f_{ij}=0, *i.e.*, unused link, is switched off and deleted from E_r that contains all switched-on links. Further, each link in E_r is set as a 'candidate' link to be switched off. Next, in Step 3, the algorithm selects one candidate link $(i, j)\in E_r$ that has the maximum r_{ij}, and switches it off. In Step 4 and 5, 2DP-SP inspects if all demands can be routed when the link is deleted.

```
2DP-SP-1($G(V,E)$, $D$, $Q_T$, $U_T$)
Begin
        $E_r = E$;
/*Step 1: Initialization */
        For each $d \in D$ do
            (i)  Generate $KSP_d$ in $G(V, E)$;
                 Call Find-2DP($G(V, E)$, $d$) to generate $DP_d$;
            (ii)  Call 2DP-Routing($G(V, E)$, $d$)
              End-For
/*Step 2: Link update*/
        For each $(i, j) \in E_r$  do
            If $f_{ij}$ == 0 then
                $E_r = E_r - \{(i, j)\}$; // switched off each unused link
            Else
                $r_{ij} = U_T * c_{ij} - f_{ij}$; // update remaining capacity of $(i, j)$
        End-For
              $E'_r = E_r$; //a set of candidate links to be switched-off
/*Step 3: Choose a candidate link */
        Select a link $(i, j) \in E'_r$ that has the maximum $r_{ij}$
        $E_r = E_r - \{(i, j)\}$; // switch off the candidate link
        $E'_r = E'_r - \{(i, j)\}$;   //update the remaining candidate links
/*Step 4: Find a set of affected demands D' */
        $D' = \varnothing$;
        For each route $R_d^\beta$ that contains the deleted link $(i, j)$ do
                $D' = D' + \{d\}$;
        End-For
/*Step 5: Reroute each affected demand */
        isFeasible = true;
        For each $d \in D'$ do   //check all affected demands
            If (2DP-Routing($G(V, E_r)$, $d$) == false) then
                isFeasible = false; $E_r = E_r + \{(i, j)\}$; // switch on the candidate link
                If $|E'_r|$ >0 then   Go to Step 3;
                Else   Return ($E - E_r$); //returns a set of switched-off links
        End-For
        If isFeasible == true then //all demands can be routed when (i,j) is off
            Go to Step 2;
End
```

Figure 5.4 2DP-SP-1 algorithm for link-disjoint paths

Specifically, in Step 4, the algorithm finds all demands, called affected demands, whose routes are disconnected when the candidate link is off; *i.e.*, the route contains the link. If any of the affected demands fails to be rerouted in Step 5, the candidate link must be switched-on, *i.e.*, put it back to E_r, and the algorithm repeats from Step 3 when there is candidate link; otherwise, the algorithm repeats from Step 2. The steps are repeated until there is no link to be switched-off, and 2DP-SP returns a topology that comprises all switched-on links in E_r; the total number of switched-off links, *i.e.*, $|E|$-$|E_r|$, gives the power saving. The following Section 5.2.2 describes the details of the five steps.

```
Find-2DP($G(V, E_r)$, $d$)
Begin
    $l$ = 1;
    For each $sp_{dx} \in KSP_d$ do
        Generate $G_1(V, E_1)$ from $G(V, E_r)$ by deleting all links in $sp_{dx}$;
        Generate $k$ paths for ($s_d$, $t_d$) from $G_1$, and store in $KSP'_d$;
        For $sp_{dy} \in KSP'_d$ do
            $dp_{dl}=\{sp_{dx}, sp_{dy}\}$;
            If $dp_{dl} \notin DP_d$ then
                Store ($sp_{dx}$, $sp_{dy}$) into $DP_d$;
                $l$++;
        End-For //for $sp_{dy}$
    End-For //for $sp_{dx}$
    Reorder $dp_{dl}$ in $DP_d$ in ascending order of path length;
    Return $DP_d$.
End
```

Figure 5.5 Function Find-2DP()

5.2.2 2DP-SP-1 for Link Disjoint Paths

As shown in Fig. 5.4, 2DP-SP-1 for 2DP-L has five main steps. Step 1 first generates a set of single paths and 2DPs for each demand $d \in D$, *i.e.*, SP_d and DP_d. Then, it distributes the traffic of each d through the paths in DP_d and/or SP_d, and calculates Q_{max}. The next four steps (Step 2 ~ Step 5) aim to maximally switch off links, to maximize power saving; however, the remaining links in set E_r should have sufficient capacity to route all traffic demands while satisfying the threshold $0 \leq Q_T \leq Q_{max}$ and

MLU constraints; initially E_r is set to E, and 2DP-SP-1 produces all switched-off links from E-E_r.

```
2DP-Routing(G(V, E_r), d)
Begin
    count = 0;
    If count<Q_T*|D| then
        If |DP_d|>0 then
                If Use-2DP (d)==true then
              count++;
            Else If Use-Non-2DP(d, b_d)==false then
              Return false;
        Else //if |DP_d|==0
            If Use-Non-2DP(d, b_d)==false then
              Return false;
    Else //if count> Q_T*|D|
        If Use-Non-2DP(d, b_d)==false then
            Return false;
End
```

Figure 5.6 Function 2DP-Routing()

Step l first uses *Yen*'s algorithm [30] to generate $k \geq 1$ shortest paths, $KSP_d=\{sp_{d1}, sp_{d2}, \ldots, sp_{dk}\} \subseteq SP_d$, for each demand d. For Fig. 2.1, Step 1 generates seven (s_d, t_d) path sets, *i.e.*, $KSP_1=\{sp_{11}=(0,1)\}$, $KSP_2=\{sp_{21}=(0,1,3)$, $sp_{22}=(0,2,3)$, $sp_{23}=(0,1,2,3)\}$, $KSP_3=\{sp_{31}=(0,1,3,5)$, $sp_{32}=(0,2,3,5)$, $sp_{33}=(0,2,4,5)$, $sp_{34}=(0,1,2,3,5)$, $sp_{35}=(0,1,3,4,5)$, $sp_{36}=(0,2,3,4,5)\}$, $KSP_4=\{sp_{41}=(2,3)\}$, $KSP_5=\{sp_{51}=(2,4)$, $sp_{52}=(2,3,4)\}$, $KSP_6=\{sp_{61}=(4,5)\}$, $KSP_7=\{sp_{71}=(3,5)$, $sp_{72}= (3,4,5)\}$. Then, the step uses function **Find-2DP**(), shown in Fig. 5.5, to generate a set of all possible disjoint paths $dp_{dl}=\{sp_{dx}, sp_{dy}\}$ for all sp_{dx}, $sp_{dy} \in KSP_d$. Specifically, for each demand d, function **Find-2DP**() generates a graph $G_1(V, E_1)$ by deleting all links in each $sp_{dx} \in KSP_d$ from G. Then, it uses *Yen*'s algorithm to generate k-shortest paths from $G_1(V, E_1)$, and stores the paths in the set KSP'_d. Finally, it generates $dp_{dl}=\{sp_{dx}, sp_{dy}\}$ for each path $sp_{dy} \in KSP'_d$ that has no common links with sp_{dx}, and stores the pair in set DP_d in increasing path length, *i.e.*, $\max\{L(sp_{dx}), L(sp_{dy})\}$, order. For the example in Fig. 2.1, the function generates $DP_2=\{dp_{21}=\{sp_{21},sp_{22}\}\}$, $DP_3=\{dp_{31}=\{sp_{31},sp_{33}\}$, $dp_{32}=\{sp_{31},sp_{36}\}$, $dp_{33}=\{sp_{32},sp_{35}\}\}$, and $DP_7=\{dp_{71}=\{sp_{71},sp_{72}\}\}$. Finally, Step 1 uses function **2DP-Routing**(), shown in Fig. 5.6, to distribute the traffic of each demand *d* via paths in DP_d and/or KSP_d.

```
Use-2DP (d)
Begin
    // Even flow distribution; Assume $B(sp_{dx}) \le B(sp_{dy})$
    For each $dp_{dl}=\{sp_{dx},sp_{dy}\} \in DP_d$ that does not contain switched-off links in $E-E_r$ do
        If $b_d/2 \le B(sp_{dx})$ then
            // Route flow $b_d/2$ through $sp_{dy}$ and $sp_d$
              Increase $f_{ij}$ of each link $(i, j)$ in $sp_{dx}$ and $sp_{dy}$ by $b_d/2$;
              Insert $sp_{dy}$ and $sp_{dx}$ in $R_d^\beta$;
               Return true;
    End for
//Uneven flow distribution; distribute $b_d$ via $dp_{dl}=\{sp_{dx}, sp_{dy}\}$ that does not contain
//switched-off links in $E-E_r$
    If $B(sp_{dy}) \ge b_d - B(sp_{dx})$ then // $b_d \le B(sp_{dx})+B(sp_{dy})$
        Increase $f_{ij}$ of each link $(i, j)$ in $sp_{dx}$ by $B(sp_{dx})$;
        Increase $f_{ij}$ of each link $(i, j)$ in $sp_{dy}$ by $b_d - B(sp_{dx})$;
        Insert $sp_{dy}$ and $sp_{dx}$ in $R_d^\beta$;
        Return true;
    Else // $b_d > B(sp_{dx})+B(sp_{dy})$
        //Route the remaining flow via one or more paths in $KSP_d$
        If Use-Non-2DP($d$, $b_d - (B(sp_{dx})+B(sp_{dy}))$)== false then
            Return false
        Else // Use-Non-2DP() returns true
            Increase $f_{ij}$ of each link $(i, j)$ in $sp_{dx}$ by $B(sp_{dx})$;
            Increase $f_{ij}$ of each link $(i, j)$ in $sp_{dy}$ by $B(sp_{dy})$;
            Insert $sp_{dy}$ and $sp_{dx}$ in $R_d^\beta$;
            Return true;
End
```

Figure 5.7 Function Use-2DP()

The **2DP-Routing**(), when $| DP_d |>0$, uses function **Use-2DP**(), shown in Fig. 5.7, to distribute the traffic of demand *d* through its 2DP; otherwise, it calls function **Use-Non-2DP**(), shown in Fig. 5.8, to distribute the traffic through one or more paths in its KSP_d starting from the shortest path. Any generated routing path for each demand *d* is stored in R_d^β when the functions return true.

```
Use-Non-2DP(d, b)
Begin
    $Temp\_R = \varnothing$;
        //Distribute traffic with multiple non-disjoint paths routing in ascending order
    For each $sp_{dq} \in KSP_d$ that does not contain switched-off links in $E$-$E_r$ do
        If $b \leq B(sp_{dq})$ then
            Insert $sp_{dq}$ in $Temp\_R$;
            Increase $f_{ij}$ of each link $(i, j) \in sp_{dq}$ in $Temp\_R$ by $b$;
            Insert each path $sp_{dq}$ in $Temp\_R$ into $R_d^{\beta}$;
            Return true;
        Else
            //Route $B(sp_{dq})$ flow of the traffic through $sp_{dq}$;
            Insert $sp_{dq}$ in $Temp\_R$;
            $b = b - B(sp_{dq})$;
    End-For
    Return false; // If flow $b$ cannot be routed through all paths in $KSP_d$
End
```

Figure 5.8 Function Use-Non-2DP()

For each demand d, function **Use-2DP()** aims to route b_d evenly through any $dp_{dl}=\{sp_{dx}, sp_{dy}\}$, *i.e.*, $b_d/2$ in each path as used in ECMP [52], prioritizing the shorter 2DP; we assume $B(sp_{dx}) \leq B(sp_{dy})$. However, if the even flow distribution is not feasible, *i.e.*, $B(sp_{dx}) < b_d/2$, the function distributes the flow via the shortest 2DP, *i.e.*, $dp_{d1}=\{sp_{dx}, sp_{dy}\}$. For this case, it allocates flow of size $B(sp_{dx})$ to sp_{dx} while the remaining flow, *i.e.*, b_d-$B(sp_{dx})$, is routed through sp_{dy}, if feasible. However, if sp_{dy} also has insufficient bandwidth, *e.g.*, $B(sp_{dy}) < b_d$-$B(sp_{dx})$, it calls function **Use-Non-2DP()** to route the remaining flow, *i.e.*, b_d-$(B(sp_{dy})+B(sp_{dx}))$, through one or more paths in its KSP_d. Note that **Use-2DP()** returns false when **Use-Non-2DP()** returns false, *i.e.*, it fails to route the remaining flows through the paths in KSP_d; for this case **2DP-Routing**() will use function **Use-Non-2DP()** to route b_d. As an example in Fig. 2.1, for demand 3, *i.e.*, (0,3,3.0), $B(sp_{31})=3.5>1.5=b_d/2$ and $B(sp_{33})=5.0>1.5$ since $r_{01}=3.5$ and $r_{02}=r_{23}=5.0$, thus sp_{31} and sp_{32} have enough capacity to route b_d evenly. However, for (3,5,6.0) with $dp_{71}=\{sp_{71}, sp_{72}\}$ and $b_d/2=3.0>B(sp_{72})=2.5$,

Use-2DP() allocates flow of size 2.5 to sp_{72}, and routes the remaining flow b_d-2.5=3.5 through path sp_{71} since B(sp_{71})=3.5; thus the flow is allocated unevenly to dp_{71}.

For each demand d with $|DP_d|=0$, **2DP-Routing**() uses function **Use-Non-2DP()** to route b_d via one or more paths starting from the shortest path in KSP_d. If the first path does not have sufficient capacity, the function will route the remaining flow through the next available shortest path. The step is repeated until b_d is completely routed, in which case the function returns true; else it returns false, in which case **2DP-Routing**() returns false since it fails to route all demands in D. When function **2DP-Routing**() returns true, *i.e.*, the traffic volume b_d has been successfully distributed, it updates R_d^{β} and the total flows on each link $(i, j)\in E_r$, *i.e.*, f_{ij}. When the function returns false, it will maintain the previous routing $R^{\beta\text{-}1}$. Notice that, for Step 1 in 2DP-SP-1 algorithm, we assume the network contains sufficient bandwidth to carry the traffic demands, and thus the function never returns false, *i.e.*, R^1 always exists. Further, for the step, we set Q_T=1.0 so that the function routes each demand through its 2DP whenever possible, and thus Q_{max}, the maximum ratio of the number of demands with $|DP_d|>0$ over the total number of demands, is set to $Q_{max}=count/|D|$, where *count* is the number of traffic demands that use 2DP routing.

From Step 2 to 5, 2DP-SP-1 uses the initial distribution of traffic using routing R^1 and Q_{max} produced by Step 1 as input, and produces a set of switched-on links E_r and routing R^{β} of all demands in D, for $\beta\geq1$. Specifically, **Step 2** switches off each link that is not used to carry any traffic, *i.e.*, each link with f_{ij}=0; otherwise, for all links with $f_{ij}>0$, it calculates the residual capacity $r_{ij}=U_T*c_{ij}-f_{ij}$. For example, the step switches off links (1,2) and (4,1) since $f_{12}=f_{41}$= 0, and computes r_{24}=0.5*10-2.5=2.5, *etc*. Then, the step uses a temporary set E'_r to store candidate links, *i.e.*, links to be switched-off; we initialize $E'_r=E_r$. **Step 3** aims to remove a candidate link (i, j) with the largest residual capacity, *e.g.*, (2,4) with $r_{2,4}$=2.5, since rerouting its traffic is more probable, and updates E'_r and E_r. **Step 4** finds each demand d whose routing R_d^{β} uses the candidate link; we use a temporary set D' to store all of the affected demands.

Finally, **Step 5** uses **2DP-Routing**() to check if all affected demands in *D'* can be rerouted through the remaining links in E_r, while satisfying the following constraints: (i) at least Q_T*|*D*| demands are routed through their two disjoint paths, and (ii) each link utilization does not exceed U_T. If the function returns true (feasible), then the algorithm repeats from Step 2. For Fig. 2.1, switching off link (2,4) disconnects dp_{31}={sp_{31}, sp_{33}} since sp_{33} contains the link. Thus, the function replaces the 2DP with dp_{32}={sp_{31}, sp_{36}} since the replacement meets traffics requirement and satisfies the two constraints.

