Robust routing optimization in resilient networks : Polyhedral model and complexity issues

Mateusz Zotkiewicz

Thèse de doctorat de Télécom SudParis dans le cadre de l'école doctorale S&I en co-accréditation avec l'Universite d'Evry-Val d'Essonne

Spécialité:
Informatique

Par:
M. Mateusz Żotkiewicz

These présentée pour l'obtention du diplôme de Docteur de Télécom SudParis

Robust routing optimization in resilient networks — polyhedral model and complexity issues

Soutenue le 4 janvier 2011 devant le jury composé de:

M. Andrzej Dąbrowski – Président – École polytechnique de Varsovie, Pologne
M. Adam Ouorou – Rapporteur – Orange Labs, France
M. Włodzimierz Ogryczak – Rapporteur – École polytechnique de Varsovie, Pologne
M. Wojciech Burakowski – Examinateur – École polytechnique de Varsovie, Pologne
M. Zbigniew Kotulski – Examinateur – École polytechnique de Varsovie, Pologne
M. Eugeniusz Toczyłowski – Examinateur – École polytechnique de Varsovie, Pologne
M. Walid Ben-Ameur – Directeur de thèse – Telecom SudParis, France
M. Michał Pióro – Directeur de thèse – École polytechnique de Varsovie, Pologne

Thèse No 2011TELE0001

Abstract

In the thesis robust routing design problems in resilient networks are considered. In the first part computational complexity of such problems are discussed. The following cases are considered:

- path protection and path restoration
- failure-dependent and failure-independent restoration
- cases with and without stub-release
- single-link failures and multiple-link failures (shared risk link group)
- non-bifurcated (unsplittable) flows and bifurcated flows

For each of the related optimization cases a mixed-integer (in the non-bifurcated cases) or linear programming formulation (in all bifurcated cases) is presented, and their computational complexity is investigated. For the $\mathcal{NP}$-hard cases original $\mathcal{NP}$-hardness proofs are provided, while for the polynomial cases compact linear programming formulations (which prove the polynomiality in the question) are discussed. Moreover, pricing problems related to each of the considered $\mathcal{NP}$-hard problems are discussed.

The second part of the thesis deals with various routing strategies in networks where the uncertainty issues are modeled using the polyhedral model. In such networks two extrema are possible. The simplest in terms of implementation, and simultaneously the least effective strategy, is the robust stable routing. On the other hand, the most effective strategy, i.e., the dynamic routing, is virtually impossible to implement in real world networks. Therefore, the major aim of this part of the thesis is to present novel routing strategies that merge the simplicity of the robust stable routing with the efficiency of the dynamic routing.

Acknowledgement

First of all, I would like to thank my supervisors Michał Pióro and Walid Ben-Ameur for suggesting unchallenged problems that are solved in this thesis and for their help in approaching those problems. Secondly, I would like to thank Artur Tomaszewski without whom one of the problems presented in the thesis would still be unsolved. Many thanks to Mateusz Dzida and Michał Zagożdżon for their time and help they provided in the initial stages of my PhD studies, and to Mariusz Mycek for his help in the final stages of my studies. Finally, good words to Fernando Solano for upgrading some graphics in the thesis, and to Susann Schrenk for her help with an extended abstract.

Table of Contents

Table of Contents **vii**

1 Introduction **1**

I Complexity Issues **9**

2 Notation **10**

3 Path Protection Problems **12**

3.1 HS: Hot-Standby 12

3.2 PD: Path Diversity 13

3.2.1 $\mathcal{NP}$-hardness proof 18

3.2.2 Non-bifurcated PD 20

3.2.3 Path generation 20

4 Path Restoration Problems with Failure-Independent Restoration **24**

4.1 FI-nSR: no Stub-Release 24

4.1.1 Complexity overview 25

4.1.2 $\mathcal{NP}$-hardness proof (many demands, IP) 28

4.1.3 $\mathcal{NP}$-hardness proof (one demand, LP) 29

4.1.4 Complexity of approximation schemes 31

4.1.5 Path generation 32

4.2 FI-SR: Stub-Release 33

4.2.1 Complexity overview 34

4.2.2 Bifurcated flows 34

4.2.3 Non-bifurcated flows . . . 35
4.2.4 Generalizations for non-bifurcated flows . . . 39
4.2.5 Path generation . . . 44

5 Path Restoration Problems with Failure-Dependent Restoration 45
5.1 FD-nSR: no Stub-Release . . . 45
5.2 FD-SR: Stub-Release . . . 48

6 Conclusions 50

II Polyhedral model 53

7 Notation and problem formulations 54
7.1 Basic problems . . . 55
7.1.1 Robust routing . . . 55
7.1.2 No sharing routing . . . 56
7.1.3 Dynamic routing . . . 57
7.2 Test cases . . . 58
7.2.1 Hose model . . . 59
7.2.2 B-S model . . . 59
7.3 Partitioning strategies . . . 60
7.3.1 The reservation vectors can be different . . . 60
7.3.2 The reservation vectors are identical . . . 63

8 Algorithms 66
8.1 Double binary search algorithm . . . 66
8.2 Limited number of extreme points (No sharing) . . . 70
8.3 Partitioning and dynamic routing . . . 72
8.4 Implementation details . . . 74
8.4.1 Cost bounds . . . 74
8.4.2 Golden ratio . . . 74
8.4.3 Column and constraint generation . . . 76

9 Volume oriented strategies **77**
9.1 Volume oriented routing 77
9.1.1 Notation 78
9.1.2 Problem formulation 78
9.1.3 Computational complexity 79
9.2 Simplified volume oriented routing 81
9.2.1 Notation 81
9.2.2 Problem formulation 82
9.2.3 Computational complexity 82
9.3 General volume oriented routing 82
9.3.1 Notation 83
9.3.2 Problem formulation 83
9.3.3 Computational complexity 84

10 Numerical results **85**
10.1 Test cases 85
10.2 Partitioning strategies 86
10.3 Volume oriented strategies 91

11 Conclusions **94**

Bibliography **98**

A Résumé de thèse **106**
A.1 Première partie: Complexité 107
A.2 Deuxième partie: Modèle polyédral 111

List of Figures

3.1 Two, three, and four failure-disjoint shortest paths. 15
3.2 Simplified network with multiple-link failures. 16
3.3 Network $\mathcal{N}$ resulting from graph $\mathcal{G}$. 18
3.4 Transformation of the PD pricing problem to Shortest Path Problem with Resource Constraints. 22

4.1 Network of two nodes and N+1 links. 27
4.2 Network $\mathcal{N}$ resulting from graph $\mathcal{G}$. 30
4.3 Network exposing differences between problems. 36
4.4 Modified graph corresponding to the graph of Figure 4.3. 37
4.5 Sample instance of Infinite Capacities F-D Paths. 40
4.6 State graph. 43

7.1 Network exposing non-continuity of the cost function. 62

8.1 Network exposing non-polynomiality of the algorithm. 69
8.2 Evaluation of the lower bound of a convex function. 75

9.1 Graph proving that a decision version of VO Routing is co-$\mathcal{NP}$-complete. 80

List of Tables

4.1 Complexity overview ($\mathcal{NPH}$: $\mathcal{NP}$-hard, $\mathcal{P}$: polynomial) 26
4.2 Possible primary/backup path groups when $F_{\mathcal{G}}$ has no solution 31
4.3 Arc-disjoint primary path group pairs and their possible backup path groups 31
4.4 Overview of the complexity of approximation schemes 33

6.1 General overview of complexity 51

10.1 Test cases 87
10.2 Congestion 88
10.3 Efficiency of the partitioning algorithms—gain [%] 90
10.4 Running times of the partitioning algorithms [s] 91
10.5 Efficiency and running times of the volume oriented strategies 93

List of Symbols

a – edge (arc);

a^s, b^s – edges from Figure 3.4;

b – bottom link;

c_a – capacity of edge a;

d – demand;

$d(D, D')$ – Hausdorff distance between polytopes D and D';

e – subset of vertices;

$e(p)$ – independent set defined by the condition: vertex v_n belongs to $e(p)$ if and only if path p survives in failure state s_n;

f_a – flow on edge a;

h_d – volume of demand d;

h_{ij} – threshold for a demand ij;

i, j, v – nodes (vertices);

(i, j) – edge from i to j;

$i \rightarrow j$ – path from i to j;

ij – demand from i to j;

p, q – paths;

$p(e)$ – path corresponding to independent set e;

$r = (p, q)$ – failure-disjoint primary/backup path-pairs;

s – failure state;

s_n – failure state encompassing simultaneous failures of b_n, and of every t_m such that in graph $\mathcal{G}$ nodes v_n and v_m are adjacent;

t – traffic matrix, or top link;

t_{ij} – volume of a demand between i and j in matrix t;

t'_{ij} – volume of a demand between i and j in matrix t routed using first routing;

t''_{ij} – volume of a demand between i and j in matrix t routed using second routing;

u_d – source of demand d;

w_d – sink of demand d;

w_a – weight of edge a;

x_p – flow of demand $d \in \mathcal{D}$ allocated to path $p \in \mathcal{P}_d$;

x_{ad} – flow of demand $d \in \mathcal{D}$ allocated to edge $a \in \mathcal{A}$;

x_{dr} – flow of demand $d \in \mathcal{D}$ allocated to pair $r = (p, q) \in \mathcal{T}_d$, or flow of demand $d \in \mathcal{D}$ allocated to sequence $r = (p, q_s : s \in \bar{\mathcal{S}}_p) \in \mathcal{W}_d$;

x_p^{ij} – proportion of a traffic demand from i to j carried through a path p;

x_a^{ij} – proportion of traffic from i to j flowing through edge a;

$x_p^{ij,t}$ – proportion of a traffic demand from i to j for traffic matrix t carried through path p;

$x_a^{ij,t}$ – proportion of traffic from i to j for traffic matrix t on edge a;

y_a – total capacity of edge a;

y'_a – primary capacity of edge a;

y''_a – protection capacity of edge a;

z – congestion;

z_e – weight variable associated with independent set e;

A_i – upper bound for outgoing traffic for node i;

B_i – upper bound for incoming traffic for node i;

C – maximum size of the cover;

D – traffic demand polytope, or number of demands;

H – sum of integer numbers from $\mathcal{H}$;

$L(D, \beta)$ – left hand side polytope, i.e., $D \cap \{t : \alpha \cdot t \leqslant \beta\}$;

M – sufficiently large constant;

N – number of nodes;

$R(D, \beta)$ – right hand side polytope, i.e., $D \cap \{t : \alpha \cdot t \geqslant \beta\}$;

X – total flow realized for a demand;

$\mathcal{A}$ – set of all edges (arcs);

$\mathcal{A}_h$ – set of edges belonging to a path pair from i to j, where $h = \{i, j\}$;

$\mathcal{B}$ – list of positions of a hyperplane;

$\mathcal{D}$ – set of all demands, also set of indices;

$\mathcal{E}$ – family of all independent sets;

$\mathcal{E}^v$ – family of all independent sets containing vertex v;

$\mathcal{G}$ – graph;

$\mathcal{H}$ – sequence of positive integer numbers, or a family of subsets;

$\mathcal{I}$ – support of a feasible solution to $C_{\mathcal{G}}$;

$\mathcal{I}^v$ – support of a feasible solution to $C_{\mathcal{G}}$ containing node v;

$\mathcal{L}$ – ordered collection of reliable edges;

$\mathcal{M}$ – state understood as a sorted multiset of vertices;

$\mathcal{N}$ – network;

$\mathcal{O}$ – normal state;

$\mathcal{P}$ – set of all candidate paths;

$\mathcal{P}_d$ – set of candidate paths for demand d;

$\mathcal{P}_d^s$ – subset of paths in $\mathcal{P}_d$ that survive in state $s \in \mathcal{S}$;

$\mathcal{P}_{ad}$ – subset of paths in $\mathcal{P}_d$ that contain link $a \in \mathcal{A}$;

$\mathcal{P}(i, j)$ – set of paths from i to j;

$\mathcal{Q}_p$ – set of all candidate backup paths that can be used for protecting path p;

$\mathcal{Q}$ – support of a feasible solution to $P_{\mathcal{G}}$;

$\mathcal{Q}^s$ – support of a feasible solution to $P_{\mathcal{G}}$ surviving in failure state s;

$\mathcal{R}''_{eds}$ – set of all pairs $r = (p, q)$ such that $a \in q$ and $s \in \bar{\mathcal{S}}_p$;

$\mathcal{S}$ – family of failure scenarios;

$\mathcal{S}_a$ – set of all failure states in which edge a is available;

$\mathcal{S}_p$ – set of all failure states in which path p is available;

$\bar{\mathcal{S}}_p$ – set of all failure states where path p fails;

$\mathcal{T}$ – graph obtained by extending $\mathcal{G}$ in the proof of Lemma 4.2.5;

$\mathcal{T}_d$ – set of all candidate failure-disjoint primary/backup path-pairs for demand d;

$\mathcal{T}'_{ad}$ – set of all pairs $r = (p, q) \in \mathcal{T}_d$ such that $a \in p$;

$\mathcal{U}$ – universe;

$\mathcal{V}$ – set of all nodes (vertices);

$\mathcal{V}^{act}$ – set of active nodes;

$\mathcal{W}$ – set of vertices for Fractional Coloring;

$\mathcal{W}'_{ad}$ – set of all sequences $r = (p, q_s : s \in \bar{\mathcal{S}}_p) \in \mathcal{T}_d$ such that $a \in p$;

$\mathcal{W}''_{ads}$ – set of sequences $r = (p, q_s : s \in \bar{\mathcal{S}}_p) \in \mathcal{T}_d$ such that $a \in q_s$ for $s \in \bar{\mathcal{S}}_p$;

$\mathbb{P}$ – problem FI-nSR;

α – direction of a hyperplane;

β – position of a hyperplane;

γ – precision;

γ_a – component of w_a defines as: $\gamma_a = \lambda_d^{\{a\}^\star}$ if $\{a\} \in \mathcal{S}$, and $\gamma_a = 0$ otherwise;

$\delta^+(v)$ – set of all edges leaving node v;

$\delta^-(v)$ – set of all edges entering node v;

λ_d^s – dual variables for (3.2b), or length of the shortest path with respect to π_a^s;

ξ_a – unit cost of edge a;

$\xi_a(f_a)$ – cost of edge a carrying flow f_a;

ξ_p – unit cost of path p;

π_a^s – dual variables for (4.1d), (4.3c), (5.1d), or for (5.4c);

π_v – dual variable for (3.5b);

$\chi_f(\mathcal{G})$ – fractional chromatic number of graph $\mathcal{G}$;

$2D_{\mathcal{G}}$ – instance of 2Div-Path for $\mathcal{G}$;

$FC_{\mathcal{G}}$ – instance of Fractional Coloring for $\mathcal{G}$;

$FD_{\mathcal{G}}$ – instance of FD-nSR, or FDM, corresponding to $FC_{\mathcal{G}}$;

$FI_{\mathcal{G}}$ – instance of FI-nSR corresponding to $2D_{\mathcal{G}}$;

$FI_{\mathcal{H}}$ – instance of FI-nSR corresponding to $PA_{\mathcal{H}}$;

$IC_{\mathcal{U}}$ – instance of Infinite Capacities F-D Paths corresponding to $SC_{\mathcal{U}}$;

$PA_{\mathcal{H}}$ – instance of Partition for $\mathcal{H}$;

$PD_{\mathcal{G}}$ – instance of PD corresponding to $VC_{\mathcal{G}}$;

$PDD_{\mathcal{G}}$ – instance of PDD corresponding to $FC_{\mathcal{G}}$;

$SC_{\mathcal{U}}$ – instance of Set Cover for $\mathcal{U}$;

$SS_{\mathcal{H}}$ – instance of Subset Sum for $\mathcal{H}$;

$VC_{\mathcal{G}}$ – instance of the Vertex Cover for $\mathcal{G}$;

$VO_{\mathcal{H}}$ – instance of VO Routing corresponding to $SS_{\mathcal{H}}$;

List of Abbreviations

B-S – Bertsimas-Sim model;

DR – Dynamic Routing;

DR/RR – Different Reservation for Robust Routing;

DWDM – Dense Wavelength Division Multiplexing;

FD – Failure-Dependent restoration;

FD-nSR – Failure-Dependent restoration without Stub-Release;

FD-SR – Failure-Dependent restoration with Stub-Release;

FGC – Fractional Graph Coloring;

FI – Failure-Independent restoration;

FI-nSR – Failure-Independent restoration without Stub-Release;

FI-SR – Failure-Independent restoration with Stub-Release;

GVO – General Volume Oriented routing;

HS – Hot-Standby;

IP – Internet Protocol, or Integer Programming;

IR/NS – Identical Reservation for No Sharing;

IR/RR – Identical Reservation for Robust Routing;

LHSP – Left Hand Side Polytope;

LP – Linear Programming;

MIP – Mixed-Integer Programming;

MPLS – Multi-Protocol Label Switching;

$\mathcal{NP}$ – Non-deterministic Polynomial;

$\mathcal{NPH}$ – $\mathcal{NP}$-hard;

NS – No Sharing;

nSR – without Stub-Release;

$\mathcal{P}$ – Polynomial;

PD – Path Diversity;

PDD – special case of PD;

PP – Path Protection;

PP/PD – Pricing Problem for Path Diversity;

PR – Path Restoration;

RHSP – Right Hand Side Polytope;

RR – Robust Routing;

SNDlib – Survivable fixed telecommunication Network Design library;

SPPRC – Shortest Path Problem with Resource Constraints;

SR – with Stub-Release;

SVO – Simplified Volume Oriented routing;

VO – Volume Oriented routing;

Chapter 1

Introduction

In core wide-area transport networks already single network elements (such as transmission facilities or switching nodes) carry large amounts of traffic. This makes such networks vulnerable to failures, in particular to cable cuts, so that appropriate recovery mechanisms have to be used to avoid major traffic losses. For example, in DWDM optical transport networks, a transmission facility has capacity up to the order of TB/s and, if unprotected, its failure can cause enormous losses of traffic before restoration actions are taken. This was the reason to devote the first part of the thesis to the complexity of resilient network optimization problems.