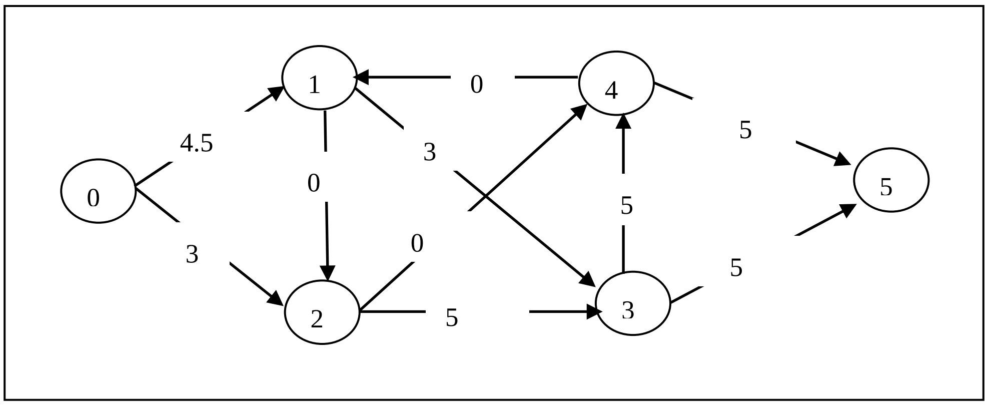


Figure 5.9 Running 2DP-SP-1 based on Fig. 2.1 with 2DP-L

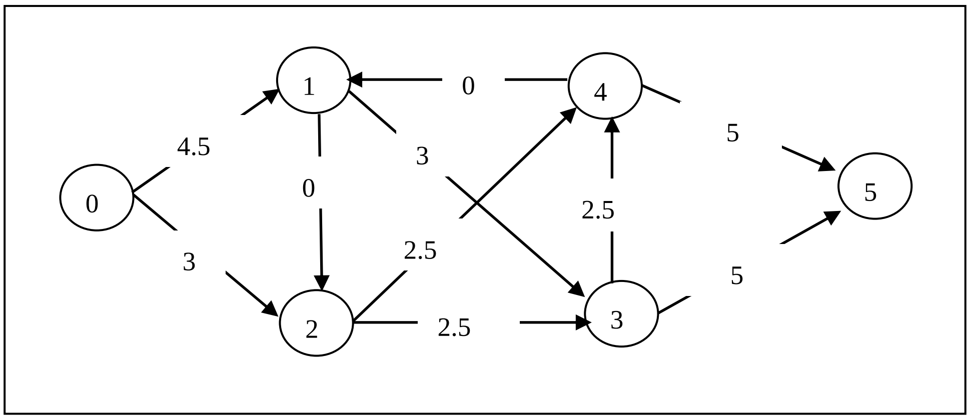


Figure 5.10 Running 2DP-SP-1 based on Fig. 2.1 with 2DP-N

Note that when **2DP-Routing**() returns true for all affected demands, the total flow in each affected link changes, and therefore Step 2 is used to update E_r, and the remaining capacity of each link. The step also reinitializes E_r'=E_r before repeating Step 3 to 5 for each candidate link. If **2DP-Routing**() fails, Step 5 switches on the candidate link (*i*, *j*), *i.e.*, E_r=E_r+{(*i*, *j*)}, and repeats from Step 3 for the next candidate link. For Fig. 2.1, 2DP-SP-1 produces the topology in Fig. 5.9, and thus is able to switch off 3 links, *i.e.*, (1,2), (2,4) and (4,1) since $f_{12}=f_{24}=f_{41}=0$. Notice that demands 2, 3 and 7 are routed through their 2DP, and each link utilization is no larger than $U_T=0.5$.

5.2.3 2DP-SP-1 for Node Disjoint Paths

In general, a network contains fewer 2DP-N than 2DP-L since each 2DP-N is also a 2DP-L, but not vice versa; thus using the latter for routing is more popular [14]. Further, using link-disjoint paths are much more energy efficient than node-disjoint paths [30]. However, 2DP-N is more resilient to failures than 2DP-L because they protect against both node and link failures. One can use 2DP-SP-1 for applications that require 2DP-N by considering only each 2DP-L, $dp_{dl}=\{sp_{dx}, sp_{dy}\}$, that is also a 2DP-N. For demand 3 in Fig. 2.1, $dp_{31}=\{sp_{31}, sp_{33}\}$ is also 2DP-N. Since, each set of 2DP-N for each (s_d, t_d) is a subset of its set of 2DP-L for each (s_d, t_d), 2DP-SP-1 is expected to switch off fewer number of links for 2DP-N as compared to 2DP-L. As shown in Fig. 5.10, 2DP-SP-1 is able to switch off only two links, *i.e.*, (1,2) and (4,1) since $f_{12}=f_{41}=0$, for 2DP-N as compared to three links for 2DP-L in Fig. 5.9.

5.2.4 Time Complexity

Yen's algorithm [30], used in Step 1 of 2DP-SP-1, requires $O(kn(m+nlogn))$ time to generate k shortest paths in a network with m links and n nodes for each demand d. Function Find-2DP() requires $O(k^2n^2logn)$ time for each demand d. Both function Use-2DP() and Use-Non-2DP() use, in the worst case, $O(km)$ time to route the flow of each d through its 2DP or multiple paths, and function 2DP-Routing() takes $O(km)$. As Step 1 considers all demands in D, its time complexity is $O(|D|(kn(m+nlogn)+ k^2n^2logn +km))=O(|D|(knm+k^2n^2logn))$. Step 2 takes $O(m)$ time, and Step 3 requires O(1) since the candidate link can be found as part of Step 2. Step 4 takes at most $O(|D|km)$ time, and Step 5 calls function 2DP-Routing() at most $O(|D'|)$ times. Notice that, in the worst case, there can be up to $(m(m-1)/2)$ candidate links since the set E'_r is updated in Step 2. Therefore, Step 2 to 5 are repeated $O(m^2)$ times and in total have time complexity of $O(|D|km^3)$. Thus, the time complexity of 2DP-SP-1 algorithm is $O(|D|(knm+k^2n^2logn) + |D|km^3) = O(|D|k^2n^2logn) + |D|km^3)$. Note that $|D|\leq n^2$.

To reduce the time complexity of 2DP-SP-1, we propose the following alternative algorithm, called 2DP-SP-A, which is different from 2DP-SP-1 only in the function Find-2DP(). Specifically, for 2DP-SP-A, we modify Find-2DP() such that it generates only one 2DP randomly instead of *all* possible 2DPs. The modification reduces the time complexity of the function to $O(km)$; the time complexity of the other functions, *i.e.*, Use-2DP(), Use-Non-2DP() and 2DP-Routing(), remain the same. Therefore, 2DP-SP-A has a time complexity of $O(|D|kn(m+nlogn)+m^2(|D|km+m+1+|D|km))= O(|D|km^3)$. As shown in Section 5.4.1.2, 2DP-SP-A significantly reduces the running time of 2DP-SP-1 but with lower power savings.

```
2DP-SP-2($G(V,E),D,Q_T,U$)
Begin
  $E_r = E, V_r = V$;
1) Generate $KSP_d$ in $G(V, E)$ for each demand $d$;
2) Call 2DPNL-Routing($G(V, E), D, Q_T=1.0, KSP$), and compute $Q_{max}$;
3) For each $v\in V_r$ do
        If $f_v == 0$ then
            $V_r = V_r - \{v\}$; // remove v
                $E_r = E_r - E_v$;
   End-For
4) For each $(i, j)\in E_r$ do
    If $f_{ij} == 0$ && flag($i, j$)==false then
      $E_r = E_r - \{(i, j)\}$; // remove (i, j)
        Else
      $r_{ij} = U*c_{ij} - f_{ij}$; // update the remaining capacity
      End-For
      Set $flag(i, j)$=false for each in $(i, j)$ in $E_r$;
5) For each $v\in V_r$ in increasing order of $|E_v|*(f_v/c_v)$ do
        //routing D in G with one less node is feasible
    If (2DPNL-Routing($G(V_r-\{v\}, E_r-E_v),D,Q_T, KSP$) == true) then
      $V_r = V_r - \{v\}$; // remove v
    $E_r = E_r - E_v$; // remove connected links
      Go to 3);
6) For each $(i, j)\in E_r$ in descending order of its $r_{ij}$ do
        //routing D in G with one less edge is feasible
    If (2DPNL-Routing($G(V_r, E_r-\{(i, j)\}),D,Q_T, KSP$) == true) then
      $E_r = E_r - \{(i, j)\}$; // remove $(i, j)$
        Go to 4);
 End-For
 Return $E-E_r$ & $V-V_r$
End
```

Figure 5.11 Algorithm 2DP-SP-2

5.3 Switching off both Nodes and Links

In this section, we present our approach, Two Disjoint Paths by Shortest Path version 2 (2DP-SP-2), shown in Fig. 5.11, to heuristically solve the EAR-2DP-2 problem. Initially, the set of remaining nodes V_r and links E_r are V and E respectively; 2DP-SP-2 produces $V-V_r$ and $E-E_r$ as its outputs. We first describe 2DP-SP-2 for 2DP-L, called 2DP-SP-2-L, in Section 5.3.1, and show how to use it for 2DP-N, called 2DP-SP-2-N, in Section 5.3.2.

5.3.1 2DP-SP-2 for Link Disjoint Paths

As shown in Fig. 5.11, 2DP-SP-2 for 2DP-L (2DP-SP-2-L) has six main steps. **Step 1** uses *Yen*'s algorithm [30] to generate $k\geq1$ shortest paths, $KSP_d=\{sp_{d1}, sp_{d2}, \ldots, sp_{dk}\}$, for each demand d; we assume each link has equal one unit delay, and thus each (s_d, t_d) path length equals to its (s_d, t_d) hop count. Note that for each demand d, we have $KSP_d \subseteq SP_d$. Let $KSP=\{KSP_d \mid d=1,\ldots,|D|\}$ be the set of all k-shortest paths for all demands D. **Step 2** uses function **2DPNL-Routing**() to distribute the traffic of each demand $d\in D$ through its candidate paths KSP_d, and computes Q_{max}; we will describe the function and Q_{max}'s calculation later. **Step 3** switches off each unused node v, *i.e.*, each node with f_v=0, and its incident links. **Step 4** turns off each unused link (i, j) with variable $flag(i, j)$=false, *i.e.*, each link with f_{ij}=0, and calculates the spare capacity $r_{ij}=U_T{*}c_{ij} - f_{ij}$ of each of other links. Specifically, $flag(i, j)$=true marks the links in the 2DP, which cannot be switched off. Let E_v be the set of incident links to node $v\in V$. **Step 5** aims to switch-off each node v, starting from v with fewer connected links $|E_v|$ and lower link utilization (f_v/c_v); such node is used by fewer flows and thus rerouting the flows is more probable. If there exists a feasible R^{β} without using v and its incident links in E_v, *i.e.*, **2DPNL-Routing**() returns true, the step switches off v and all links in E_v and repeats Step 3 updating flow of each affected node and edge; otherwise, **Step 5** is repeated using the next candidate node. **Step 6** aims to switch off each link (i, j) with the largest spare capacity since rerouting its traffic is more probable; this step uses function **2DPNL-Routing**() to check if all traffic can be routed through remaining links in E_r, while satisfying required constraints. The step is repeated for next candidate link if **2DPNL-Routing**() fails to generate a feasible R^β; otherwise, we repeat Step 4.

```
2DPNL-Routing($G(V_r, E_r)$, $D$,$Q_T$, $KSP$)
Begin
  $Temp\_TP = \varnothing$;
  For each $d \in D$ do
    /* Part 1 */
      Call Find-2DP($G(V_r, E_r)$, $d$) to generate $DP_d$;
    /* Part 2 */
      If $|Temp\_TP|<Q_T*|R^\beta|$ then
          If $|DP_d|>0$ then
                If Distribute-2DP ($d$)==true then
                    Insert $R_d^\beta$ into $Temp\_TP$;
              Else If Use-Non-2DP($d$)==false then
                    Return false;
          Else //if $|DP_d|==0$
              If Use-Non-2DP($d$)==false then
                  Return false;
      Else //Routing with multiple paths for energy saving
          If Use-Non-2DP($d$)==false then
              Return false;
  End-For
  If $|Temp\_TP|<Q_T*|R^\beta|$ then
      $TP^\beta=Temp\_TP$;
      Return true;
  Else
      Return false;
End
```

Figure 5.12 Function 2DPNL-Routing()

Function **2DPNL-Routing**(), shown in Fig. 5.12, contains two parts. **Part 1** uses function **Find-2DP**(), shown in Fig. 5.5, to generate all 2DP-Ls, $dp_{dl}=\{(sp_{dx}, sp_{dy})| sp_{dx}\in KSP_d\}$, for each demand d. **Part 2** routes all traffic demands under the constraint Q_T. It uses a set *Temp_TP*, initially empty to stores R_d^β for each demand d. If the requirement of Q_T is satisfied, *i.e.*, $|Temp\_TP|\geq Q_T*|R^\beta|$, the remaining traffic demands can be routed through via any routes using function **Use-Non-2DP**() , shown in Fig. 5.8. The function aims to distribute each demand (s_d, t_d, b_d) via its shortest (s_d, t_d) path. However if the path does not have sufficient capacity, the function will route the remaining flow through the next available shortest path. The step is repeated until b_d is completely routed and the function returns true; otherwise, it returns false and

2DPNL-Routing() returns false since it fails to route all demands in D. The benefit of using general multiple paths is using fewer links for routing a demand than using 2DP-L. However, if $|Temp\_TP|<Q_T*|R^{\beta}|$ and at least one 2DP-L of demand d exists, *i.e.*, $|DP_d|>0$, **Part 2** uses function **Distribute-2DP**(), described later, to distribute traffic volume of demand d via its dp_{dl} and insert R_d^{β} into $Temp\_TP$. For each demand d, if $|DP_d|=0$ or **Distribute-2DP()** returns false, **2DPNL-Routing**() uses function **Use-Non-2DP()** to route b_d via one or more paths starting from the shortest path in KSP_d. Finally, if all traffic demands are allocated successfully and the requirement $|Temp\_TP|\geq Q_T*|R^{\beta}|$ is satisfied, $Temp\_TP=TP^{\beta}$ and **2DPNL-Routing**() returns true. When function **2DPNL-Routing**() returns true, *i.e.*, it has successfully routed all demands in D, it updates R_d^{β} for each demand d and the total flows on each link $(i, j)\in E_r$, *i.e.*, f_{ij}. When function returns false, it will maintain the previous routing $R^{\beta-1}$.

```
Distribute-2DP ($KSP_d$ , $DP_d$, $b_d$)
Begin
 //Distribute traffics with 2DP-N
  For each $dp_{dl}=\{sp_{dx}, sp_{dy}\}\in DP_d$ in increasing length order do
     If $b_d \leq$ B($sp_{dx}$)+ B($sp_{dy}$) then
        If $b_d \leq$ B($sp_{dx}$) then
            Increase $f_{ij}$ of each link $(i, j)$ in $sp_{dx}$ by $b_d$;
            Set $flag(i, j)$=true for each link $(i, j)$ in $sp_{dy}$;
        Else
            Increase $f_{ij}$ of each link $(i, j)$ in $sp_{dx}$ by B($sp_{dx}$);
            Increase $f_{ij}$ of each link $(i, j)$ in $sp_{dx}$ by $b_d$–B($sp_{dx}$);
        Insert $sp_{dy}$ and $sp_{dx}$ in $R_d^{\beta}$ ;
          Return true;
  End for
  Increase $f_{ij}$ of each link $(i, j)$ in $sp_{dx}\in dp_{d1}$ by B($sp_{dx}$);
  Increase $f_{ij}$ of each link $(i, j)$ in $sp_{dy}\in dp_{d1}$ by B($sp_{dy}$);
  Insert $sp_{dy}$ and $sp_{dx}$ in $R_d^{\beta}$ ;
  Return Use-Non-2DP($KSP_d$, $b_d$ – B($sp_{dx}$) – B($sp_{dy}$));
End
```

Figure 5.13 Function Distribute-2DP()

For each demand with $|DP_d|>0$, **Distribute-2DP**(), shown in Fig. 5.13, routes traffic demand b_d. The function aims to route the traffic d through its 2DP-L, *i.e.*, $(sp_{dx}, sp_{dy})\in dp_{dl}$, and we assume $B(sp_{dx})\leq B(sp_{dy})$. If $b_d\leq B(sp_{dx})$ then it routes b_d through only a single path sp_{dx}, and sets $flag(i, j)$=true for each (i, j) backup path sp_{dy} that can enhance routing reliability. This flow distribution is different than in [10] that splits flow equally. If B(sp_{dx})≤b_d≤ B(sp_{dx})+B(sp_{dy}) then it routes the traffic volume B(sp_{dx}) via sp_{dx}, and routes volume b_d-B(sp_{dx}) via sp_{dy}. Note that, B(sp_{dq})=min{r_{ij}| (i, j)∈sp_{dq}}. However, if B(sp_{dx})+B(sp_{dy})<b_d, it uses function **Use-Non-2DP()** to distribute the remaining flow, *i.e.*, b_d-(B(sp_{dy})+B(sp_{dx})). Note that **Distribute-2DP()** returns false when **Use-Non-2DP()** returns false, *i.e.*, it fails to route the remaining flows through the paths in KSP_d; for this case **2DPNL-Routing**() will use function **Use-Non-2DP()** to route b_d. Recall that **Step 2** of 2DP-SP-2-L in Fig. 5.11 uses function **2DPNL-Routing**() to initialize the traffic distribution in the network. In this step, we set Q_T=1.0 so that the function routes each demand through its 2DP-L whenever possible, and thus Q_{max} is set |TP^1|/|D|, where TP^1 includes all demands in D that are routed via 2DP-L routing in the original network.

5.3.2 2DP-SP-2 for Node Disjoint Paths

In general, a network contains fewer 2DP-Ns than 2DP-Ls since each 2DP-N is also a 2DP-L, but not vice versa; thus using the latter for routing is more popular [31]. Further, using link-disjoint paths are much more energy efficient than node-disjoint paths [113]. However, 2DP-N is more resilient to failures than 2DP-L because they protect against both node and link failures. One can use 2DP-SP-2 for applications that require 2DP-N (2DP-SP-2-N) by considering only each 2DP-L, dp_{dl}={sp_{dx}, sp_{dy}}, since each set of 2DP-N for each (s_d, t_d) is a subset of its set of 2DP-L for each (s_d, t_d). However, similar to 2DP-SP-1, 2DP-SP-2-N is expected to switch off less number of links than 2DP-SP-2-L due to fewer candidate 2DP-Ns.