In fact, transport networks are best protected through the restoration mechanisms of the transport layer so that the transport operators (carriers) can fully control the resiliency of their networks and use the most efficient means in terms of restoration time and consumption of extra protection capacity of transmission facilities. Moreover, the carriers do know what failures can be expected (as for example failures of certain conduits) and can effectively cope with them, also at the network planning stage. Another factor here is that having reliable IP links (protected in the transport layer), IP network operators can concentrate on providing resilience to the failures occurring in the IP layer, e.g., node breakdowns, and do not bother about the failures in the transport layer. Note that failures in the transport layer, if not protected, are often seen at the IP layer as complicated multiple IP link failures on which the operators have no direct control.

One of the most common mechanisms for securing transport communication networks against failures is path protection (PP). In optical networks the mechanisms of this kind protect end-to-end lambda paths. An example is the 1+1 hot-standby protection mechanism in which a primary (working) path is assigned a dedicated disjoint protection (backup) path which carries the same signal and therefore assures the end-to-end transmission whenever the primary path fails. The 1+1 mechanism is efficient in terms of restoration time, but costly in terms of extra (protection) capacity. The protection capacity is not shared between different primary paths in different failure situations and therefore must be at least equal

to the working capacity. Some of its variants (as $n : m$ protection and path diversity) can alleviate the issue of high protection capacity cost, but only to a limited extent.

Because of that, active mechanisms called path restoration (PR) can be considered instead. Although PR can be complex in terms of implementation and can have longer restoration times than PP, they are among the most efficient mechanisms as far as the extra link capacity required for path flow protection is concerned. The idea of a PR mechanism is as follows. The traffic demand (a set of lambda paths in the case of an optical transport network) between a pair of nodes is realized using several different network paths to route the connections (i.e., path flows, either non-bifurcated or bifurcated) in the normal (failure-free) state. When a failure occurs, the affected primary connections are restored using backup connections (again, by means of either non-bifurcated or bifurcated path flows) along the surviving routes. In any failure state the total capacity of the temporarily established backup flows and the primary flows that survive is equal to the requested volume of the traffic demand.

The backup flows can be established using the separate protection (backup) capacity of the links so that the protection link capacity is separated from the working link capacity used by primary flows that traverse a given link. In effect, the working and protection link capacity form two separate pools of resources, so the backup flows do not use the working capacity released on links as a by-product of a failure. It is said that backup flows do not exploit the phenomenon of *stub-release*. Another possibility is when backup flows are allowed to use also the working capacity that is released by the failed primary connections on surviving links, so that the backup flows can exploit stub-release. This in fact leads to the usage of links' capacity as a common pool of resources. In both cases it is important that backup capacity *is shared* between restoration flows established in different failure states for different traffic demands, so the path restoration *is not* the dedicated hot-standby protection.

The restoration of primary flows can be either failure-independent or failure-dependent. In the first case the path flows have to be restored in exactly the same way whatever is the reason of their failure. In the second case, the backup flows used to restore a given primary flow can be different in different failure situations.

In general, the complexity of optimization problems corresponding to a restoration mechanism can depend on the assumed failure scenario, in particular whether the failure scenario contains only a set of single element (link) failures or a set of failures consisting of multiple elements' failures; multiple failures are often referred to as *shared risk resource (link) groups*. In the first part the following thesis is verified.

Most of resilient multi-commodity flow network optimization problems are $\mathcal{NP}$-hard, even when flows are allowed to be bifurcated.

To this aim, complexity of various variants of the multi-commodity flow network optimization problems related to the above described PP and PR mechanisms is studied. All the considered problems proved to be either $\mathcal{NP}$-hard or polynomial, and all of them are $\mathcal{NP}$-hard for multiple failures. This fact is not commonly recognized, especially when bifurcated (fractional) flows are admitted. Depending on the problem either $\mathcal{NP}$-hardness or polynomiality proof is given. Some of the results discussed in this thesis can be found in the literature, while others are results of the present thesis.

The complexity issues of resilient network design have been considered by several authors, see [20,34,47,51,55,59,65]. In fact, the previous results do not directly discuss the complexity of the basic optimization problems but are rather devoted to the complexity of the so called pricing problem required for column generation (see [1]) in the related non-compact linear formulations of the basic problems (see [51]). These results are important (they will be used while discussing efficient ways of dealing with the considered problems), still they do not decide on the complexity of the basic problems which is the main subject of the first part of the thesis.

Other authors were studying similar problems with only slightly different settings, like [4, 5], where the authors were considering one single failure only, or they were trying to provide good quality lower bounds for the problems considered in the thesis, like in [10].

It is worth noticing that the complexity results for multi-commodity flow optimization problems related to design of resilient networks are important as they help to develop proper algorithms used in network design tools based on contemporary linear/integer programming solvers such as CPLEX or XPRESS.

The second part of the thesis is devoted to the polyhedral traffic demand matrix uncertainty model. The reason was that modern telecommunication networks carry traffic generated by a variety of different applications, and provide services to a large number of users. This makes prediction of traffic patterns difficult. Moreover, the customers' mobility makes a traffic demand matrix change in short periods of time. All those factors do not allow for considering only one traffic demand matrix. Instead they encourage us to work on traffic demand matrix uncertainty models.

Some such models were proposed in the past. The first approach consists in building a traffic matrix based on the worst case for each traffic component. Routing is then computed based on this matrix. While this approach is simple, it can provide expensive solutions, because in fact it does not allow non-coincident traffic components to share resources.

The second approach is based on probabilistic modeling of traffic variations. After specifying such a model, one may look for routing that optimizes a probabilistic criterion: throughput expectation, average delays, blocking probability, etc. The solution obtained in this way is good on average, but can be very bad in some cases. Moreover, it requires knowledge of probability models, and they are usually difficult to obtain. This kind of approach is generally called stochastic programming, and was used for example in [44, 57], where a finite number of traffic scenarios with a known probability are considered.

A different general approach called robust optimization (see [42]) takes into account a finite number of possible situations. In this case a solution that supports all situations is of interest. In the networking context, a routing scheme is to be determined such that each traffic matrix belonging to a given finite set of traffic matrices can be carried in the network.

Another robust model was proposed in [27] (and independently in [23]). It assumes that the outgoing traffic of each node is bounded (limitations on the incoming traffic can also be considered). Then the traffic matrix can be any matrix satisfying this kind of constraints. This uncertainty model is called the hose model. Several network design problems based on this model have received a considerable attention in the literature (see [18] and references therein).

A different model is considered in [13] where the authors assume that not all nodes can generate the maximum amount of traffic simultaneously. In fact this model, and the hose model presented above, are special cases of a more general polyhedral model considered in [6–9]. The model assumes that each possible traffic matrix belongs to a polytope D. In [9] a polynomial time algorithm is proposed to compute a routing scheme that solves this problem. The routing is robust and stable (robust because it is compatible with all matrices, and stable because it does not change when the matrix changes — it is matrix-independent). Although the model was defined for traffic matrices belonging to a polytope, it can also successfully deal with problems assuming that traffic matrices belong to a union of polytopes.

A different approach to deal with uncertainty consists in allowing a network to change routing in a dynamic way, when there is a considerable change in terms of a traffic matrix. This problem was intensively studied in the context of circuit-switched networks. Different rules can be applied to connect calls depending on the current situation, e.g., alternate dynamic routing, sequential routing, trunk reservation (see [3] and references therein). While this kind of routing has many benefits, it is generally difficult to implement, and it is also not optimal because the rules that are used are fixed in advance. However, it can be used as a reference point (lower bound) to evaluate approaches presented earlier. Lately it has been proven that the optimal cost of a network based on robust stable routing (fractional or integral) may be a factor of $\Omega(\log n)$ larger than the cost required when using dynamic routing

[30]. For a more detailed discussion of different approaches to the problem of uncertainty proposed in the past, readers can consult a recent survey that can be found in [18]. What is more, other works related to traffic uncertainty can be found in [2,17,25,26,45,49,53,54,63].

In [9] the following question was raised: given an uncertainty traffic demand polytope D, is it theoretically easy to compute a fully dynamic routing (routing scheme that depends on the current traffic matrix)? Recently, it was proved in [19] that this problem is in general difficult: it is co-$\mathcal{NP}$-hard to decide whether a given network with known capacities is capable of carrying each traffic matrix $t \in D$, when routing is dynamic. Moreover, it is clearly difficult to implement this kind of routing for technical reasons.

Therefore, it is worth considering something between robust stable routing and fully dynamic routing. Instead of computing a robust solution, it is possible to partition the traffic demand polytope into some subsets, and compute a robust routing for each of them. This class of problems was considered in [6]. In the second part of the thesis, algorithms solving problems introduced in [6] are presented.

In the sequel of this thesis it is assumed that a direction of a hyperplane dividing the traffic demand polytope is known, but its position is subject to optimization. For example an amount of traffic generated between a pair of nodes (like in the experiments presented in Chapter 10), or an amount of traffic generated by a specific node can be considered as a direction of the hyperplane. In such a situation a position of a threshold that triggers changes in routing is optimized. In other words, two routings are specified, the first is used when the amount of traffic, which was set as a direction of the hyperplane, is smaller than the threshold, otherwise the second routing is used. Another possibility is to consider time as a direction of the hyperplane. In this case time is represented by an additional dimension of the uncertainty set, and the moment when routing should be changed is optimized.

The major problem with this kind of approaches is that a routing can significantly differ between subsets, and implementing those changes can result in serious traffic losses (due to instability). So it is also worth considering something that will make the routing changes less abrupt. Therefore, in the second part the following thesis is verified.

There exist routing mechanisms that are decentralized and simple in implementation, and are capable of dealing with traffic demands described using a polyhedral approach.

To this aim, a novel routing mechanism, called volume oriented routing (and its polynomially solvable modifications: simplified volume oriented routing and general volume oriented routing), is presented in the thesis. It takes advantage of the simplicity of robust stable routing and the efficiency of dynamic routing. Moreover, it does not involve abrupt changes in network flows.

The whole thesis is divided into two parts. The first part deals with the complexity of resilient network optimization problems, while the second part is devoted to the polyhedral traffic demand matrix uncertainty model.

The first part of the thesis is organized as follows. In Chapter 2 notation is introduced. After that, in Chapter 3 optimization models related to path protection (hot-standby, HS, and path diversity, PD) are discussed. With HS, i.e., 1+1 protection, each demand is realized by means of a single working path which is protected by a dedicated single path. PD can be considered as a generalization of HS. It protects traffic against failing network components by over-provisioning, i.e., by routing more traffic than specified by the demand value in the failure-less state, and ensuring that at least the requested volume survives in each failure state in the considered failure scenario without rerouting any flow.

In Chapters 4 and 5, analogous models related to various variants of path restoration mechanisms are considered. Path restoration (PR) are active mechanisms in the sense that they restore the working flows affected by a failure. Certainly, for achieving that, extra protection capacity is required as compared to capacity supporting normal state only. Still, the amount of extra capacity is typically much smaller for PR than for PP because, as a rule, with PR protection capacity is shared by different demands and different failure situations. The computational complexity of all the introduced models is discussed by summarizing known facts and presenting novel results. The first part of the thesis ends with concluding remarks in Chapter 6.

The second part of the thesis is organized as follows. Chapter 7 begins with Section 7.1, where a notation and formulations of three basic models is presented, i.e., robust routing, no-sharing routing, and dynamic routing. Robust routing assumes that traffic matrices are always routed in the same way regardless of their locations in the traffic demand polytope. No-sharing routing is a special case of robust routing which assumes that capacities cannot be shared between different demands. The last model, namely dynamic routing, let each traffic matrix from the traffic demand polytope be routed in a different way. It is shown which of these problems are polynomial, and how they can be successfully approached. In Section 7.2, models that were used to create test cases are presented, while in Section 7.3 the basic models are enhanced by introducing two versions of partitioning of the traffic demand polytope: capacities have to be reserved once for the whole traffic demand polytope, or capacities can be rereserved each time the routing changes. In the thesis special features of these partitioning problems and algorithm that solve them are presented.

In Chapter 8, different algorithms solving the presented problems are discussed. In Section 8.1 the first partitioning problem is discussed. Its complexity is considered, and some of its special features are outlined. Moreover, an algorithm that can solve it is presented. Another

(faster) algorithm is presented in Section 8.2. However, it can be applied only to a subset of basic problems presented in a special way. The whole Section 8.3 is devoted to dynamic routing. It is shown when partitioning is useful for this problem, and how it can be solved. The major interest is given to a special polynomial case of the problem, i.e., the traffic demand polytope is defined as a convex hull of a given set of traffic matrices. Finally, Section 8.4 consists of implementation details that accelerate the presented algorithms. It deals with all the problems presented before, and contains enhancements that do not change the theoretical complexity of the presented algorithms but can accelerate actual implementations.

In Chapter 9, volume oriented strategies are presented. First, in Section 9.1, unrestricted volume oriented routing is discussed. This novel routing mechanism involves a division of a traffic demand polytope. However, in this case, only a part of flow that exceeds a threshold is routed using a different set of paths. The complexity of this approach is $\mathcal{NP}$-hard in general. However, it encompasses some special cases that are polynomial even when more that one hyperplane is taken into account. Those special cases, namely simplified volume oriented routing and general volume oriented routing, are presented in Sections 9.2 and 9.3, respectively.

Numerical results showing applicability of the algorithms are presented in Chapter 10. Congestions obtained for the considered basic models are compared first. Secondly, results showing possible gains while applying different partitioning strategies are provided. Finally, efficiency of volume oriented strategies is discussed. Numerical results are followed by conclusions and suggestions of further research in Chapter 11.

Some of the results discussed in the first part of the thesis have been presented at the 5th Polish-German Teletraffic Symposium (PGTS), Berlin, Germany, in September 2008 [72], and subsequently in European Transactions on Telecommunications in 2009 [73]. While other results were presented in Networks: an International Journal in 2010 [64]. Finally, some results were presented at International Symposium on Combinatorial Optimization (ISCO), Hammamet, Tunisia in March 2010 [70], and published in IEEE Communications Letters also in 2010 [71].

The results concerning the partitioning strategies discussed in the thesis were first presented by the author at the 16th Polish Teletraffic Symposium, Łódź, Poland, in September 2009 [67], and at GLOBECOM, Honolulu, USA, in December 2009 [68]. They are also covered in an article, written by the author and Walid Ben-Ameur, published in International Transactions in Operational Research [11]. The results concerning the volume oriented strategies were presented at the 14th International Telecommunications Network Strategy and Planning Symposium, Warsaw, Poland, in September 2010 [69].

Part I

Complexity Issues

Chapter 2

Notation

The considered network is modeled as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ composed of set $\mathcal{V}$ of nodes and set $\mathcal{A}$ of links (directed arcs). Set $\mathcal{D}$ represents directed end-to-end (traffic) demands. The number of all demands, $|\mathcal{D}|$, will be denoted by D. The source and target nodes of demand $d \in \mathcal{D}$ are denoted by u_d and w_d, respectively. The volume of demand $d \in \mathcal{D}$ is given by h_d. The demand volumes are realized by means of path flows assigned to (directed) paths form u to w. The cost of realizing one unit of demand on link $a \in \mathcal{A}$ is denoted by ξ_a. Each link $a \in \mathcal{A}$ has its *working* (*primary, basic*) *capacity*, denoted by y'_a, used for realizing flows in the normal operating state (i.e., *working* or *primary flows*), and the *protection* or *backup capacity* y''_a used for restoration of failed primary flows by means of *backup* or *restoration flows*. Both y'_a and y''_a, $a \in \mathcal{A}$ will be variables subject to optimization. In the cases when it is not necessary to distinguish between the working and protection capacity, variables y_a will be used to denote the total capacity of link $a \in \mathcal{A}$.

Notice that bounds on the capacity variables are not considered. Adding those bounds cannot simplify the considered problems, thus $\mathcal{NP}$-hard problems will remain $\mathcal{NP}$-hard after the modification. However, it is possible that some cases that were in $\mathcal{P}$ can become $\mathcal{NP}$-hard when the bound are added.

The family of all failure states (called a *failure scenario*) considered in a particular design problem is denoted by $\mathcal{S}$ where each *failure state* $s \in \mathcal{S}$ (also called a failure situation or simply a failure) is identified by the set of failing links, so $s \subset \mathcal{A}$. Family $\mathcal{S}$ is assumed to include the failure-less state $\mathcal{O}$ (formally equal to the empty subset of the set $\mathcal{A}$) in which all links are operational (state $\mathcal{O}$ is sometimes called the *normal state*). It is assumed that links fail totally. The set $\mathcal{S}_a = \{s \in \mathcal{S} \ : \ a \notin s\}$ will denote the set of all states $s \in \mathcal{S}$ in which link $a \in \mathcal{A}$ is available. A failure scenario is called a *single-link failure scenario* if it admits only states s consisting of singletons (not necessarily all) plus the normal state $\mathcal{O}$. Otherwise, a scenario is called *multiple-link failure scenario*.

Node failures are not explicitly considered in this thesis. In fact, they do not add difficulty to the considered problems as they can be modeled as single-link failures through a suitable

transformation of the network graph (see Section 4.6.1 in [55] and Section 9.3 in [51]).

The set of all candidate paths (in general, these are not all possible paths) that can be used for carrying flows is denoted by $\mathcal{P} = \bigcup_{d\in\mathcal{D}} \mathcal{P}_d$, where $\mathcal{P}_d$ is the set of candidate paths for demand $d \in \mathcal{D}$. It is assumed that the paths are elementary (do not contain loops) so, for each demand $d \in \mathcal{D}$, set $\mathcal{P}_d$ is a subset of the set of all elementary paths from u_d to w_d, and each path $p \in \mathcal{P}$ can be identified with the set of the links it traverses, hence $p \subseteq \mathcal{A}$. Further, $\mathcal{S}_p = \{s \in \mathcal{S} \ : \ p \cap s = \emptyset\}$, $(\mathcal{S}_p \subseteq \mathcal{S})$ denotes the set of all states $s \in \mathcal{S}$ in which path $p \in \mathcal{P}$ is available, and $\bar{\mathcal{S}}_p = \mathcal{S} \setminus \mathcal{S}_p$ is the set of all states $s \in \mathcal{S}$ in which path $p \in \mathcal{P}$ fails. The subset of paths in $\mathcal{P}_d$ that survive in state $s \in \mathcal{S}$ is denoted by $\mathcal{P}_d^s$, and the subset of paths in $\mathcal{P}_d$ that contain link $a \in \mathcal{A}$ by $\mathcal{P}_{ad}$.