5.3.3 2DP-SP Implementation

While the results shown in Section 5.4 show great opportunities to save power consumption in real networks, today's technology may not fully support the selective shutdown of links required by 2DP-SP. Based on the discussion of implementation issues in Section 2.5.4, we describe the implementation of 2DP-SP as follows.

For compatibility with existing protocols, we assume 2DP-SP is run in a centralized controller at the NOC. 2DP-SP also requires hardware support, *i.e.*, line-cards that can go into sleep or active state in milliseconds [59]. To minimize packet loss during transitions, it is important that links are not powered off immediately due to buffered packets. In practice, we will need to wait for all alternative paths to be set up before shutting down links.

The controller collects network information and traffic matrix from OSPF's LSAs and Management Information Bases (MIB), *i.e.*, MPLS's SNMP counter. We collect IP traffic statistics on all line-cards where NetFlow [114] is enabled. The collected information is then used by the controller when executing 2DP-SP. Specifically, the controller runs 2DP-SP to select the subset of resources that must be powered on to meet current traffic demands, and issues a control message containing a list of links to be powered off to routers; this operation can be achieved using an OSPF extension such as RFC 3630 [98]. The network configuration can be adjusted every 30 minutes, which is compatible with the slow and daily variation of traffic in current backbone networks. Note that fewer network reconfigurations reduce potential power saving but mitigate latencies incurred when changing power state. Similar to the model in [88], we assume that a network operator changes network configuration infrequently, *i.e.*, only during the off-peak periods, to reduce the risk of network oscillations.

Multipath forwarding is supported by commercial routers such as Juniper networks [115] and Cisco [116]. In 2DP-SP, data packets are forwarded along either OSPF paths or using MPLS tunnels (*i.e.*, LSP). Specifically, if a 2DP-SP path is the shortest path, the traffic is simply transmitted as native IP packets as per OSPF; otherwise the traffic is carried using MPLS tunnels over the 2DP-SP path, with the traffic split as per ECMP [52].

5.4 Evaluation

In this section, we evaluate the performance of two versions of 2DP-SP, *i.e.*, both 2DP-SP-1 and 2DP-SP-2.

5.4.1 Experimental Simulation for 2DP-SP-1

In this section, we provide detailed experimental findings and present numerical results to confirm the effectiveness of 2DP-SP-1. We evaluate 2DP-SP-1 and show that it can achieve considerable power savings in real networks with low impact on link utilization, path length, and RR.

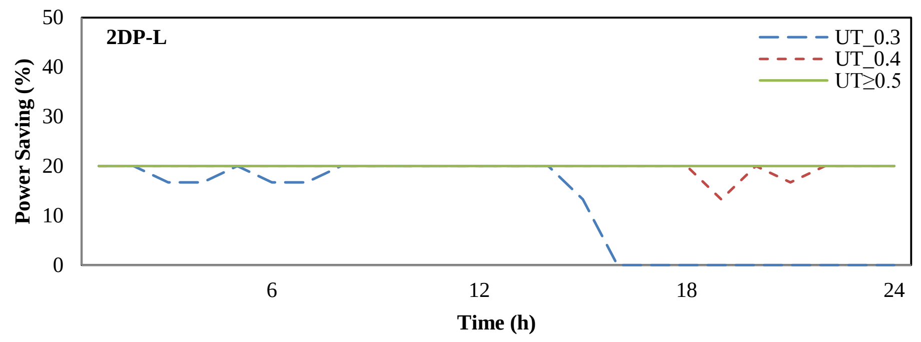
5.4.1.1 Experiment Setup

We explore the power saving using 2DP-SP-1 for different network topologies and traffic matrices, shown in Section 2.6. We used Phase 1 of 2DP-SP-1 to obtain Q_{max}=83.3%, Q_{max}=100%, and Q_{max}=37.4% for Abilene, GÉANT, and Sprint, respectively. In other words, when all links are switched-on while using shortest path routing, there are only 83.3%, 100%, and 37.4% of demands in Abilene, GÉANT, and Sprint networks that can be routed through their 2DP, respectively.

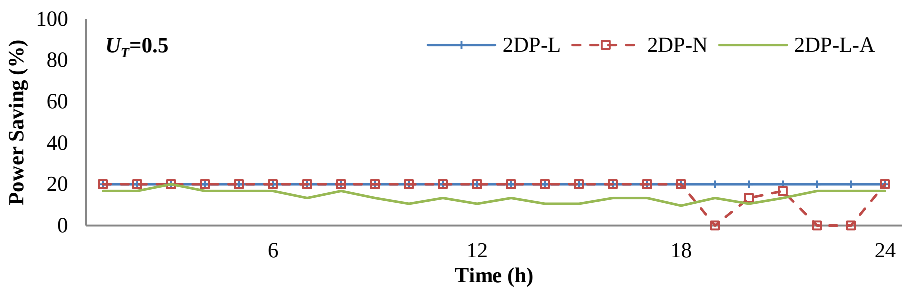
5.4.1.2 Power Saving

Fig. 5.14(a) shows the power savings for the Abilene network using 2DP-SP-1 for 2DP-L when we set U_T from 0.1 to 1.0, and Q_T=Q_{max}=83.3%. Specifically, we aim to see the effect of link utilization constraint on power saving. Note, 2DP-SP-1 could not switch off any link for 2DP-L with $U_T \leq 0.2$ for all traffic demands. For 2DP-L with U_T=0.3, 2DP-SP-1 is able to switch off between 16.5% and 20% of the links except after time 16h due to insufficient network capacity during peak traffic flow; see Fig. 5.14(a). Further, for $U_T \geq 0.5$, 2DP-L distributes traffic flow through the same set of paths, and thus produces the same power savings. Fig. 5.14(b) compares the performance, in term of power saving, between 2DP-SP-1 for 2DP-L and 2DP-N, and 2DP-SP-A for 2DP-L, *i.e.*, 2DP-L-A, when we set U_T=0.5 and Q_T=Q_{max}=83.3%; we do not show the results of 2DP-SP-A for 2DP-N since they are the same as those for 2DP-L-A. The figure shows that 2DP-L obtains higher power savings than 2DP-N in the peak hours (19h~22h); 2DP-SP-A is the worst performer, producing on average 6%

less power saving than 2DP-L. However, as shown in Table 5.1, 2DP-SP-A runs significantly faster than 2DP-SP-1 on all networks, *i.e.*, Abilene, GÉANT, and Sprint. Notice that the running times of 2DP-SP-1 for 2DP-L and 2DP-N are comparable across all three topologies. 2DP-SP-1 is significantly slower than 2DP-SP-A because it requires generating all possible 2DPs for each demand so that the demand can be routed equally on its two paths when possible to support ECMP routing. Specifically, 2DP-SP-A does not support ECMP routing.



(a) Energy saving under 2DP-L



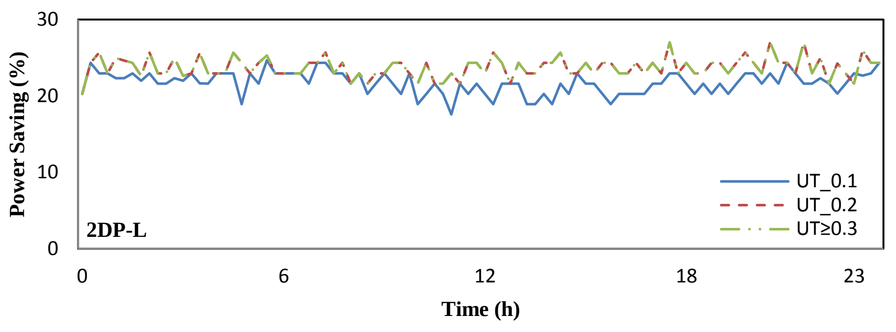
(b) Comparison of energy saving between 2DP-SP-1 and 2DP-SP-A

Figure 5.14 Power saving on Abilene using 2DP-SP-1 and 2DP-SP-A

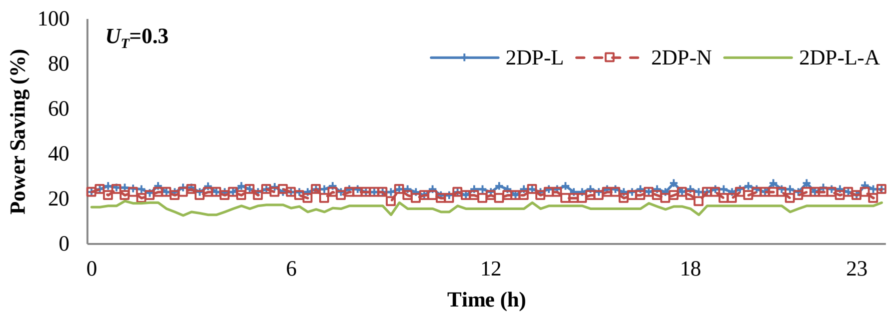
Table 5.1 Average Running Time (Seconds)

Network	**2DP-L**	**2DP-N**	**2DP-L-A**
Abilene	5.83	4.91	0.081
GÉANT	685.86	543.26	25.87
Sprint	5436.23	5127.85	683.6

Fig. 5.15(a) presents the power saving on GÉANT for 2DP-L when we set Q_T to its highest possible constraint, *i.e.*, Q_T=Q_{max}=100%. Unlike for Abilene, 2DP-SP-1 is able to switch off 24.67% of links in the GÉANT for U_T=0.1.



(a) Energy saving under 2DP-L



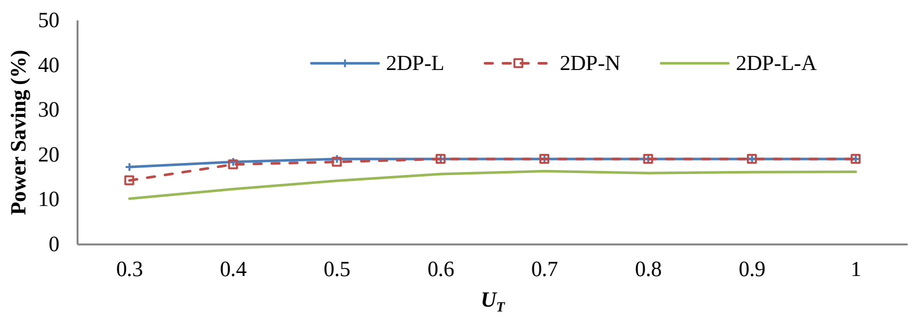
(b) Comparison of energy saving between 2DP-SP and 2DP-SP-A

Figure 5.15 Power saving on GÉANT using 2DP-SP-1 and 2DP-SP-A

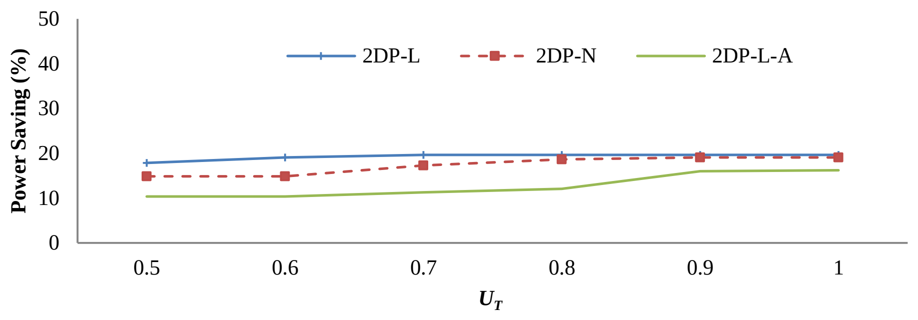
In Fig. 5.15, our approach obtains the same power saving for U_T=0.3 to U_T=1.0, and thus we only show the results for U_T≥0.3. Notice that, as described later in Section 5.4.1.4, U_T=0.2 and U_T=0.3 produce the same power saving but incur different path length. Although the power saving curve fluctuates during the day due to traffic changes, it always remains around 20.27%~25.97%. As shown Fig. 5.15(b), 2DP-SP-1 produces less power saving for 2DP-N as compared to for 2DP-L when we set U_T=0.3 since 2DP-N has fewer 2DPs than 2DP-L as candidate paths. Although 2DP-L-A runs faster than 2DP-L and 2DP-N, its power savings are lower than 2DP-L by up to 8% because 2DP-SP-A only generates one random 2DP.

Fig. 5.16(a), (b) and (c) compare the power savings produced by 2DP-SP-1 for both 2DP-L and 2DP-N, and 2DP-SP-A for 2DP-L on Sprint with traffic matrices SP(30%), SP(60%), and SP(80%) respectively with U_T ranging from 0.1 to 1.0. Note that, in Fig. 5.16, 2DP-SP-1 produces the same set of routes for SP(10%), SP(20%) and SP(30%), and thus we only reported the result only for SP(30%); similarly for

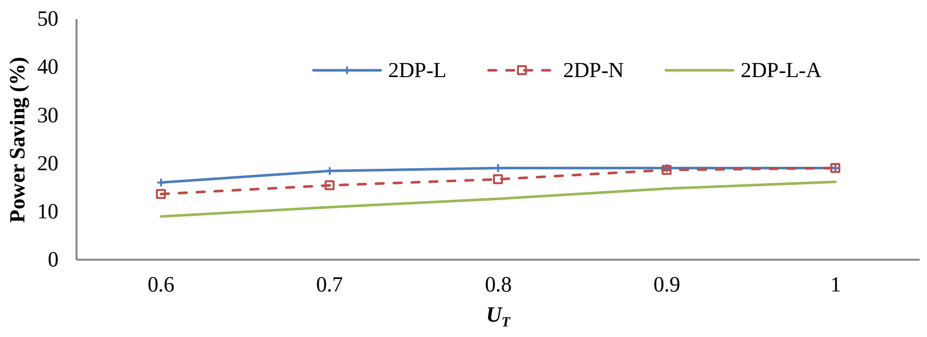
SP(40%) to SP(60%), and for SP(70%) to SP(80%). Further, 2DP-SP-1 failed to switch-off links for SP(90%) and SP(100%).



(a) SP(30%)



(b) SP(60%)

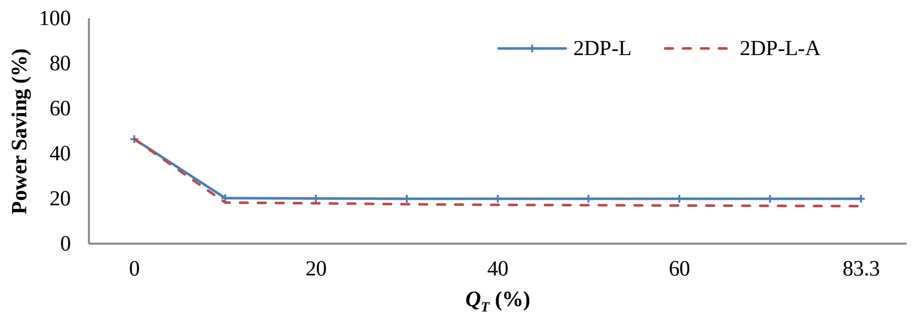


(c) SP(80%)

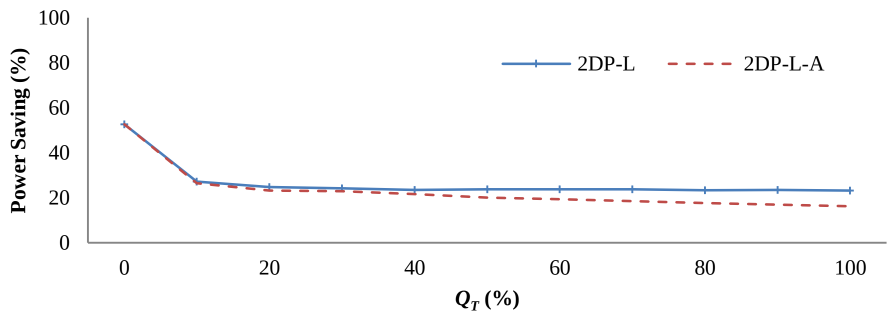
Figure 5.16 Power saving on Sprint using 2DP-SP-1 and 2DP-SP-A

As shown in Fig. 5.16(a), the power saving using 2DP-L increases from 17.26% to 19.04% when U_T increases from 0.3 to 0.5. The figure shows similar trend for 2DP-N but with less power savings, *i.e.*, at 14.29% for U_T=0.5. However, the power savings for both 2DP-L and 2DP-N are equal when U_T>0.6. 2DP-SP-1 for 2DP-L produces better results since it can use more alternative 2DP-L as compared to 2DP-N.