For a given candidate path $p \in \mathcal{P}_d$ assigned to demand $d \in \mathcal{D}$, the set of all candidate backup paths that can be used for protecting path p is denoted by $\mathcal{Q}_p$. Certainly, $\mathcal{Q}_p \subseteq \mathcal{P}_d$ and $s \in \mathcal{S}_q$ for all $s \in \bar{\mathcal{S}}_p$, i.e., paths p and q never fail simultaneously (and therefore are called *failure-disjoint*). In this context, path $p \in \mathcal{P}$ is called the *primary path* (or working path) and all paths from $\mathcal{Q}_p$ are called its *backup paths* (or protection or restoration paths). The set of all candidate failure-disjoint primary/backup path-pairs $r = (p, q)$ for demand $d \in \mathcal{D}$ will be denoted by $\mathcal{T}_d$, $\mathcal{T}_d = \{r = (p, q) \ : \ p \in \mathcal{P}_d, q \in \mathcal{Q}_p\}$. For each link $a \in \mathcal{A}$, and demand $d \in \mathcal{D}$, the set of all pairs $r = (p, q) \in \mathcal{T}_d$ such that $a \in p$ will be denoted by $\mathcal{T}'_{ad}$, and the set of all pairs $r = (p, q) \in \mathcal{T}_d$ such that $a \in q$ – by $\mathcal{T}''_{ad}$.

For the failure-dependent restoration still another path structures will be needed. The set $\mathcal{W}_d$, defined for each $d \in \mathcal{D}$, contains predefined candidate sequences $r = (p, q_s : s \in \bar{\mathcal{S}}_p)$ of paths from u_d to w_d with the property that path q_s works in state $s \in \bar{\mathcal{S}}_p$ (i.e., when the primary path p fails). Then, $\mathcal{W}'_{ad}$ is the set of all sequences $r = (p, q_s : s \in \bar{\mathcal{S}}_p) \in \mathcal{T}_d$ such that $a \in p$, and $\mathcal{W}''_{ads}$ is the set of sequences $r = (p, q_s : s \in \bar{\mathcal{S}}_p) \in \mathcal{T}_d$ such that $a \in q_s$ for $s \in \bar{\mathcal{S}}_p$.

Depending on a particular optimization problem, the following flow variables will be used: x_p – flow of demand $d \in \mathcal{D}$ allocated to path $p \in \mathcal{P}_d$, x_{ad} – flow of demand $d \in \mathcal{D}$ allocated to link $a \in \mathcal{A}$, x_{dr} – flow of demand $d \in \mathcal{D}$ allocated to pair $r = (p, q) \in \mathcal{T}_d$ or flow of demand $d \in \mathcal{D}$ allocated to sequence $r = (p, q_s : s \in \bar{\mathcal{S}}_p) \in \mathcal{W}_d$.

Notice that in the thesis both links and demand are directed. Still, the majority of proofs to be presented can be easily transformed to cover undirected cases. However, there exist cases when the transformation is impossible to achieve, e.g., in a proof of Section 4.1.3 2Div-Path problem is used, which is $\mathcal{NP}$-hard for directed cases but remains in $\mathcal{P}$ for undirected cases.

Chapter 3

Path Protection Problems

Path protection (PP) mechanisms are passive in the sense that no flows are restored, and hence the flows surviving in any of the considered failure states must be sufficient to realize the demands. In this chapter the basic hot-standby PP mechanism (HS), and its extension called path diversity (PD) are presented. In both cases we do not have to consider protection capacity on the links but rather talk about redundant working capacity.

3.1 HS: Hot-Standby

With HS, i.e., 1+1 protection, each demand $d \in \mathcal{D}$ is realized by means of a single working path p which is protected by a dedicated single path q, $r = (p, q) \in \mathcal{T}_d$. The two paths are failure-disjoint, i.e., they never fail together (of course for the assumed failure scenario $\mathcal{S}$). Both paths are assigned flow h_d which consumes h_d units of capacity on each link along path p, and also h_d units along path q, so if there is a link $a \in \mathcal{A}$ belonging to both paths then the amount of capacity consumed on this link by the considered demand is equal to $2h_d$. The design problem corresponding to HS (referred to as problem HS) can be formulated as the following MIP (mixed-integer programming) problem.

Problem HS

$$\text{minimize } F(y) = \sum_{a \in \mathcal{A}} \xi_a y_a, \tag{3.1a}$$

$$\sum_{r \in \mathcal{T}_d} x_{dr} = 1, \qquad d \in \mathcal{D}, \tag{3.1b}$$

$$\sum_{d \in \mathcal{D}} h_d \Big(\sum_{r \in \mathcal{T}'_{ad}} x_{dr} + \sum_{r \in \mathcal{T}''_{ad}} x_{dr} \Big) \leqslant y_a, \qquad a \in \mathcal{A}, \tag{3.1c}$$

$$x_{dr} \in \{0, 1\}, \qquad d \in \mathcal{D},\ r \in \mathcal{T}_d. \tag{3.1d}$$

Observe that the use of binary flow variables x_{dr} forces any feasible solution of problem (3.1) to be non-bifurcated (unsplittable) because for each demand $d \in \mathcal{D}$ it assigns the

whole demand volume h_d to the pair $r = (p, q) \in \mathcal{T}_d$ for which x_{dr} is equal to 1 (by (3.1b) there is only one such pair). Moreover, due to (3.1c), $y_a \geqslant 0$ for all $a \in \mathcal{A}$. In fact, the above formulation yields a binary (i.e., non-bifurcated) optimal vertex solution even if the integrality condition in (3.1d) is relaxed to $0 \leqslant x_{dr} \leqslant 1$, i.e., when the linear relaxation is considered instead of the MIP formulation (3.1). In other words, in any optimal vertex solution $(x^\star, y^\star)$ of the linear relaxation, for each demand $d \in \mathcal{D}$ there is exactly one path-pair $r \in \mathcal{T}_d$ with $x^\star_{dr} = 1$, and for the rest of such pairs $x^\star_{dr} = 0$.

Note that the linear relaxation of (3.1) is not compact because the number of variables x_{dr} is in general exponential (as the number of path-pairs $r = (p, q)$ is exponential). Hence, to take into account all possible variables we have to use the column generation (path-pair generation in this case) method of LP (see [43, 48]), based on solving the so called *pricing problem*. For the considered problem the pricing problem consists in finding, separately for each demand $d \in \mathcal{D}$, a shortest pair $r = (p, q)$ of failure-disjoint paths, i.e., a pair minimizing its primal cost $\sum_{a \in p} \xi_a + \sum_{a \in q} \xi_a$. Observe that in this particular case, finding such shortest path-pairs directly resolves the problem (3.1) (see [51]).

When dealing with a single-link failure scenario, finding a shortest pair of failure-disjoint paths for a given demand $d \in \mathcal{D}$ (and hence solving problem HS (3.1)) can be done in polynomial time using the Suurballe algorithm [60] or its modification given in [14]. In fact, this problem is equivalent to the single-commodity min-cost-flow problem with flow value 2 and link capacities 1, see [1].

For the general case of multiple-link failures, however, the problem (3.1) is $\mathcal{NP}$-hard since, as shown in [34], even a simpler problem of finding just any pair of failure-disjoint paths can be reduced from the proven $\mathcal{NP}$-complete problem Set Splitting (see p. 221 in [29]).

3.2 PD: Path Diversity

The main disadvantage of HS is that it requires at least twice the capacity of an unprotected network. Therefore, better PP mechanisms in terms of capacity efficiency are of interest. One such mechanism, probably most economical among PP mechanisms in terms of capacity, is called path diversity (PD). PD protects traffic against failing network components by over-provisioning, i.e., by routing more traffic than specified by the demand value in the failure-less state $\mathcal{O}$, and ensuring that at least the volume h_d survives in each failure state in the considered failure scenario without rerouting any flow. The concept of PD has been studied in the literature under different names, for example *diversification* [21] and *demand-wise shared protection* [40, 41, 66].

Each path $p \in \mathcal{P}_d$ is assigned a flow variable denoted by x_p that expresses a part of the

required demand volume h_d to be carried on path p. Thus, the quantity $\sum_{p\in\mathcal{P}_a} x_p$ is the load of link a ($a \in \mathcal{A}$) expressed in the flow units. For a given flow vector $x = (x_p : \ p \in \mathcal{P})$ and with the non-negative cost of carrying one flow unit on link $a \in \mathcal{A}$ denoted by ξ_a ($\xi_a \geqslant 0$), we can express the cost of using link a as $\xi_a(\sum_{p\in\mathcal{P}_a} x_p)$, and the overall cost of the network by $F(x) = \sum_a \xi_a(\sum_{p\in\mathcal{P}_a} x_p)$. The problem studied in this section, Path Diversity Design (abbreviated by PD), is as follows.