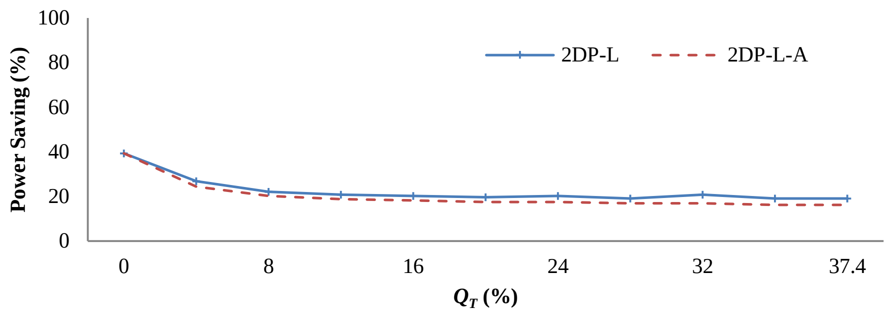
Although 2DP-SP-1 switches off more links than 2DP-SP-A, the latter runs significantly faster than the former. Notice that for heavier traffics, *i.e.*, SP(60%) and SP(80%) versus SP(30%), both 2DP-SP-1 and 2DP-SP-A can switch-off links only for larger U_T values, *i.e.*, above 0.5 and 0.6 respectively.



(a) Abilene



(b) GÉANT

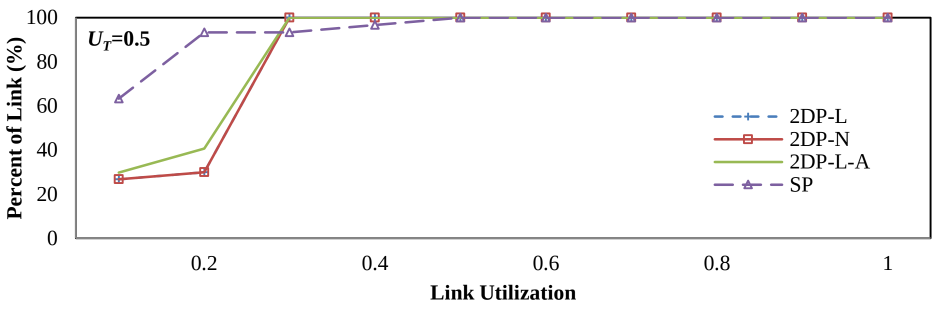


(c) Sprint

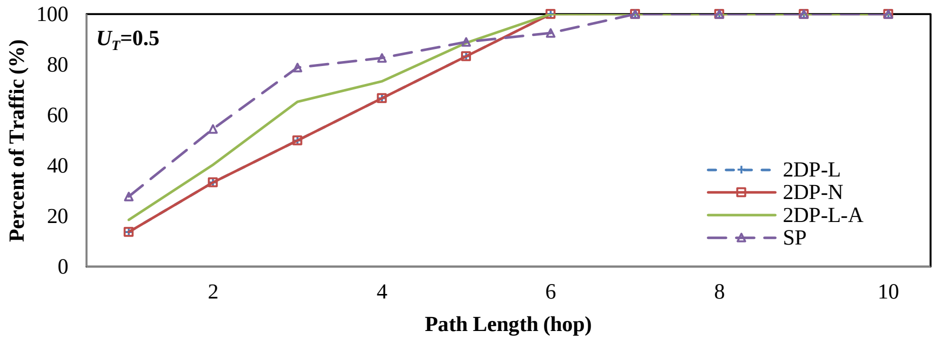
Figure 5.17 Power saving for different Q_T for three real topologies

5.4.1.3 Effects of Different Q_T on Power Saving

In this subsection, we analyze the effects of different Q_T on power saving for U_T=1.0. We do not show the results for 2DP-SP-1 for 2DP-N because the results show the same trend as for 2DP-L. As shown in Fig. 5.17(a), the power saving on the Abilene network is 46.38% when Q_T is set to 0% because both 2DP-L and 2DP-L-A can switch-off more links when demands are routed via multiple shortest paths, *i.e.*, non-2DP. For $Q_T \geq 13.3\%$, 2DP-L and 2DP-L-A produce power saving of only around 20% and 17% respectively.



(a) Link utilization under 2DP-SP-1, 2DP-SP-A and SP



(b) Path length under 2DP-SP-1, 2DP-SP-A and SP

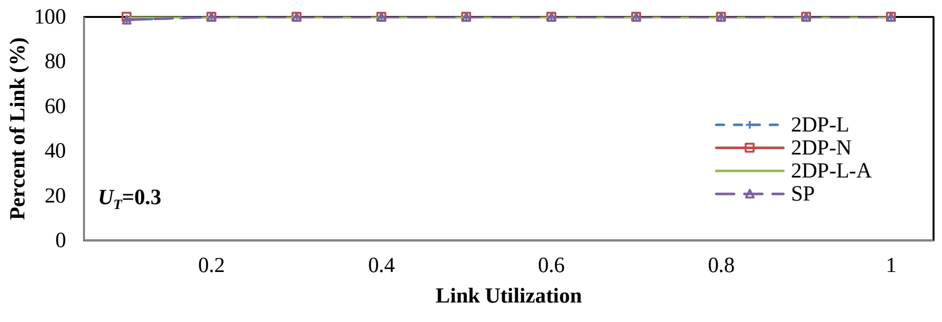
Figure 5.18 CDF of link utilization and path length on Abilene

Similarly, Fig. 5.17(b) and Fig. 5.17(c) show that the power saving of GÉANT and Sprint decreases from 52.7% to 23.26% and from 39.29% to 19.05% for 2DP-L when Q_T increases from 0% to 100% and from 0% to 37.4%, respectively. Notice that, after a sharp decline in power saving at Q_T=13.3%, Q_T=15%, and Q_T=8% for Abilene, GÉANT, and Sprint respectively, for both 2DP-L and 2DP-L-A, increasing Q_T does not significantly affect their power savings. Thus, networks that offer 2DP should use the algorithms with the highest Q_T, that is Q_T=Q_{max}, to optimize fault-tolerance that is important for real time applications such as video on demand and voice over Internet Protocol [1].

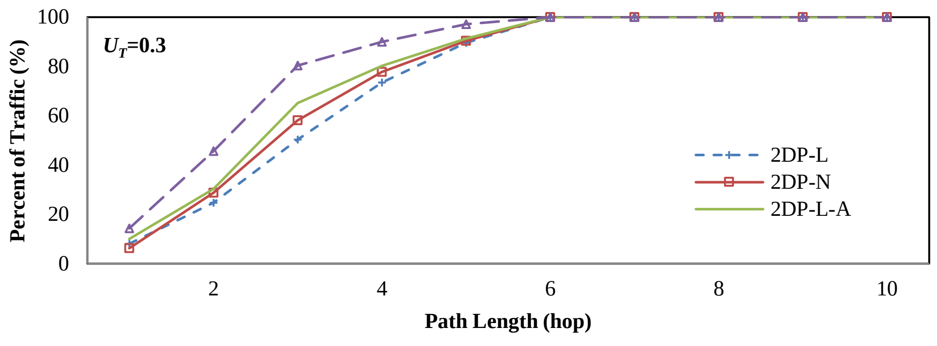
5.4.1.4 Link Utilization and Path Length

In this subsection, we show the effect of switching off links on link utilization and path length using 2DP-SP-A for 2DP-L, *i.e.*, 2DP-L-A, and 2DP-SP-1 for both 2DP-L and 2DP-N. As the benchmark, we use the link utilization and path length incurred by the SP. Fig. 5.18(a) shows the CDF of link utilization for the Abilene while setting U_T≥0.5 for SP, 2DP-L-A, 2DP-L and 2DP-N. As shown in the figure, the link utilization for 2DP-L, 2DP-N and 2DP-L-A never exceeds 0.3, which is better as compared to SP that has 10% of links with utilization above 0.3. However, 80% of links in SP has utilization no more than 0.2, better as compared to the 2DP-L and 2DP-N with only 30% and 2DP-L-A with 40%. The results show the benefit of our 2DP-SP-1 since it also reduces energy; see Section 5.4.1.2. Notice that 2DP-L and 2DP-N produce the same results because this case produces set of disjoint paths that are both link disjoint and node disjoint. Fig. 5.18(b) shows the CDF of path length for the four schemes. The figure shows that 2DP-L, 2DP-N and 2DP-L-A use paths that are longer as compared to SP. Our schemes trade off longer alternative paths for on average 20% power savings. However, like in SP, their paths are no longer than the network diameter of 6 hops. Notice that 2DP-L-A uses paths with slightly shorter length as compared to 2DP-L and 2DP-N that use paths with equal length because it switches off fewer links.

As shown in Fig. 5.19(a) for GÉANT, 2DP-L, 2DP-N, 2DP-L-A and SP have the same CDF for their link utilization where all links have utilization no more than 0.1. However, 2DP-L, 2DP-N and 2DP-L-A are able to reduce energy, and therefore they are more favorable than SP. For delay, as shown in Fig. 5.19(b), more paths in SP have shorter length as compared to 2DP-N, 2DP-L and 2DP-L-A. 2DP-L-A is the second best performer while 2DP-L is the worst but, as discussed in Section 5.4.1.2, it can save the largest amount of energy. However, all schemes use paths with length no more than the network diameter (6 hops).



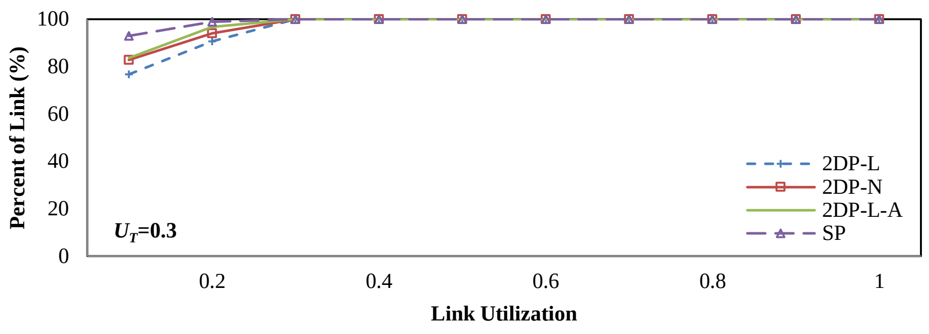
(a) Link utilization under 2DP-SP, 2DP-SP-A and SP



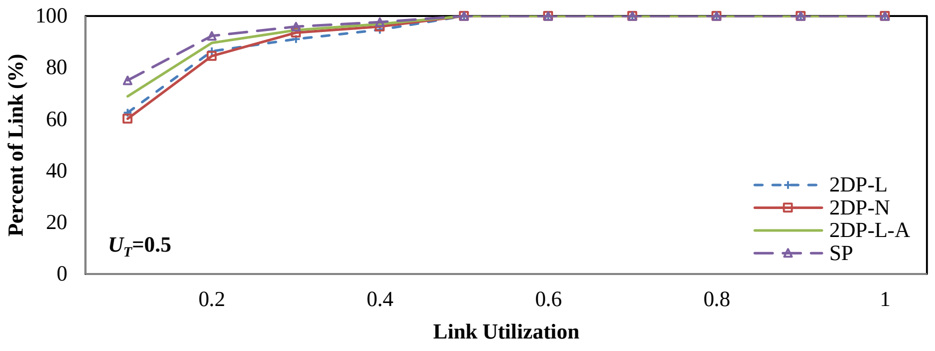
(b) Path length under 2DP-SP, 2DP-SP-A and SP

Figure 5.19 CDF of link utilization on Sprint on GÉANT

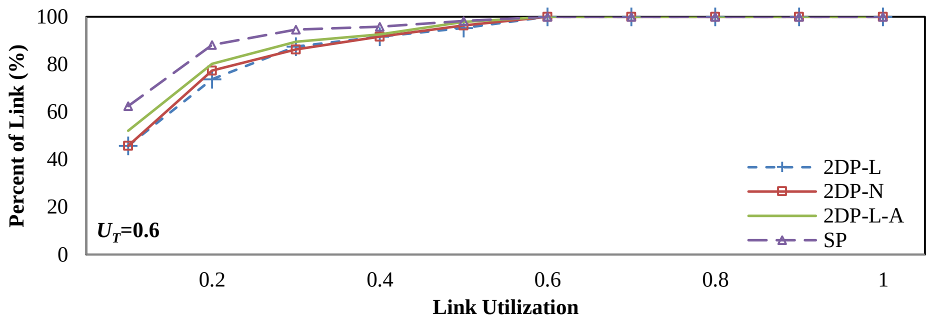
Fig. 5.20 shows the CDF of link utilization for (a) SP(30%) with $U_T \geq 0.3$, (b) SP(60%) with $U_T \geq 0.5$, and (c) SP(80%) with $U_T \geq 0.6$. The three figures consistently show that 2DP-SP-1 for both 2DP-L and 2DP-N, and 2DP-SP-A for 2DP-L, *i.e.*, 2DP-L-A have slightly worse results of CDF as compared to SP since they switched-off links that increase link utilization of the remaining links; as in other simulations, 2DP-L-A is the second best performer. Consistent with the results for Abilene and GÉANT, the CDF of path length for Sprint, shown in Fig. 5.21 (a), (b) and (c), confirms that 2DP-L, 2DP-N and 2DP-L-A use longer paths to route demands to switch-off links. However, each path length is no longer than the network diameter (8 hops).



(a) SP(30%)



(b) SP(60%)

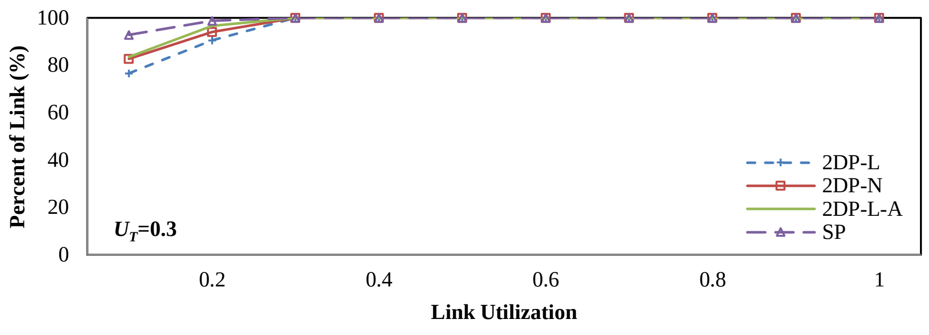


(c) SP(80%)

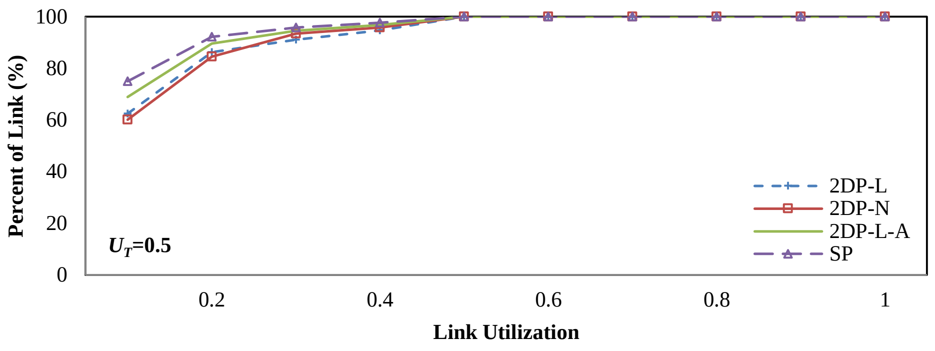
Figure 5.20 CDF of link utilization on Sprint

5.4.1.5 Effect on Route Reliability

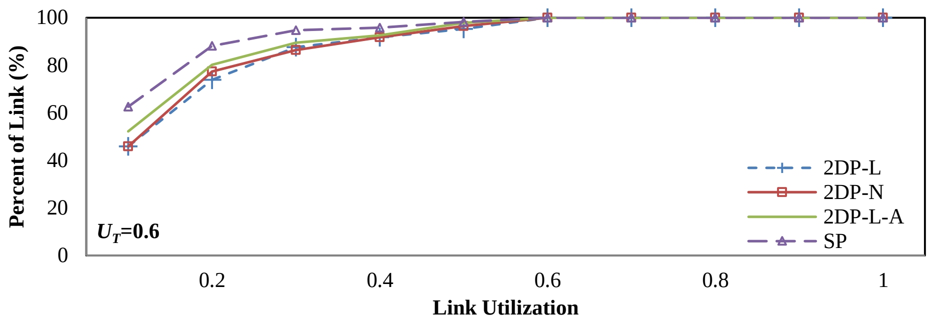
In this sub-section, we evaluate the effect of power saving using 2DP-SP-1 for both 2DP-L and 2DP-N, *i.e.*, switching off links, on the reliability of each route used to transmit demand *d*; we call the reliability as RR. Note that, since 2DP-L-A has the same results on RR as compared to 2DP-L, we do not show its results.



(a) SP(30%)



(b) SP(60%)



(c) SP(80%)

Figure 5.21 CDF of path length on Sprint

Let link reliability, $0 \leq \rho_{ij} \leq 1.0$, be the probability of link (i, j) being functional. In this experiment, we assume each link has the same ρ_{ij}=0.9. We consider each demand d to be routed in three different ways (a) via a single path, (b) via two-link disjoint paths, (c) via multiple non-disjoint paths. One can compute the reliability of route in (a) by multiplying the reliability ρ_{ij} of each link in the path. For (b), its reliability is analogous to a two parallel system [119]. Specifically, the reliability of routing a demand d via 2DP, each with reliability x and y is 1-(1-x)(1-y). Finally, we use

CAREL [109] to compute the RR for each demand *d* that is routed via non-disjoint multiple paths, *i.e.*, case (c).

Table 5.2 Variation of RR on Abilene (U_T=0.6, Q_T=83.3%) (%)

2DP-SP-1	+Δ		-Δ		Δ=**0**		**PS**
	SP	**2DP**	**SP**	**2DP**	**SP**	**2DP**	
2DP-L	75.8	3	13.6	81.8	10.6	15.2	20
2DP-N	75.8	0	13.6	84.9	10.6	15.1	20

Table 5.3 Variation of RR on GÉANT (U_T=0.3, Q_T=100%) (%)

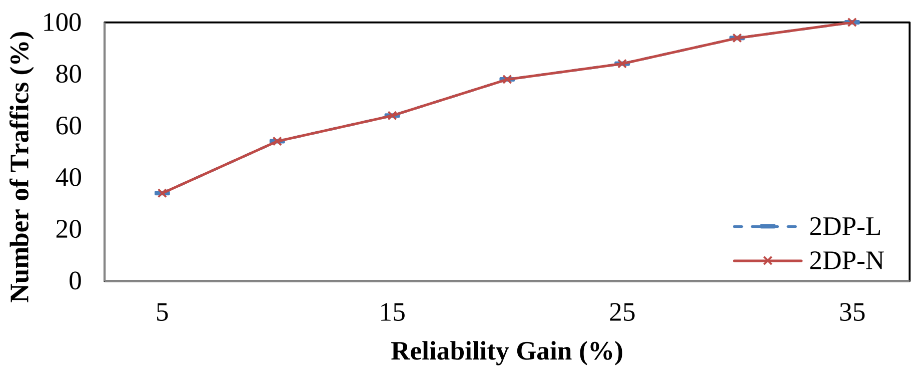
2DP-SP-1	+Δ		-Δ		Δ=**0**		**PS**
	SP	**2DP**	**SP**	**2DP**	**SP**	**2DP**	
2DP-L	94.2	0	5.8	76.8	0	23.2	1.6
2DP-N	97.3	1.1	2.7	69.3	0	29.6	20.2

We use the reliability of the SP routing as the benchmark. Table 5.2, 5.3, and 5.4 show the RR produced by 2DP-L and 2DP-N as compared to SP. Specifically, "+Δ" represents percentage of alternative paths produced by either 2DP-L or 2DP-N with reliability higher than SP or 2DP on the original network, *i.e.*, without power savings. Similarly, "-Δ" ("Δ=0") represents percentage of alternative paths produced by either 2DP-L or 2DP-N with reliability lower (equal) than SP and 2DP routing without power savings on the original network. Note that "PS" in each table represents power savings for 2DP-L and 2DP-N.

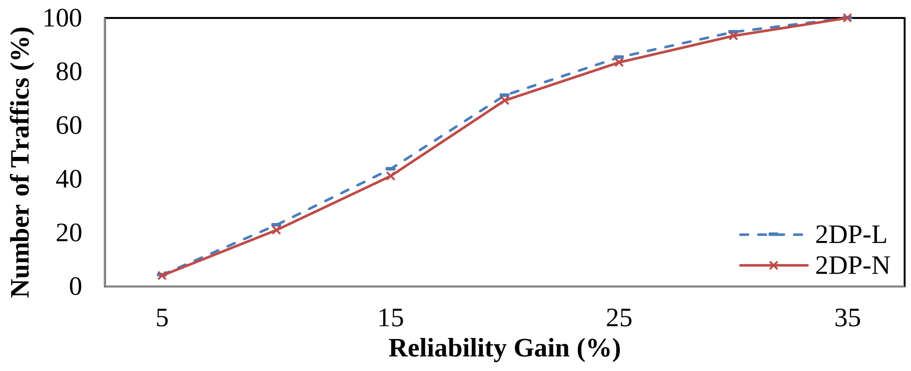
Table 5.4 Variation of RR on Sprint (U_T=0.6, Q_T=37.4%) (%)

2DP-SP-1	+Δ		-Δ		Δ=**0**		**PS**
	SP	**2DP**	**SP**	**2DP**	**SP**	**2DP**	
2DP-L	36.8	1.3	10.2	32.5	53	66.2	19.2
2DP-N	36.9	1	9.5	28.6	53.6	70.4	16.7

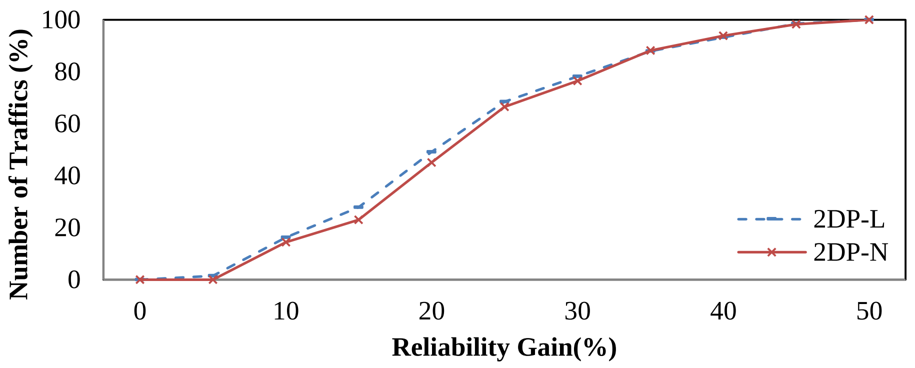
As shown in Table 5.2, for Abilene with U_T=0.6 and Q_T=83.3%, 2DP-SP-1 produces 75.76% of alternative paths for 2DP-L and 2DP-N with higher reliability as compared to their original SP routing, with only 13.64% having lower reliability. The reliability improvement is due to the use of 2DP-L as compared to only a single path in SP. Note that some alternative paths are less reliable since they use more links than the original paths.



(a) Abilene



(b) GÉANT

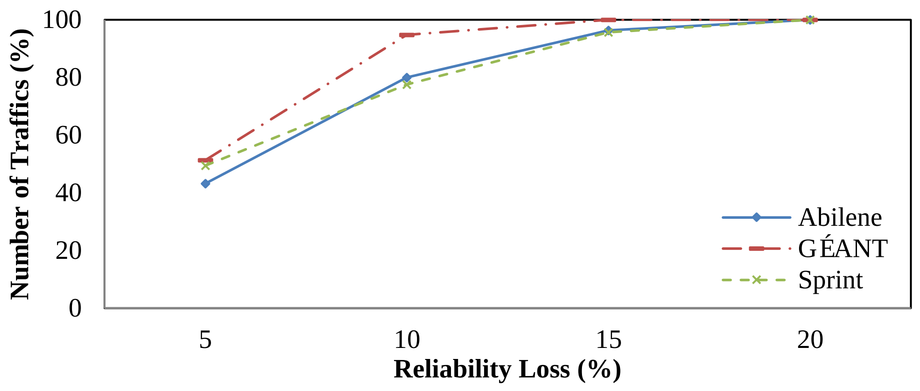


(c) Sprint

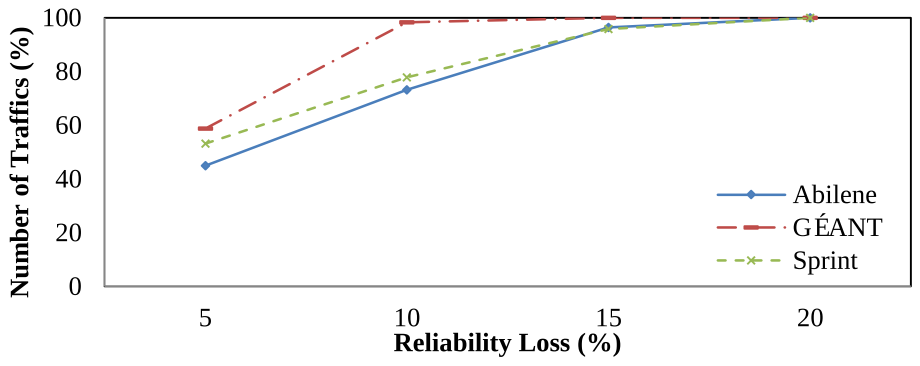
Figure 5.22 CDF of RR gain using 2DP-SP-1

In contrast, 3% (81%) of the generated paths for 2DP-L have lower (higher) reliability as compared to using 2DP-L routing on the original network; 15.2% of them have the same reliability since 2DP-SP-1 does not generate alternative paths, as shown in Table 5.2. The results for 2DP-N are similar. The lower RR is justifiable since 2DP-SP-1 is able to switch off 20% of links, and thus is forced to find longer/less reliable alternative paths. For GÉANT, Table 5.3 shows 2DP-SP-1 for 2DP-L (2DP-N) is able to route 94.2% (97.3%) of all demands through paths with increased reliability as compared to using their original SP routing, with only 5.8% (2.7%)

decreases in RR. The results are significant since 2DP-SP-1 increases the RR while reducing power saving by up to 21.6%. As expected, 2DP-N offers higher RR as compared to 2DP-L but with less power saving. However, similar to the results for Abilene, the generated alternative paths are less reliable as compared to their corresponding 2DP-L and 2DP-N paths on the original network. Table 5.4, for Sprint with U_T=0.6 and Q_T=37.4%, also shows the merits of using 2DP-SP-1. Specifically, as compared to SP on network without power saving, 36.8% (36.9%) of total routes using 2DP-L (2DP-N) are more reliable as compared to their SP routings on original network. The Table shows that 2DP-SP-1 switch off up to 19.2% links to produce up to 32.5% paths with lower reliability as compared to 2DP routing on the original network.



(a) 2DP-L



(b) 2DP-N

Figure 5.23 CDF of reliability loss using 2DP-SP-1

To further evaluate the effect of switching off links on RR, in Fig 5.22 (Fig. 5.23) we plot the CDF of reliability gain (loss) of paths that increase (decrease) in reliability with respect to the reliability of shortest paths. Fig. 5.22(a), (b) and (c) present CDF of reliability gain (in percent) while using 2DP-L and 2DP-N for Abilene, GÉANT, and Sprint respectively. As shown in Fig. 5.22, almost 40% (5%) traffics for either 2DP-L or 2DP-N have 5% (35%) increases in RR. Further, more than 50% (80%) of traffics in GÉANT (Sprint) increase their reliability by at least 15%, with maximum increase of 35% (50%). The results show the merits of using either 2DP-L or 2DP-N as compared to SP for applications that require higher path reliabilities. As shown in Fig. 5.23, although 2DP-SP-1 may generate routes with lower reliability as compared to using 2DP routing on the original networks, the reliability loss is less than 20%. Specifically, for Abilene and Sprint with 2DP-L (2DP-N), about 80% (70%) of paths decrease their reliability by 10%, with only 5% decrease their reliability by 20%. Notice that 2DP-SP-1 performs the best on GÉANT, with about 95% of paths lower their reliability by up to 10%.

5.4.1.6 2DP-SP-1 versus Existing Green Routing Approaches

In this subsection, we compare the performances of 2DP-SP-1 for 2DP-L, in terms of power saving and RR, against the existing state-of-the art green routing algorithms, *i.e.*, FGH [15], GreenTE [16], MSPF [40], and SSPF [38]. Note that FGH, MSPF, and SSPF consider each network link contains one or more cables that can be switched-off independently. For fair comparison, we set each link to contain only one cable since GreenTE and 2DP-SP-1 do not support bundled links. Further, we set U_T=100% since FGH does not support link utilization constraint. Finally, because FGH, GreenTE, MSPF, and SSPF do not support the constraint Q_T, we only use Q_T=Q_{max} for 2DP-SP-1.

Fig. 24, 25 and 26 show the power savings of five energy-aware routing approaches, *i.e.*, FGH, GreenTE, SSPF, MSPF, and 2DP-SP-1, on Abilene, GÉANT and Sprint respectively. As shown in the figures, SSPF performs best on small and medium networks, while MSPF performs best on the largest network, *i.e.*, Sprint, since large topologies provide sufficient candidate paths for multiple paths routing. Further, 2DP-SP-1 only obtains nearly half of the power saving of MSPF since the former requires constraint Q_T; 2DP-SP-1 produces the same power savings as compared to MSPF when we set Q_T=0.

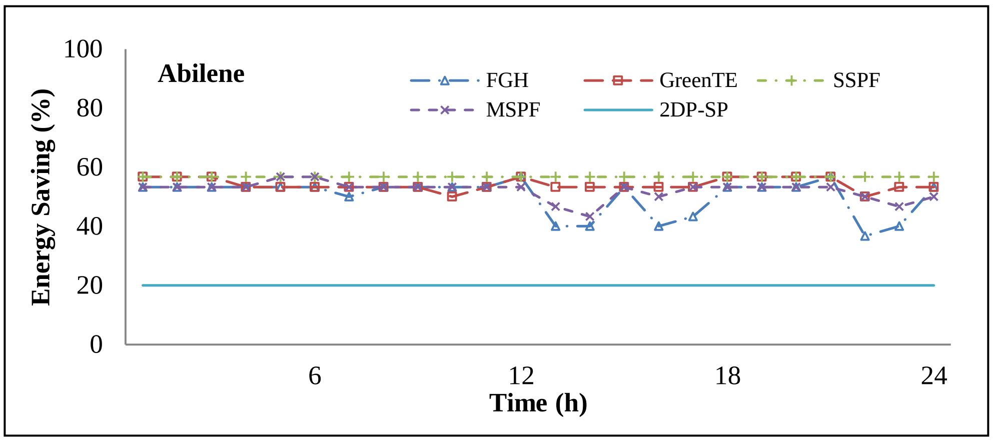


Figure 5.24 Comparison of power saving on Abilene

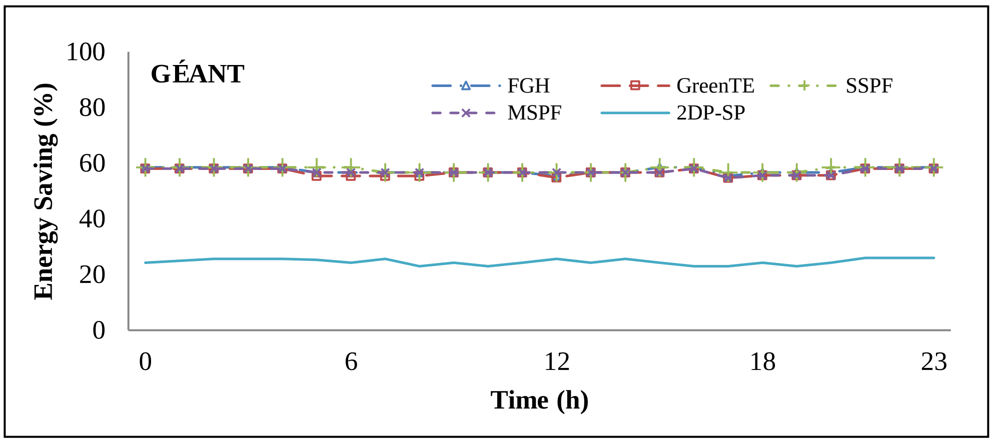


Figure 5.25 Comparison of power savings on GÉANT

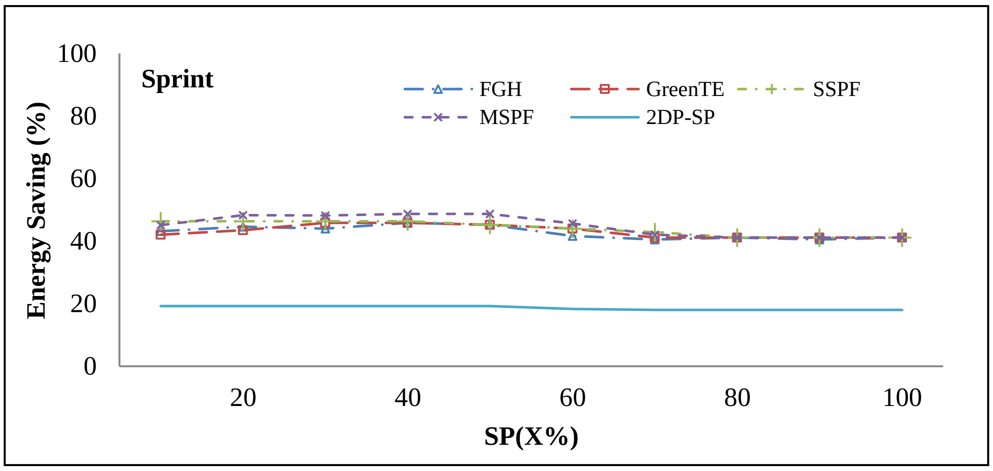


Figure 5.26 Comparison of power savings on Sprint

In contrast, as shown in Fig. 27, 28 and 29, 2DP-SP-1 is the best performer in term of RR. Note that, CDF of average RR is calculated for the traffics over 24 hours for Abilene and GÉANT, and ten SP(X%) for Sprint. As shown in the figures, 94%, 37% and 52% (65%, 22% and 37%) of paths produced by 2DP-SP-1 (MSPF) for Abilene, GÉANT and Sprint have paths with reliability at least 0.8, 0.9, 0.8 respectively; in this category, MSPF is the second best performer.