Problem PD

$$\text{minimize } F(x) = \sum_{a\in\mathcal{A}} \xi_a\Big(\sum_{p\in\mathcal{P}_a} x_p\Big), \tag{3.2a}$$

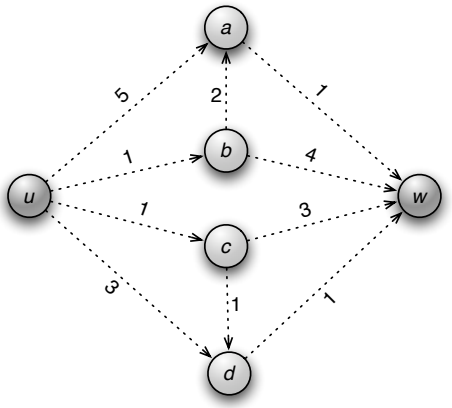
$$\sum_{p\in\mathcal{P}_d^s} x_p \geqslant h_d, \qquad d \in \mathcal{D},\ s \in \mathcal{S}, \tag{3.2b}$$

$$x \geqslant 0. \tag{3.2c}$$

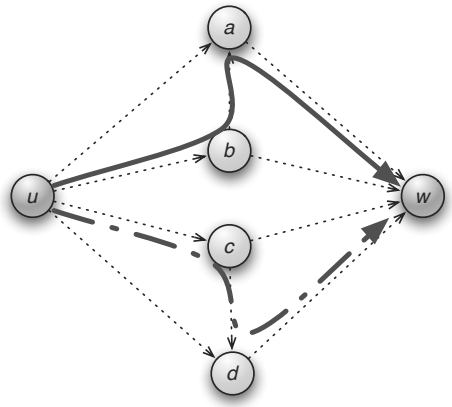
Problem PD was studied by Koster et al. [41], Wessäly et al. [66], Orlowski and Pióro [51], and in a slightly different setting by Dahl and Stoer [21]. Observe that formulation (3.2) specifies a linear programming problem, and as such can be effectively treated by LP solvers but only when the predefined sets of candidate paths $\mathcal{P}_d$, $d \in \mathcal{D}$, are of reasonable size. Since in general the proper sets of candidate paths are not known in advance, formally all possible (elementary) paths in the network have to be considered. Consequently, formulation (3.2) is not compact since in general the number of all elementary paths in a graph increase exponentially with the number of nodes. Thus, to solve problem (3.2), column generation has to be used (see for example [1]) to achieve the proper sets of candidate paths that contain all paths required in an optimal solution.

Path generation issues for problem (3.2) are discussed in [51]. Path generation is known to be polynomial for the case when the failure scenario $\mathcal{S}$ contains only single-link failures, i.e., when $\mathcal{S}$ is composed of singletons: $\mathcal{S} \subseteq \{\{a\} : a \in \mathcal{A}\}$. In fact, path generation for problem (3.2) remains polynomial even if single-node failures on top of single-link failures are also considered. However, in the case of a general multiple-link failure scenario, path generation for PD becomes $\mathcal{NP}$-hard [50].

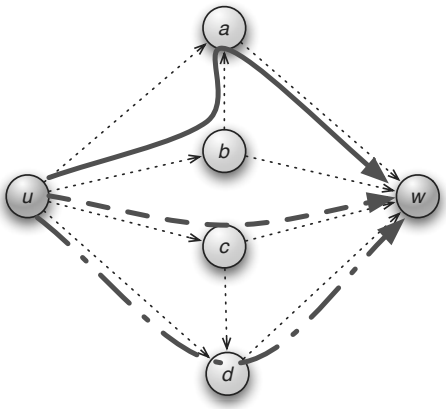
Due to the linearity of the objective function, problem (3.2) decomposes into $|\mathcal{D}|$ independent sub-problems, one for each demand $d \in \mathcal{D}$. Without loss of generality in each such sub-problem $h_d = h$ can be assumed. Then, using notations $\mathcal{P} = \mathcal{P}_d$, $\mathcal{P}^s = \mathcal{P}_d^s$ and $\xi_p = \sum_{a\in p} \xi_a$, the decomposed problem for each particular demand $d \in \mathcal{D}$ is of the following form.



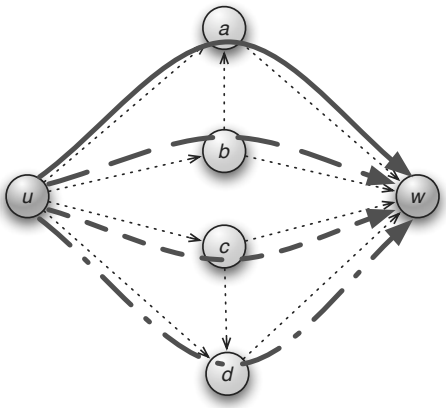
(a) Network illustrating problem PD.



(b) Two failure-disjoint shortest paths.



(c) Three failure-disjoint shortest paths.



(d) Four failure-disjoint shortest paths.

Figure 3.1: Two, three, and four failure-disjoint shortest paths.

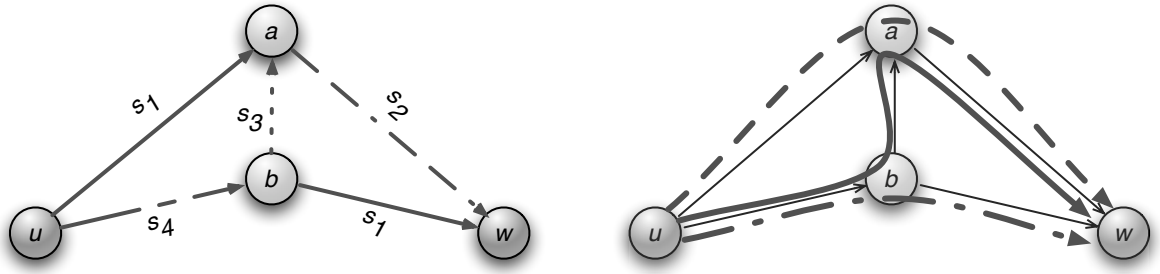
Problem PDD (special case of PD)

$$\text{minimize } F(x) = \sum_{p \in \mathcal{P}} \xi_p x_p, \tag{3.3a}$$

$$\sum_{p \in \mathcal{P}^s} x_p \geqslant h, \qquad s \in \mathcal{S}, \tag{3.3b}$$

$$x \geqslant 0. \tag{3.3c}$$

Example To illustrate problem PD consider the network depicted in Figure 3.1a with a single demand of volume h from node u to node w. For the failure scenario consisting of single failures of all links the optimal solution consists of three flows equal to $x_p = \frac{h}{2}$ assigned to each of the three link-disjoint shortest paths depicted in Figure 3.1c. The cost $F(x)$ of this solution is $6h$. Note that the feasible solution that assigns flow $x_p = h$ to each of the two disjoint shortest paths of Figure 3.1b has the cost $F(x) = 7h$, and the feasible solution that assigns flow $x_p = \frac{h}{3}$ to each of the four disjoint paths of Figure 3.1d has the cost $F(x) = \frac{19}{3}h$,



(a) Multiple-link failure scenarios.

(b) Non-disjoint paths solution.

Figure 3.2: Simplified network with multiple-link failures.

and hence both of them are not optimal.

The character of the optimal solution changes when multiple-link failures are admitted. Consider the four node network (being a subnetwork of the network from Figure 3.1a) with failure scenario $\mathcal{S} = \{s_1, s_2, s_3, s_4\}$ as shown in Figure 3.2a. The (unique) optimal solution uses non-disjoint paths, as illustrated in Figure 3.2b. All the three path flows are still equal to each other (they are equal to h). Clearly, the cost of this solution is $15h$. □

In fact, for any scenario $\mathcal{S}$ consisting of single-link failures, $\mathcal{S} \subseteq \{\{a\} : a \in \mathcal{A}\}$ (in $\mathcal{S}$ not necessarily all links are subject to failure), problem PDD can be expressed in the compact node-link LP formulation, and hence solved in polynomial time taking (indirectly) into account all possible candidate paths (see [51]).

Problem PDD (compact formulation)

$$\text{minimize } F(x) = \sum_{a \in \mathcal{A}} \xi_a x_a, \tag{3.4a}$$

$$\sum_{a \in \delta^+(v)} x_a - \sum_{a \in \delta^-(v)} x_a = 0, \qquad v \in \mathcal{V} \setminus \{u, w\}, \tag{3.4b}$$

$$\sum_{a \in \delta^+(u)} x_a - \sum_{a \in \delta^-(u)} x_a = X, \tag{3.4c}$$

$$X - x_a \geqslant h, \qquad \{a\} \in \mathcal{S}, \tag{3.4d}$$

$$x \geqslant 0. \tag{3.4e}$$

Above, each variable x_a denotes the flow realizing the considered demand (between nodes $u, w \in \mathcal{V}$) on link $a \in \mathcal{A}$, X is the total flow realized for the demand, and $\delta^+(v)$ and $\delta^-(v)$ are the sets of all links, respectively, outgoing from and incoming to node $v \in \mathcal{V}$.