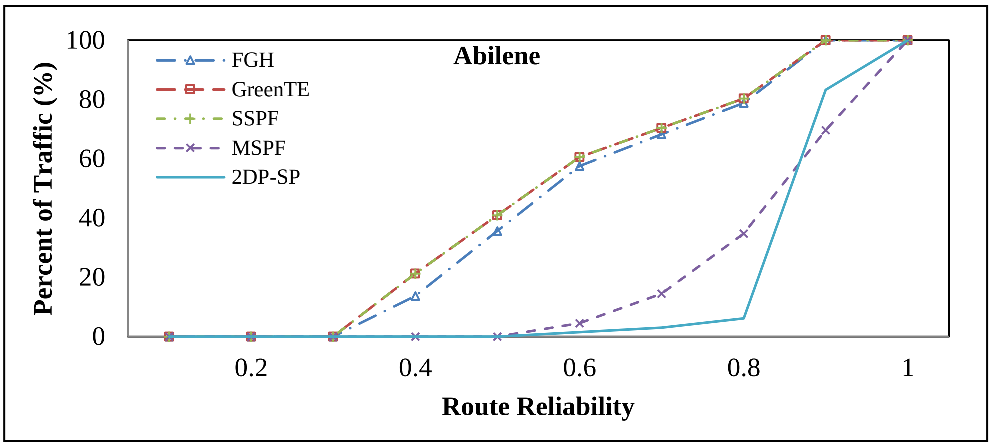


Figure 5.27 CDF of RR on Abilene

The figures show that GreenTE, FGH and SSPF consistently perform among the worst on the three networks in terms of RR. Specifically, on Abilene, only 40% of its routes have reliability larger than 0.6, none of which has reliability above 0.8. Note that GreenTE and SSPF have the same worst CDF; FGH is slightly better. For GÉANT, FGH and SSPF are the worst and second worst, where 65.92% and 62.8% of traffics have reliability no higher than 60%. For Sprint, 67.72% and 69.92% of routes produced by FGH and GreenTE have reliability no larger than 60%. 2DP-SP-1 outperforms MSPF because the former enforces each demand to be routed via possible disjoint paths, which has significantly higher RR as compared to the non-disjoint multiple paths used in the latter. Thus, one can expect that larger constraint Q_T will increase RR but reducing power saving because the constraint needs more switched-on links. Consequently, one can set lower (higher) value of Q_T to achieve higher power savings (RR), which makes 2DP-SP-1 the most flexible approach among the evaluated algorithms.

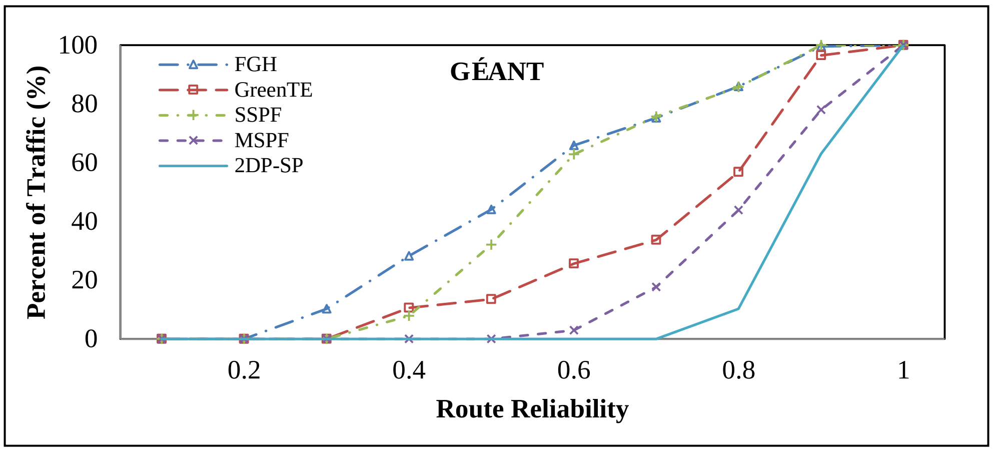


Figure 5.28 CDF of RR on GÉANT

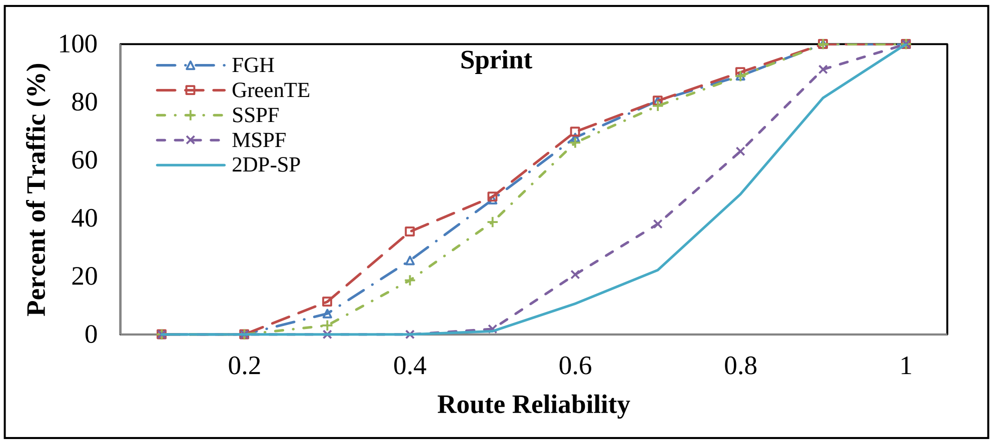


Figure 5.29 CDF of RR on Sprint

5.4.1.7 2DP-SP-1 versus Optimal Solution

Ideally, we aim to evaluate the effectiveness of our algorithms against the optimal solution for the EAR-2DP problem in term of power savings. Unfortunately, due to the exponential time complexity of the problem, as described in Section 3.2, we could not generate the optimal result for the simulated networks. As an alternative, we consider only two extreme cases of optimal solutions, *i.e.*, for (i) Q_T=Q_{max}, and (ii) Q_T=0%. For case (i), we assume each demand *d* can be routed through its DP_d, when such route exists in its original network, *i.e.*, all links are switched-on; recall that Q_{max} is the fraction/percentage of demands that have 2DP routes in the original network. Consequently, this case considers only those Q_{max}*|*D*| demands. For this case, we adopt the LP formulation of [44]. Specifically,

Minimize

$$\sum\nolimits_{(i,j)\in E} x_{ij} \tag{5.16}$$

Subject to

$$\sum\nolimits_{(i,j)\in E} Q_{ij}^{d} - \sum\nolimits_{(j,i)\in E} Q_{ji}^{d} = \begin{cases} 2, i = s_d \\ 0, i \neq s_d, t_d, \forall d \in D \\ -2, i = t_d \end{cases} \tag{5.17}$$

$$\sum\nolimits_{d} Q_{ij}^{d} b_d / 2 \leq U_T c_{ij} x_{ij}, \forall (i,j) \in E \tag{5.18}$$

Equation (5.16) is the objective of EAR-2DP for Q_T=Q_{max} representing the goal to minimize total number of powered-on links; x_{ij}=1 (x_{ij}=0) when link (*i*, *j*) is switched on (off). Equation (5.17) states flow conservation constraints for 2DP, which splits traffic volume equally on 2DP-L. The flow conservation constraint ensures that no flow is lost or created except at the source and destination. Note that each $Q_{ij}^{d}, \forall (i,j) \in E, d \in D$ is a binary value that is set to one when the flow of demand *d* is allocated to link (*i*, *j*). Equation (5.18) is a capacity constraint, which ensures that no link (*i*, *j*) carries more traffic flow than its available capacity, *i.e.*, U_T*c_{ij}. For extreme case (ii), *i.e.*, Q_T=0%, the EAR-2DP reduces to the ILP formulation in [24]; for brevity, we do not reproduce the formulation. The formulations for case (i) and (ii) are solved using CPLEX [97].

As shown in Table 5.5, we use four different networks for our comparisons; the column labeled "Network" shows the total number of links (*m*), demands (|*D*|), and Q_{max} of each network. We use the seven demands shown in Fig. 2.1, traffic matrix at time 24h for Abilene, traffic matrix at time 0h for GÉANT, and SP(30%) for Sprint. Each column marked "Traffic (%)" shows the percentage of traffic considered in both CPLEX and 2DP-SP-1; the demands are randomly selected, and for case (i) the percentage cannot be larger than its Q_{max}. Table 5.5 shows that, for smaller networks, *i.e.*, Fig. 2.1 and Abilene, 2DP-SP-1 produces optimal results for case (i); in the column marked "Switched-off Links", each number in the bracket is the optimal result. Note, to ensure a fair comparison, we require 2DP-SP-1 to distribute each demand equally through its 2DP. For case (ii), 2DP-SP-1 also produces optimal result for Fig. 2.1, but only 12% off from optimal for Abilene. However, for the larger networks, *i.e.*, GÉANT and Sprint, CPLEX [97] could not produce results for both cases even after running for more than 24 hours. Note that the authors in [45] could generate results using the ILP for case (ii) only when there are between 1% to 10% end-to-end traffic demands in their simulated networks. Therefore, for each of the two networks, we tested the effectiveness of 2DP-SP-1 using only 1% and 10% its total demands, *i.e.*, 5 and 51, and 27 and 265 demands for GÉANT and Sprint, respectively.

As shown in Table 5.5, CPLEX could generate result only for GÉANT (1%), in which 2DP-SP-1 generates results that are off only by 6.7% and 9.7% off from optimal for case (i) and (ii) respectively. Further, 2DP-SP-1 requires only less than 0.25% of the running time of CPLEX. Finally, while CPLEX fails to produce results, 2DP-SP-1 could save energy between 19.6% and 53% for case (i) and 39.3% and 60.1% for case (ii), while using CPU times between 8.2 and 748.6 seconds.

Table 5.5 2DP-SP-1 versus Optimal Solutions (U_T=0.5)

Network	**Q_T=Q_{max}**			**Q_T=0%**		
	Traffic (%)	**Switched-off links**	**Time (Second)**	**Traffic (%)**	**Switched-off links**	**Time (Second)**
Fig. 2.1 (*m*=10) $\lvert D\rvert$=7; Q_{max}=42.9%	42.9	3 (3)	0.01 (0.1)	100	3 (3)	0.01 (0.1)
Abilene (*m*=30) $\lvert D\rvert$=132; Q_{max}=83.3%	83.3	8 (8)	0.38 (4.0)	100	15 (17)	0.52 (11.0)
GÉANT (*m*=74) $\lvert D\rvert$=506; Q_{max}=100%	1	28 (30)	0.46 (185.4)	1	56 (62)	1.85 (3741.9)
	10	20 (NA)	8.2 (>24h)	10	44 (NA)	10.6 (>24h)
Sprint (*m*=168) $\lvert D\rvert$=2642; Q_{max}=37.4%	1	89 (NA)	18.5 (>24h)	1	101(NA)	20.3 (>24h)
	10	33 (NA)	647.2 (>24h)	10	66 (NA)	748.6 (>24h)

5.4.2 Evaluation for 2DP-SP-2

5.4.2.1 Power Saving of Nodes and Links

There are two types of nodes in the network: access nodes and transport nodes. Access nodes are sources and destinations of information and are connected to the ISP transport network while transport nodes are neither sources nor destinations of traffic. We assume that OC-192/STM-64 (10 Gbps) line card is used for all links and thus the maximum delay of single-hop is around 200 ns [118]. Similar to [88], we assume each cable in link (*i*, *j*) has the same power consumption p_{ij}=0.6kw and each node *v* consumes the same power p_v=3kw. The PS of each network is calculated as Equation (2.2).

Table 5.6 Power Saving by Shutting Down Nodes and Links

Power Saving	**R_Abilene**	**R_GÉANT**	**R_Sprint**
Off Transit Nodes	1	7	7
Off Links	7	40	87
Power Saving	13.3%	39.7%	28.5%

In this subsection, we use 2DP-SP-2 when both nodes and links can be powered off with R_Abilene, R_GÉANT and R_Sprint in Table 2.5. The power savings of these three topologies are shown in Table 5.6. We can find that running 2DP-SP-2 can switch off 1 node and 7 links on Abilene network, with power saving of up to 13.3%; for GÉANT, more than half of transit nodes (7/13) can be switched off, and the power saving reaches 39.7%; for Sprint, 7 transit nodes and 87 links can be powered off, achieving 28.5% power savings.

5.4.2.2 Effects on Link Utilization and Path Length

Intuitively, switching off nodes and links will affect the link utilization and maximum routing path length (MRPL) since fewer nodes and links are available to carry traffic. Fig. 5.30, 5.31 and 5.32 show the CDF of link utilization and path length for three topologies, *i.e.*, R_Abilene, R_GÉANT and R_Sprint respectively, using 2DP-SP-2 and two link disjoint paths on the original network with all switched on nodes and links, denoted by 2DP-SPF in the figures. .

From Fig. 5.30 (top), we see that 61% of links using 2DP-SPF have utilization no larger than 0.1, which is better as compared to 0% using 2DP-SP-2. However, while saving 13.3% power usage, 2DP-SP-2 also generates 95% of links with utilization between 0.1 and 0.2, which is better than 2DP-SPF that generates only 32% of links with utilization in the range. Further, 2DP-SP-2 generates MLU of 0.3, which is better as compared to 2DP-SPF with MLU reaching 0.5. For path length, shown in Fig. 5.30 (bottom), 5% of traffic demands have MRPL (7 hops) larger than the network diameter (6 hops) for 2DP-SP-2 while MRPL of all traffics for 2DP-SPF is no larger than network diameter.

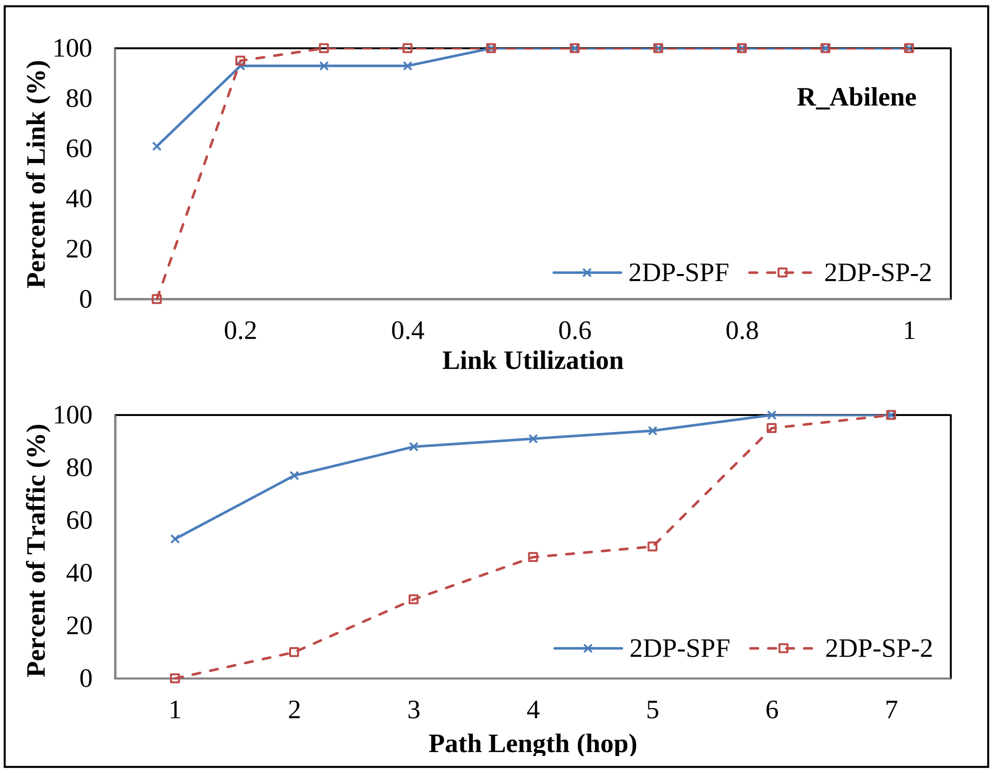


Figure 5.30 CDF of link utilization and path length on R_Abilene

In Fig. 5.31 (top), due to off-peak period, 56% of links using 2DP-SPF have utilization no larger than 0.1, which is better than the result generated by 2DP-SP-2, only at 11%. Different from R_Abilene network, 2DP-SPF generates MLU value only at 0.8 on R_GÉANT network, which is better than 2DP-SP-2, up to 1.0.

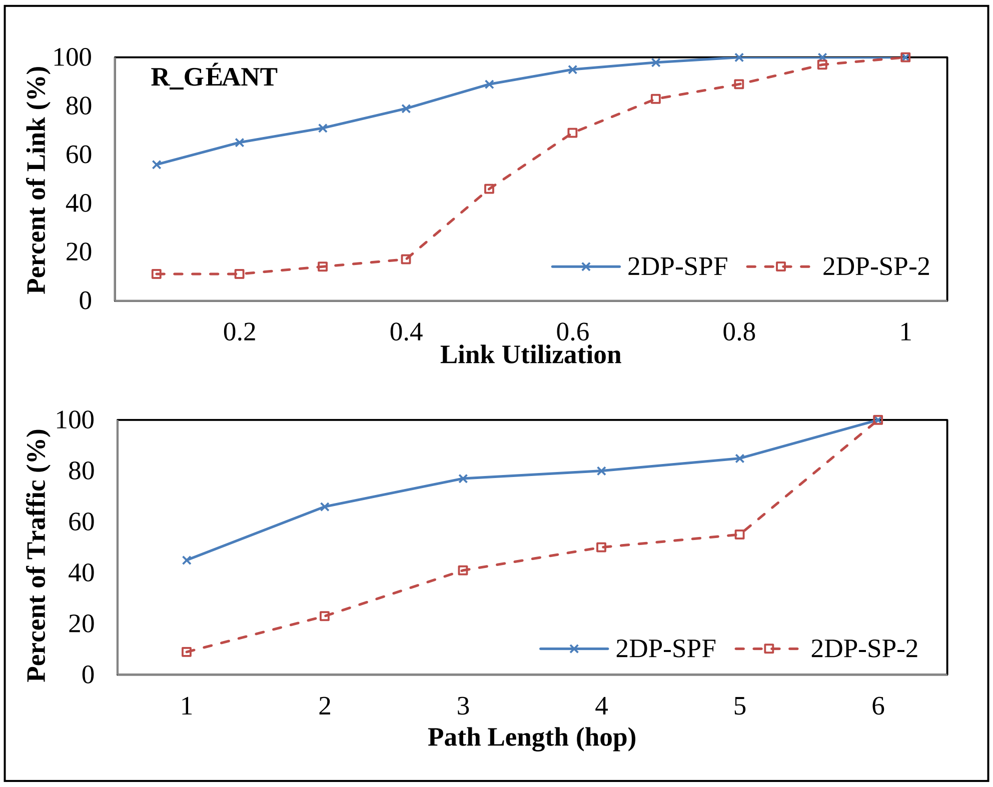


Figure 5.31 CDF of link utilization and path length on R_GÉANT

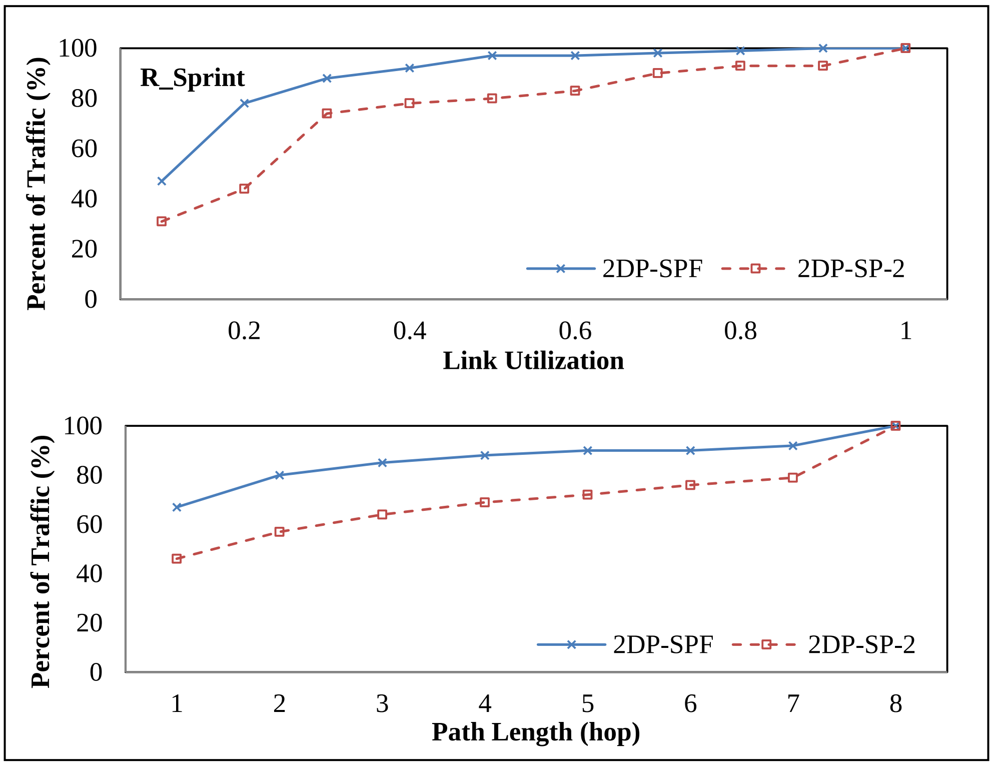


Figure 5.32 CDF of link utilization and path length on R_Sprint

As shown at Fig. 5.31 (bottom), for 2DP-SPF, 45% demands route its traffics through single hop paths, but only 9% demands use single hop routing for 2DP-SP-2.

However, running 2DP-SPF and 2DP-SP-2 obtains the same MRPL, which is less than the network diameter, at six hops.

For link utilization, Fig. 5.32 (top) shows that 2DP-SP-2 obtains 99% of links with utilization no larger than 0.8, which is only 6% higher than the value generated by 2DP-SPF. For routing path length, the results of running 2DP-SP-2 and 2DP-SPF on Sprint network are more closer than above two networks because the topology is larger; hence there are more paths for each (s_d, t_d) pair that have routing path length close to the shortest path. Further, running 2DP-SP-2 and 2DP-SPF obtains the same MRPL, which is equal to the network diameter (8 hops); see Fig. 5.32 (bottom).

5.5 Summary

In this chapter, we have presented a new energy-aware routing problem, EAR-2DP, and formally proved its NP-completeness. The two versions of the problem, EAR-2DP-1 and EAR-2DP-2, aim to reduce power expenditure by maximally switching off unnecessary nodes and/or links during off-peak periods such that the remaining powered on nodes and/or links are sufficient to route traffic demands subject to constraints on two parameters: link utilization and the minimum number of 2DP used to route traffic demands. We have proposed efficient and effective heuristic techniques, 2DP-SP-1 and 2DP-SP-2, to solve the EAR-2DP-1 and EAR-2DP-2 respectively, which are applicable for both 2DP-L and 2DP-N. Extensive simulations on both real and synthetic network topologies and traffic demands have shown their benefits in reducing power consumption while addressing critical network performances, such as link utilization, RR, and path length.

Chapter 6

Conclusions

6.1 Summary

We have suggested three methods of energy-aware TE, *i.e.*, SSPF, MSPF, and 2DP-SP, to minimize the power usage of IP networks while satisfying their required quality of service. Each method solves an energy-aware TE problem. We have used linear programming formulation to describe each problem that considers networks with bundled links, *i.e.*, $w_{ij} \geq 1$. Further, we have used both real and synthetic topologies and traffic demands to evaluate the effectiveness and efficiency of using our methods.

Specifically, we have proposed the following three problems and their solutions.

- We have proposed the single path green routing problem in which each traffic demand must be routed via a single path while guaranteeing MLU. We have proposed SSPF to address this problem. SSPF aims to switch off redundant cables in core routers using bundled links to reduce the power consumption significantly, while aiming to route each traffic demand via its feasible shortest path.

- We have formulated a green routing problem to route each traffic demand using multiple paths that must satisfy path length and MLU constraints. Specifically, we set the path length not longer than twice the length of the shortest path, or the diameter of the original network. We have described MSPF to address the problem. Our simulations show the efficiency of MSPF on the power saving under the two constraints.

- We have proposed a green routing problem, EAR-2DP, to route each traffic demand via its 2DP. The problem considers MLU and the fraction of using 2DP constraints. Furthermore, we have provided the NP-complete proof of the problem. We have described two versions of solution, *i.e.*, 2DP-SP-1 and 2DP-SP-2, to solve EAR-2DP. The solution provides a good trade-off between fault-tolerance and power consumption. Our simulation shows that 2DP-SP-1 and 2DP-SP-2 are effective in reducing power usage.

6.2 Future work

The presented results are promising. However, further efforts on the software and hardware design on the network protocol implementation for the proposed techniques are still needed. In particular, network devices have to support stand-by modes and power-on primitives, and network protocols have to consider the variation in the structure of the network derived from variation of the powered on devices. In our future work, we will consider these open challenges.

In Chapter 5, we have studied the impact of switching off cables using five recent energy-aware routing mechanisms, namely FGH [15], GreenTE [16], SSPF [40], MSPF [45], and 2DP-SP [106], on RR measure that computes the reliability of each selected path to route traffic in an event of link failures. The results in Section 5.4.1.6 show the significant effects of the green algorithms on a network's RR. We believe the results hold for other energy-aware routing algorithms, and thus it is imperative to incorporate reliability measures in green routing protocols. Furthermore, in the future, we will analyze the impact of green routing on terminal reliability (TR) - the probability of obtaining at least one source (s_d) to terminal (t_d) operational path for each demand d. Note that TR gives the probability of using route restoration in case of link failures. We will discuss whether green routing algorithms significantly reduce a network's TR, and thus significantly reduce the possibility of any path restoration in the event of link failures. For our future work, we will propose the reliable energy-aware-routing (R-EAR) problem, which aims to switch-off as many cables as possible to maximally save energy while maintaining required levels of TR or RR and MLU of the network. Further, we will propose a reliable Green-Routing algorithm to solve R-EAR and analyze its time complexity.

Bibliography

[1] C. Bianco, F. Cucchietti, G. Griffa, "Energy Consumption Trends in the Next Generation Access Network - A Telco Perspective". In *INTELEC*, Rome, Italy, 2007, pp. 737-742.

[2] S. N. Roy, "Energy Logic: A Road Map to Reducing Energy Consumption in Telecommunications Networks". In *INTELEC*, San Diego, USA, 2008, pp. 1-9.

[3] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy efficiency in the future internet: a survey of existing approaches and trends in energy-aware fixed network infrastructures". In *IEEE Communications Surveys & Tutorials*, vol. 13, no. 2, 2011, pp. 223-244.

[4] N. Horowitz, Hardy G, Swofford J, Watters P, Driscoll D, Dayem K, "Small Network Equipment Energy Consumption in U.S. Homes". In *NRDC Issue Paper*, June 2013.

[5] Neilson, D.T., "Photonics for switching and routing". In *IEEE Journal of Selected Topics in Quantum Electronics (JSTQE)*, vol. 12, no. 4, 2006, pp. 669-678.

[6] J. Guichard, F. L. Faucheur, and J.-P. Vasseur, Definitive MPLS Network Designs. In *Cisco Press*, 2005.

[7] Global e-Sustainibility Initiative (GeSI), SMART 2020: Enabling the Low Carbon Economy in the Information Age. [Available]: http://www. theclimategroup.org/assets/resources/publications/ Smart2020Report.pdf.

[8] European Commission DG INFSO, "Impacts of Information and Communication Technologies on Energy Efficiency". In Final report, Tender No. CPP 16 A-2007/2007/s 68-082361, 2008.

[9] U.S. Energy Information Administration (EIA), Official Energy Statistics from the U.S. Government. [Available]: http://www.eia.coe.gov.

[10] W.V. Heddeghem, M. D. Groote, W. Vereecken, D. Colle, M. Pickavet, and P. Demeester, "Energy efficiency in telecommunication networks". In *Optical Network Design and Modeling (ONDM)*, Kyoto, Japan, 2010, pp. 1-6.

[11] M. Menth, R. Martin, J. Charzinski, “Capacity overprovisioning for networks with resilience requirements”. In *ACM SIGCOMM*, New York, USA, 2006, pp. 87-98.

[12] M. Gupta, S. Singh, “Greening of the Internet”. In *ACM SIGCOMM*, Karlsruhe, Germany, 2003, pp. 19-26.

[13] R. Hays, “Active/idle Toggling with 0base-x for Energy Efficient Ethernet”. In *IEEE P802.3az Energy Efficient Task Force*, Atlanta, GA, USA, 2007.

[14] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright, “Power Awareness in Network Design and Routing”. In *IEEE INFOCOM*, Phoenix, AX, USA, 2008, pp. 457-465.

[15] W. Fisher, M. Suchara, and J. Rexford, “Greening Backbone Networks: Reducing Energy Consumption by Shutting off Cables in Bundled Links”. In *Green Networking*, New Delhi, India, 2010, pp. 29-34.

[16] M. Zhang, C. Yi, B. Liu, and B. Zhang, “GreenTE: Power-Aware Traffic Engineering”. In *ICNP*, Beijing, China, 2010, pp. 21-30.

[17] R. Doverspike, K. K. Ramakrishnan, and C. Chase, Structural Overview of ISP Networks. In *Guide to Reliable Internet Services and Applications (C. Kalmanek, S. Misra, and R. Yang, eds.)*, Springer London, 2010.

[18] IEEE, IEEE Standard 802.1 AX: Link Aggregation, 2008.

[19] J. Rajahalme, A. Conta, B. Carpenter and S. Deering, IPv6 Flow Label Specification. In *IETF RFC* 3697, 2004.

[20] J. Moy, OSPF Version 2. In *IETF RFC* 232*8*, 1998.

[21] D. Oran, OSI IS-IS Intra-Domain Routing Protocol. In *IETF RFC* 1142, 1990.

[22] E. W. Dijkstra, “A Note on Two Problems in Connexion with Graphs”. In *Numerische Mathematik*, vol. 1, no. 1, 1959, pp. 269-271.

[23] U. Ranadive and D. Medhi,“ Some Observations on the Effect of Route Fluctuation and Network Link Failure on TCP”. In *IEEE International Conference on Computer Communications and Networks (ICCCN)*, Scottsdale,

AZ, 2001, pp. 460-467.

[24] M. Pioro and D. Medhi, Routing, Flow, and Capacity Design in Communication and Computer Networks. In *ELSEVIER*, 2004.

[25] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, Network Flows: Theory, Algorithms, and Applications. In *Prentice-Hall, Englewood Cliffs,* New Jersey, USA, 1993.

[26] F. H. Moss, P. M. Merlin, “Some Theoretical Results in Multiple Path Definition in Networks”. In Networks, vol. 11, no. 4, 1981, pp. 401-411.

[27] W. Hoffman and R. Pavley, “A Method for the Solution of the Nth Best Problem”. In *J. of ACM*, vol. 6, no. 4, 1959, pp. 506-514.

[28] R. Bellman and R. Kalaba, “On k^{th} Best Policies”. In *J. of SIAM*, vol. 8, no. 4, 1960, pp. 582-588.

[29] M. Sakarovitch, “The *k* Shortest Chains in a Graph”. In *Transportation Research*, vol. 2, no. 1, 1968, pp. 1-11.

[30] JY. Yen, “Finding the *k* Shortest Loopless Paths in a Network”. In *Management Science*, vol. 17, no. 11, 1971, pp. 712-716.

[31] J. W. Suurballe and R. E. Tarjan. “A Quick Method for Finding Shortest Pairs of Disjoints Paths”. In *Networks*, vol. 14, no. 2, 1984, pp. 325-336.

[32] R. Bhandari, “Optimal Diverse Routing in Telecommincation Fiber Networks”. In *IEEE INFOCOM*, Toronto, Canada, vol. 3, 1994, pp. 1498-1508.

[33] K. Xiong, Z. D. Qiu, Y. Guo, and H. Zhang. “Multi-Constrained Shortest Disjoint Paths for Reliable QoS Routing”. In *ETRI Journal*, vol. 31, no. 5, 2009, pp. 534-544.

[34] K. Kar, M Kodialam, T. V. Lakshman, “Routing Restorable Bandwidth Guaranteed Connections using Maximum 2-Route Flows”. In *IEEE INFOCOM*, New York, USA, vol. 1, 2002, pp. 113-121.

[35] M. Kodialam, T. V. Lakshman, "Dynamic Routing of Bandwidth Guaranteed Tunnels with Restoration". In *IEEE INFOCOM*, Tel Aviv, Israel, vol. 2, 2000, pp. 902-911.

[36] M. Kodialam, T. V. Lakshman, "Restorable Dynamic Quality of Service Routing". In *IEEE Communications Magazine*, vol. 40, no. 6, 2002, pp.72–81.

[37] W. Liang, "Robust routing in wide-area WDM networks". In *Int'l Parallel and Distributed Processing Symposium*, San Francisco, USA, 2001.

[38] D. O. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "Overview and principles of Internet traffic engineering". In *RFC*3272, 2002.

[39] Bernard Fortz, Jennifer Rexford, and Mikkel Thorup, "Traffic engineering with traditional IP routing protocols". In *IEEE Communication Magazine*, vol. 40, no. 10, 2002, pp. 118-124.

[40] G. Lin, S. Soh, K. Chin and M. Lazarescu, "Efficient Heuristics for Energy-Aware Routing in Networks with Bundled Links". In *Computer Networks*, vol.57, no. 8, 2013, pp.1774-1788.

[41] S. Kandula, D. Katabi, B. Davie and A. Charny, "Walking the Tightrope: Responsive yet Stable Traffic Engineering". In *ACM SIGCOMM*, PA, USA, 2005, pp. 253-264.

[42] A Nucci, B. Schroeder, S. Bhattacharyya, N. Taft and C. Diot, "IGP Link Weight Assignment for Transient Link Failures". In *International Teletraffic Congress (ITC)*, Berlin, Germany, 2003, pp. 321-330.