Moreover, for the particular failure scenario consisting of single failures of all links ($\mathcal{S} = \{\{a\} : a \in \mathcal{A}\}$) there always exists an optimal solution of PDD that uses only a set of k link-disjoint paths, so that the character of the solution exhibited in the first part of

Example 3.2 is general (see [15]). The optimal solution assigns equal flows $x_p = \frac{h}{k-1}$ to each of these paths (note that $x_p = \frac{X}{k}$). Knowing this, it is possible to use the following iterative algorithm to resolve PDD for the scenario that assumes single failures of every link:

Step 0: Set $k = 1$ and $C(k) = +\infty$.

Step 1: Find a minimum-cost set $P(k+1)$ of $k+1$ link-disjoint paths. If the set does not exist go to Step 3, otherwise denote the total cost of the paths by $C(k+1) = \sum_{p \in P(k+1)} \xi_p$.

Step 2: If $C(k+1) \cdot (k-1) < C(k) \cdot k$ set $k = k+1$ and go to Step 1.

Step 3: Assign flow equal to $\frac{h}{k-1}$ to each path from $P(k)$.

The algorithm is effective. Its core (Step 1) consists of finding a minimum-cost set $P(k+1)$ of $k+1$ link-disjoint paths for the consecutive values of k so essentially it is an application of the Suurballe approach [60] (see its modification in [14]) which finds a shortest set of $(k+1)$ disjoint paths from the previously found shortest set of k disjoint paths. Consequently, set $P(k+1)$ is determined by finding a shortest "interlacing" path for set $P(k)$. Such a path can be computed using the original Suurballe algorithm or by any shortest path algorithm admitting negative link weights in a graph with no negative cycles. However, a more efficient way is to use the transformation of [24] (before finding a consecutive interlacing path) to assure nonnegative link weights and then apply the classical Dijkstra algorithm to find the interlacing path. Using the transformation, and the fact that there cannot be more than $|\mathcal{E}|$ link-disjoint paths, the overall complexity of the algorithm becomes $O(|\mathcal{A}|^2 \log_{(2+|\mathcal{A}|/|\mathcal{V}|)} |\mathcal{V}|)$ [62, Chapter 8].

The above result on using disjoint paths (and the resulting algorithm) can be extended to the case when not all links are subject to (single) failures, i.e., when $\mathcal{S} \subsetneq \{\{a\} : a \in \mathcal{A}\}$. Then there always exists an optimal solution with flows equal to $\frac{h}{k-1}$ assigned to a set of k *failure-disjoint* paths, i.e., to a set of paths $\{p_1, p_2, \ldots, p_k\}$ from u to w such that each pair (p_i, p_j) $(1 \leqslant i < j \leqslant k)$ has no common failing link (i.e., $a \in p_i \cap p_j \Rightarrow a \in \mathcal{A} \setminus \mathcal{S}$). Such a solution can be found by the above algorithm applied to the network modified for each k by replacing each non-failing link a ($a \in \mathcal{A} \setminus \mathcal{S}$) by a set of k parallel links (with the same unit cost equal to ξ_a) that are subject to failure. Certainly, the parallel links are shrunk to the original link upon completion of the algorithm. Note that this solution has to be compared with the best possible solution that uses only one non-failing path (provided such a path exists). Still, the version of Path Diversity Design admitting multiple-link failures and using all possible elementary paths is $\mathcal{NP}$-hard.

Proposition 3.2.1. Path Diversity Design *is $\mathcal{NP}$-hard when multiple failures are admitted.*

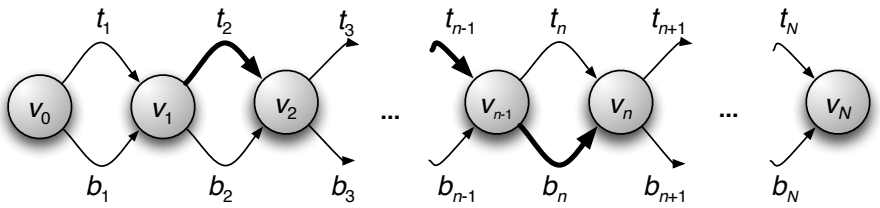


Figure 3.3: Network $\mathcal{N}$ resulting from graph $\mathcal{G}$.

The proposition can be proved using a reduction of the fractional graph coloring problem (referred to as Fractional Coloring) to Path Diversity Design. The reduction and the fractional graph coloring problem itself are presented below.

3.2.1 $\mathcal{NP}$-hardness proof

Fractional graph coloring problem

Consider a graph $\mathcal{G}$ with the set of vertices $\mathcal{W}$. Any non-empty subset of vertices $e \subseteq \mathcal{W}$ is called an *independent set* of graph $\mathcal{G}$ if no two vertices in e are connected by an edge. Let $\mathcal{E}$ be the family of all independent sets of graph $\mathcal{G}$ and let $z = (z_e \geqslant 0 : e \in \mathcal{E})$ be a vector of (weight) variables associated with those independent sets. Finally, let $\mathcal{E}^v = \{e \in \mathcal{E} : v \in e\}$ be the family of all independent sets containing a given vertex $v \in \mathcal{W}$. A (non-compact) LP formulation of Fractional Coloring is as follows.

Problem FGC (fractional graph coloring)

$$\text{minimize } F(z) = \sum_{e \in \mathcal{E}} z_e, \tag{3.5a}$$

$$\sum_{e \in \mathcal{E}^v} z_e \geqslant 1, \qquad v \in \mathcal{W}, \tag{3.5b}$$

$$z \geqslant 0. \tag{3.5c}$$

Any optimal solution $z^\star$ of problem FGC determines the *fractional chromatic number* $\chi_f(\mathcal{G})$ which is equal to $f(z^\star)$. Fractional Coloring is known to be $\mathcal{NP}$-hard (its decision version is $\mathcal{NP}$-complete); see [31, 46].

$\mathcal{NP}$-hardness proof

Consider an undirected graph $\mathcal{G}$ with the set $\mathcal{W} = \{v_1, v_2, \ldots, v_N\}$ of N vertices. Let $\mathcal{E}$ be the family of all independent sets of $\mathcal{G}$. The resulting instance of Fractional Coloring (i.e., of problem FGC (3.5)) will be denoted by $FC_{\mathcal{G}}$.

The instance $PDD_{\mathcal{G}}$ of Path Diversity Design (i.e., of problem PDD (3.3)) corresponding to $FC_{\mathcal{G}}$ is modeled by means of a directed network $\mathcal{N} = (\mathcal{V}, \mathcal{A})$ depicted in Figure 3.3. The set of nodes $\mathcal{V} = \{v_0, v_1, \ldots, v_N\}$ consists of $N+1$ elements: all vertices from graph $\mathcal{G}$ plus the additional node v_0. The set of links $\mathcal{A} = \{t_1, t_2, \ldots, t_N, b_1, b_2, \ldots, b_N\}$ is composed of $2 \cdot N$ elements. For each $n = 1, 2, \ldots, N$, links t_n (*top link* number n) and b_n (*bottom link* number n) connect nodes v_{n-1} and v_n. The unit cost ξ_a of each link is equal to 1. The (single) demand is defined from node v_0 to node v_N, with the requested demand volume equal to 1. The set $\mathcal{P}$ of paths consists of all elementary paths from node v_0 to node v_N in network $\mathcal{N}$. The set $\mathcal{S}$ of failure states consists of N states, $\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$. Each state $s_n \in \mathcal{S}$ encompasses simultaneous failures of the bottom link b_n, and of every top link t_m such that in graph $\mathcal{G}$ vertices v_n and v_m are linked by an edge: $s_n = \{b_n\} \cup \{t_m : \ v_m$ and v_n are adjacent in $\mathcal{G}\}$. For example, thick links in Figure 3.3 represent the links that fail in state s_n under assumption that vertex v_n in graph $\mathcal{G}$ is adjacent to vertex v_2 and to vertex v_{n-1}, and to no other vertices.

Lemma 3.2.2. *The optimal objective* (3.5a) *of* $FC_{\mathcal{G}}$ *is equal to* K *(i.e.,* $\chi_f(\mathcal{G}) = K$*) if and only if the optimal objective* (3.3a) *of* $PDD_{\mathcal{G}}$ *is equal to* $N \cdot K$*.*

Proof. Each feasible solution $z = (z_e : e \in \mathcal{E})$ of $FC_{\mathcal{G}}$ can be identified with its support $\mathcal{I} = \{e \in \mathcal{E} : z_e > 0\}$ and the vector $z = (z_e : e \in \mathcal{I})$ of positive weights assigned to the independent sets from the support. Analogously, each feasible solution $x = (x_p : p \in \mathcal{P})$ of $PDD_{\mathcal{G}}$ can be identified with its support $\mathcal{Q} = \{p \in \mathcal{P} : x_p > 0\}$ and the vector $x = (x_p : p \in \mathcal{Q})$ of positive flows assigned to the paths from the support. The subfamily of all independent sets from $\mathcal{I}$ containing a given vertex $v \in \mathcal{W}$ will be denoted by $\mathcal{I}^v$, and the set of all paths from $\mathcal{Q}$ surviving in a given state $s \in \mathcal{S}$ by $\mathcal{Q}^s$. It will be demonstrated that each feasible solution $\mathcal{I}, z = (z_e : e \in \mathcal{I})$ of $FC_{\mathcal{G}}$ corresponds to a feasible solution $\mathcal{Q}, x = (x_p : p \in \mathcal{Q})$ of $PDD_{\mathcal{G}}$ such that $N \cdot f(z) = F(x)$, and vice versa.