[43] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS Weights in a Changing World". In *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 4, 2002, pp. 756-767.

[44] M. Suchara, D. Xu, R. Doverspike, D. Johnson and J. Rexford, "Network Architecture for Joint Failure Recovery and Traffic Engineering". In *ACM SIGMETRICS*, San Jose, USA, 2011, pp. 97-108.

[45] G. Lin, S. Soh, K. Chin and M. Lazarescu, "Power-Aware Routing in Networks

with Delay and Link Utilization Constraints". In *IEEE LCN*, Clearwater Beach, Florida, USA, 2012, pp. 272-275.

[46] Telemark Survey 2006. [Available]: http://www.telemarkservices.com/.

[47] W. Zhang, J. Tang, C. Wang, and S. de Soysa, "Reliable Adaptive Multipath Provisioning with Bandwidth and Differential Delay Constraints". In *IEEE INFOCOM*, San Diego, USA, 2010, pp. 1-9.

[48] Y. Wang, H. Wang, A. Mahimkar, R. Alimi, Y. Zhang, L. Qiu, and Y. R. Yang, "R3: Resilient Routing Reconfiguration", in *ACM SIGCOMM*, New Delhi, India, 2010, pp. 291-302.

[49] Y. Bejerano, Y. Breitbart, A. Orda, R. Rastogi and A. Sprintson, "Algorithms for Computing QoS Paths with Restoration". In *IEEE/ACM Transactions on Networking*, vol. 13, no. 3, 2003, pp. 648-661.

[50] Y. Guo, F. Kuipers and P. V. Meeghem. "Link-Disjoint Paths for Reliable QoS Routing". In *Int. J. Communication System*, vol. 16, no. 9, 2003, pp. 779-798.

[51] R. Izmailov and D. Niculescu, "Flow Splitting Approach for Path Provisioning and Path Protection Problems". In *Merging Optical and IP Technologies workshops on High Performance Switching and Routing*, Kobe, Japan, 2002, pp. 93-98.

[52] D. Thaler, and C. Hopps, "Multipath Issues in Unicast and Multicast Next-Hop Selection". In *IETF RFC* 2991, 2000.

[53] J. Y. He and J. Rexford, "Towards Internet-Wide Multipath Routing". In *IEEE Network*, vol. 22, no. 2, 2008, pp.16-21.

[54] S. Vutukury, and J. J. Garcia-Luna-Aceves, "A Traffic Engineering Approach Based on Minimum-Delay Routing". In *Proc. International Conference on Computer Communications and Networks (ICCCN)*, Las Vegas, USA, 2000, pp. 42-47.

[55] U. Javed, M. Suchara, J. He, and J. Rexford, "Multipath Protocol for

Delay-Sensitive Traffic". In *COMSNETS*, Bangalore, Indian, 2009, pp. 1-8.

[56] F. Barahona, "Network Design Using Cut Inequalities". In *SIAM Journal on Optimization*, vol. 6, no. 3, 1996, pp. 823-837.

[57] A. P. Bianzino, C. Chaudet, D. Rossi, and J. Rougier, "A Survey of Green Networking Research". In *IEEE Communications Surveys & Tutorials*, vol. 14, no. 1, 2010, pp. 3-20.

[58] G. Da Costa, J.-P. Gelas, Y. Georgiou, L. Lefevre, A.-C. Orgerie, J.-M. Pierson, O. Richard, and K. Sharma, "The GREEN-NET Framework: Energy Efficiency in Large Scale Distributed Systems". In *High Performance Power Aware Computing Workshop (HPPAC) in conjunction with IPDPS*, Rome, Italy, 2009, pp. 1-8.

[59] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall, "Reducing Network Energy Consumption via Sleeping and Rate-Adaptation". In *USENIX Symposium on Networked Systems Design and Implementation (NDSI)*, San Francisco, California, USA, 2008, pp. 323-336.

[60] C. Gunaratne, K. Christensen et al., "Reducing the Energy Consumption of Ethernet with Adaptive Link Rate (ALR)". In *IEEE Transactions on Computers*, vol. 57, no. 4, 2008, pp. 448-461.

[61] M. Baldi and Y. Ofek, "Time for a 'Greener' Internet". In *International Workshop on Green Communications (Green-Comm) in conjunction with the IEEE International Conference on Communications*, Dresden, Germany, 2009, pp. 1-6.

[62] B. Sans`o and H. Mellah, "On Reliability, Performance and Internet Power Consumption". In *International Workshop on Design of Reliable Communication Networks (DRCN)*, Washington, D.C., USA, 2009, pp. 259-264.

[63] S. Nanda and T.-C. Chiueh, "A Survey on Virtualization Technologies". In *Technical Report TR179*, Department of Computer Science, SUNY at Stony

Brook, 2005.

[64] N. M. Kabir Chowdhury and R. Boutaba, “A Survey of Network Virtualization”. In *Technical Report* CS-2008-25, University of Waterloo, Oct. 2008.

[65] L. A. Barroso and U. Holzle, “The Case for Energy-Proportional Computing”. In *IEEE Computer,* vol. 40, no. 12, 2007, pp. 33-37.

[66] M. Weiser, B. Welch, A. Demers, and S. Shenker, “Scheduling for Reduced CPU Energy”. In *USENIX Conference on Operating Systems Design and Implementation (OSDI)*, Monterey, California, 1994, pp. 13-23.

[67] C. Gunaratne, K. Christensen, and S. W. Suen, “Ethernet Adaptive Link Rate (ALR): Analysis of a buffer threshold policy”. In *IEEE Global Communications Conference (GLOBECOM)*, San Francisco, California, USA, Nov. 2006, pp. 1-6.

[68] M. Subramanian, Network Management - Principles and Practice. In *Reading,* MA, USA: Addison-Wesley, 2000.

[69] B. Addis, A. Capone, G. Carello, L. G. Gianoli, and Brunilde, “Energy Management Through Optimized Routing and Device Powering for Greener Communication Networks”. In *IEEE/ACM Transactions on Networking*, vol. 22, no. 1, 2014, pp. 313-325.

[70] K. Christensen, C. Gunaratne, B. Nordman, and A. George, “The Next Frontier for Communications Networks: Power Management”. In *Computer Communications*, vol. 27, no. 18, 2004, pp. 1758-1770.

[71] K. Christensen, C. Gunaratne, B. Nordman, “Managing Energy Consumption Costs in Desktop PCs and LAN Switches with Proxying, Split TCP Connections, and Scaling of Link Speed”. In *Internat. Journal of Network Management*, vol. 15, no. 5, 2005, pp. 297-310.

[72] Ada M. Alvarez, Jose Luis Gon ´azlez-Velarde, and Karim De-Alba. “Grasp embedded scatter search for the multicommodity capacitated network design problem”. In *Journal of Heuristics*, vol. 11, no. 3, 2005, pp. 233–257.

[73] S. Even, A. Itai, and A. Shamir, “On the Complexity of Time Table and Multi-commodity Flow Problems”. In *IEEE Foundation of Computer Science (FOCS)*, Berkeley, CA, USA, 1975, pp. 184-193.

[74] D. Berger, B. Gendron, J.-Y. Potvin, S. Raghavan, P. Soriano, “TabuSearch for a Network Loading Problem with Multiple Facilities”. In *Journal of Heuristics*, vol. 6, no. 2, 2000, pp. 253–267.

[75] B. Gendron, J.-Y. Potvin, P. Soriano, “Diversification Strategies in Local Search for a Non-bifurcated Network Loading Problem”. In *European Journal of Operational Research*, vol. 142, no. 2, 2002, pp. 231-241.

[76] Y.K. Agarwal, “Design of Capacitated Multi-commodity Networks with Multiple Facilities”. In *Operations Research*, vol. 50, no. 2, 2002, pp. 333-344.

[77] A. Atamturk, “On Capacitated Network Design Cut-set Polyhedral”. In *Mathematical Programming*, vol. 92, 2002, pp. 425-437.

[78] D. Bienstock, S. Chopra, O. Günlük, C. Tsai, “Minimum Cost Capacity Installation for Multicommodity Network Flows”. In *Mathematical Programming*, vol. 81, no. 2, 1998, pp. 177–199.

[79] A. Frangioni, B. Gendron, “0-1 Reformulations of the Multi-commodity Capacitated Network Design Problem”. In *Discrete Applied Mathematics*, vol. 157, no. 6, 2009, pp. 1229-1241.

[80] M. Gupta, S. Singh, “Energy Conservation with Low Power Modes in Ethernet LAN Environments”. In *IEEE INFOCOM (mini symposium)*, Anchorage, Alaska, USA, 2007, pp. 2451-2455.

[81] N. Vasic and D. Kostic, “Energy-Aware Traffic Engineering”. In *ACM SIGCOMM*, New Delhi, India, 2010, pp. 169-178.

[82] E. Gelenbe and T. Mahmoodi. “Energy-aware Routing in the Cognitive Packet Network”. In *Performance Evaluation*, vol. 68, no. 4, 2011, pp. 338-346.

[83] A. P. Bianzino, L. Chiaraviglio and M. Mellia, “GRiDA: a Green Distributed Algorithm for Backbone Networks”. In *IEEE Online Green Communications Conference*, New York, USA, 2011, pp. 113-119.

[84] S. S. W. Lee, P. K. Tseng and A. Chen. “Link Weight Assignment and Loop-free Routing Table Update for Link State Routing Protocols in Energy-aware Internet”. In *Future Generation Computer Systems*, vol. 28, no. 2, 2012, pp. 437-445.

[85] N. Vasic, P. Bhurat, D. Novakovic, M. Canini, S. Shekhar, D. Kostic, “Identifying and Using Energy-Critical Paths”. In *ACM CoNEXT*, Tokyo, Japan, no. 18, 2011, pp. 1-12.

[86] Y.M. Kim, E. –J, Lee, H.-S.Park, J.-K. Choi, H.-S.Park, “Ant Colony Based Self-Adaptive Energy Saving Routing for Energy Efficient Internet”. In *Computer Networks*, vol. 56, no. 10, 2012, pp. 2343-2354.

[87] A. Coiro, M. Listanti, A. Valenti, F. Matera, “Energy-Aware Traffic Engineering: A Routing-Based Distributed Solution for Connection-Oriented IP Networks”. In *Computer Networks*, vol. 57, no. 9, 2013, pp. 2004-2020.

[88] L. Chiaraviglio, M. Mellia, and F. Neri, “Minimizing ISP Network Energy Cost: Formulation and Solutions”. In *IEEE/ACM Transactions on Networking*, vol. 20, no. 2, 2012, pp. 463-476.

[89] E. Amaldi, A. Capone, L.G. Gianoli, L. Mascetti, “A MILP-based Heuristic for Energy-Aware Traffic Engineering with Shortest Path Routing”. In *International Network Optimization Conference (INOC)*, Hamburg, Germany, 2011, pp. 464-477.

[90] F. Cuomo, A.Cianfrani, M. Polverini and D.Mangione, “Network Pruning for Energy Saving in the Internet”. In *Computer Networks*, vol. 56, no. 10, 2012, pp. 2355-2367.

[91] A. Cianfrani, V. Eramo, M. Listanti, M. Polverini and A. V. Vasilakos, “An

OSPF-Integrated Routing Strategy for QoS-Aware Energy Saving in IP Backbone Networks". In *IEEE Transactions on Network and Service Management*, vol. 9, no.3, 2012, pp. 254-267.

[92] E. Amaldi, A. Capone, L. Gianoli, L. Mascetti, "Energy Management in IP Traffic Engineering with Shortest Path Routing". In *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Lucca, Italy, 2011, pp. 1-6.

[93] H. Takeshita, N. Yamanaka, S. Okamoto, S. Shimizu, S. Gao, "Energy Efficient Network Design Tool for Green IP/Ethernet Networks". In *Optical Switching and Networking*, vol. 9, no. 3, 2012, pp. 264-270.

[94] R. G. Garroppo, S. Giordano, G. Nencioni, M. Pagano, M. G. Scutella. Energy Saving Heuristics in Backbone Networks, In *Proc. SustainIT*, Pisa, Italy, 2012, pp. 1-9.

[95] R. G. Garroppo, S. Giordano, G. Nencioni, M. Pagano, M. G. Scutella. Network Power Management: Models and Heuristics Approaches, In *Proc. IEEE Globecom*, Houston, TX, USA, 2011, pp. 1-5.

[96] L. Liu, B. Ramamurthy. A Dynamic Local Method for Bandwidth Adaptation in Bundle Links to Conserve Energy in Core Networks, In *Proc. IEEE International Conference on Advanced Networks and Telecommunication System (ANTS)*, Bangalore, India, 2011, pp. 1-6.

[97] AMPL/CPLEX. [Available]: http://www.ampl.com/DOWNLOADS/.

[98] D. Katz, K. Kompella, and D. Yeung, Traffic Engineering (TE) Extensions to OSPF Version 2. In *IETF RFC* 3630, 2003.

[99] Yin Zhang's Abilene Traffic Matrix. [Available]: http://www.cs.utexas.edu/~yzhang/research/AbileneTM.

[100] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing Public Introdomain Traffic Matrices to the Research Community". In *ACM SIGCOMM Computer Communication Review*, Pisa, Italy, vol. 36, no. 1, 2006, pp. 83-86.

[101] N. Spring, R. Mahajan and D. Wetherall, "Measuring ISP Topologies with Rocketfuel". In *IEEE/ACM Transactions on Network*, vol. 12, no. 1, 2004, pp. 2-16.

[102] GT-ITM: Georgia Tech Internetwork Topology Models (software). [Available]: http://www.cc.gatech.edu/projects/gtitm.

[103] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True, "Deriving Traffic Demands for Operational IP Networks: Methodology and Experience". In *IEEE/ACM Transactions on Networking*, vol. 9, no. 3, 2001, pp. 265-279.

[104] A. Soule, A. Nucci, E. Leonardi, R. Cruz, and N. Taft, "How to Identify and Estimate the Top Largest Traffic Matrix Elements in a Changing Environment". In *ACM SIGMETRICS*, New York, USA, 2004, pp.73-84.

[105] A. Nucci, A. Sridharan and N. Taft, "The Problem of Synthetically Generating IP Traffic Matrices: Initial Recommendations". In *ACM SIGCOMM*, PA, USA, vol. 35, no. 3, 2005, pp. 19-32.

[106] G. Lin, S. Soh, K. Chin and M. Lazarescu, "Energy-Aware Two Link- Disjoint Paths Routing". In *IEEE International Conference on High Performance Switching and Routing (HPSR)*, Taipei, Taiwan, 2013, pp. 103-108.

[107] G. Lin, S. Soh, K. Chin and M. Lazarescu, "On the Effects of Energy-Aware Traffic Engineering on Routing Reliability". In *IEEE Asia Pacific Conference on Communications (APCC)*, Bali, Indonesia, 2013, pp. 69-74.

[108] GNU Linear Programming Kit (GLPK). [Available]: http://www.gnu.org/s/glpk/.

[109] S. Soh, S. Rai, "CAREL: Computer Aided Reliablity Evaluator for Distributed Computing Networks". In *IEEE Transactions on Parallel and Distributed Systems*, vol. 2, no. 2, 1991, pp. 199-213.

[110] F.A. Kuipers, T. Korkmaz, M. Krunz and P. Van Mieghem, "An Overview of Constraint-Based Path Selection Algorithms for QoS Routing". In *IEEE*

Communications Magzine, vol. 40, no. 12, 2002, pp. 50-55.

[111] G. Lin, S. Soh, K. Chin and M. Lazarescu, “Reliable Green Routing by Two Disjoint Paths”. In *IEEE International Conference on Communications (ICC)*, Sydney, Australia, (To be presented in June 2014).

[112] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to algorithms, Second Edition. In *MIT Press*, 2005, pp. 1014-1016.

[113] A. Srinivas and E. Modiano, “Minimum Energy Disjoint Path Routing in Wireless Ad-Hoc Networks”. In *ACM/IEEE Int’l Conf. Mobile Computing and Networking (MOBICOM)*, San Diego, USA, 2003, pp. 122-133.

[114] Cisco, Cisco 7600 Series Route Switch Processor 720. [Available]: http://www.cisco.com/en/US/prod/collateral/routers/ps368/product_data_sheet 0900aecd8057f3b6.html.

[115] JUNOS: MPLS fast reroute network operations guide. In *Juniper networks, Inc.*, 2013.

[116] E. Osborne and A. Simha, Traffic Engineering with MPLS. In *Cisco Press*, 2002.

[117] Cisco, Power Management for the Cisco 12000 Series Router. [Available]: http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/12spower.html.

[118] “Spirent Signal Delay Network Emulator – DG/OTU/CPRI”. In *Datasheet of Spirent Communications, Inc.*, 2010.

[119] M. L. Shooman, Reliability of Computer Systems and Networks. In *WILEY*, 2002.

[120] G. Lin, S. Soh, K. Chin and M. Lazarescu, “Power-Aware Routing in Networks with Quality of Services Constraints”. In *Transactions on Emerging Telecommunications Technologies*, (In print, 2014).

[121] N Megiddo, “On the complexity of linear programming”. In *Advances in economic theory: Fifth world congress*, T. Bewley, ed., Cambridge University

Press, Cambridge, 1987, pp. 225–268.