Let $\mathcal{I}$, $z = (z_e : e \in \mathcal{I})$ be a given feasible solution of $FC_{\mathcal{G}}$. The corresponding feasible solution of $PDD_{\mathcal{G}}$ consists of a set $\mathcal{Q} = \{p(e) : e \in \mathcal{I}\} \subseteq \mathcal{P}$ of paths with the flow assignment defined as $x_{p(e)} = z_e, e \in \mathcal{I}$. For each n ($1 \leqslant n \leqslant N$), path $p(e)$ uses the top link t_n, if $v_n \in e$; otherwise, it uses the bottom link b_n. In effect, since $s_n = \{b_n\} \cup \{t_m : \ v_m$ and v_n are adjacent in $\mathcal{G}\}$ and $p(e) = \{t_n : v_n \in e\} \cup \{b_m : v_m \notin e\}$, it can be observed that $\mathcal{Q}^{s_n} = \{p(e) : e \in \mathcal{I}^{v_n}\}$ and therefore for each $s_n \in \mathcal{S}$ we have $\sum_{p \in \mathcal{Q}^{s_n}} x_p = \sum_{e \in \mathcal{I}^{v_n}} x_{p(e)} = \sum_{e \in \mathcal{I}^{v_n}} z_e \geqslant 1$, so the inequalities (3.3b) hold. As every path p in $\mathcal{P}$ has the unit cost $\xi_p = N$, the total cost $F(x)$ given by (3.3a) is equal to $N \cdot f(z)$.

Conversely, each feasible solution $\mathcal{Q}, x = (x_p : p \in \mathcal{Q})$ of $\mathrm{P}_{\mathcal{G}}$ defines a feasible solution $\mathcal{I} = \{e(p) : p \in \mathcal{Q}\} \subseteq \mathcal{E}, z = (z_{e(p)} : p \in \mathcal{Q})$ of $\mathrm{C}_{\mathcal{G}}$, where $z_{e(p)} = x_p, p \in \mathcal{Q}$. Each set $e(p) \in \mathcal{I}$ is defined by the condition: vertex v_n belongs to $e(p)$ if and only if path p survives in failure state s_n. Each $e(p)$ is an independent set since if $v_n, v_m \in e(p)$ and v_n and v_m are adjacent in $\mathcal{G}$ then path p cannot survive in both states s_n and s_m because link t_n fails in state s_m and link b_n fails in state s_n. Moreover, $\mathcal{I}^{v_n} = \{e(p) : p \in \mathcal{Q}^{s_n}\}$ and hence $\sum_{e \in \mathcal{I}^{v_n}} z_e = \sum_{e(p) \in \mathcal{Q}^{s_n}} z_{e(p)} = \sum_{p \in \mathcal{Q}^{s_n}} x_p \geqslant 1$, so the inequalities (3.5b) are fulfilled. Certainly, $N \cdot f(z) = F(x)$.

The above construction shows that instances $FC_{\mathcal{G}}$ and $PDD_{\mathcal{G}}$ represent essentially the same optimization problem. Thus, $K = \chi_f(\mathcal{G})$ if and only if $N \cdot K$ is the optimal objective function value of $PDD_{\mathcal{G}}$. □

3.2.2 Non-bifurcated PD

Certainly, LP formulation (3.2) of problem PD is bifurcated as it does not pose any restrictions on assigning path flows x_p except (3.2b). A non-bifurcated version of PD would require that each path in $p \in \mathcal{P}_d$ can carry either flow equal to 0 or to h_d. This variant of PD can be easily formulated as a MIP using binary flow variables.

The non-bifurcated version of PD is polynomial in the case of single failures because its optimal solution is equivalent to a corresponding solution of HS, i.e., it consists of a pair of failure disjoint paths. To see this assume (without loss of generality) that there is only one demand in a network, and replace all non-failing links with a pair of failing, parallel ones. In this way an equivalent problem in which all links can fail is obtained. Secondly, the capacities allocated to links resulting from a solution to PD (optimal or not) are reduced, so that they do not exceed h_d on any link. Now it is possible to observe that the capacity of the minimal cut in the resulting network is at least equal to $2h_d$ (otherwise, all paths of the PD solution would traverse the same failing link). Then we can exclude from the network, one by one, links that do not belong to any of the $2h_d$-capacity cuts. The resulting network consists only of links that belong to $2h_d$-capacity cuts and are of capacity h_d. In such a network the only way to route $2h_d$ of flow is to allocate it to two arc-disjoint paths. Notice that according to the max-flow min-cut theorem (see [1]) such paths can always be found in the resulting network.

For multiple failures the non-bifurcated version of PD is $\mathcal{NP}$-hard by the following reduction from the $\mathcal{NP}$-complete problem Vertex Cover (see [37]). Consider problem $VC_{\mathcal{G}}$ of finding the minimal vertex cover in graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ that consists of $|\mathcal{V}|$ vertices and $|\mathcal{A}|$ edges. The corresponding instance PD, denoted by $PD_{\mathcal{G}}$, is specified by a network composed of two nodes u and w connected by $|\mathcal{V}|$ parallel links. Each link in $PD_{\mathcal{G}}$ corresponds to one node in $VC_{\mathcal{G}}$. There is only one demand (between nodes u and w), and the failure scenario consists of $|\mathcal{A}|$ failure states. Each of the states corresponds to a link (denoted by a) in $VC_{\mathcal{G}}$ and it affects all links in $PD_{\mathcal{G}}$ that correspond to nodes which are not adjacent to the considered link a in $VC_{\mathcal{G}}$. Problems $PD_{\mathcal{G}}$ and $VC_{\mathcal{G}}$ are equivalent, as each feasible solution to $PD_{\mathcal{G}}$ can be transformed to a feasible solution $VC_{\mathcal{G}}$ with the same cost, and vice versa. The transformation applies to all feasible solutions to both problems.

3.2.3 Path generation

Recall that the path generation method (called column generation in the general LP context) is a way to generate paths outside the current set of candidate paths $\mathcal{P}$ that can possibly improve the solution obtained for the given $\mathcal{P}$, or to find out that such paths do not exist.

By the general *separation theorem* (see for example [32]), $\mathcal{NP}$-hardness of Path Diversity Design implies difficulty (non-polynomiality) of path generation for the non-compact formulation PDD (3.3) (and hence for the more general formulations PD (3.2)). In the case of PDD, however, it is easy to show that path generation is in fact $\mathcal{NP}$-hard itself. $\mathcal{NP}$-hardness of path generation for PDD (3.3) follows directly from the hardness of column generation for problem FGC (3.5). The problem dual to FGC is as follows.

Dual to problem FGC

$$\text{maximize } W(\pi) = \sum_{v \in \mathcal{W}} \pi_v, \tag{3.6a}$$

$$\sum_{v \in e} \pi_v \leqslant 1, \qquad e \in \mathcal{E}, \tag{3.6b}$$

$$\pi \geqslant 0. \tag{3.6c}$$

Note that variable π_v corresponds to (3.5b). Hence, to generate a new column for the current optimal dual solution $\pi^\star = (\pi_v^\star : v \in \mathcal{W})$, i.e., a new independent set e to be added to the current set $\mathcal{E}$, we have to solve the problem of finding an independent set in graph $\mathcal{G}$ maximizing the sum $\sum_{v \in e} \pi_v^\star$ (this problem is $\mathcal{NP}$-hard as it contains the problem of finding a maximal independent set in a graph, see [29]). If the resulting maximum for solution e is greater than 1 then the new independent set e is added to $\mathcal{E}$. Clearly, generating new paths for problem PDD (3.3) is equivalent to generating new columns for FGC (3.5), and hence the former problem must be $\mathcal{NP}$-hard as well.

As explained in [51], path generation for problem PD (3.2) is based on the following *pricing problem* PP/PD, solved separately for each demand $d \in \mathcal{D}$. PP/PD consists of finding a path p between the end nodes u_d and w_d of demand $d \in \mathcal{D}$ minimizing the generalized path length.

$$\langle p \rangle = \sum_{a \in p} \xi_a + \sum_{s \in \bar{\mathcal{S}}_p} \lambda_d^{s\star}. \tag{3.7a}$$

In the above expression, $\lambda_d^{s\star}$ denotes the optimal value of the dual variable corresponding to inequality (3.2b) while $\bar{\mathcal{S}}_p$ stands for the set of all failure states $s \in \mathcal{S}$ that affect path p (p is assumed to be elementary and is identified with the set of links it traverses). Note that $\lambda_d^{s\star} \geqslant 0$ so it is clear that $\langle p \rangle \geqslant 0$. If the resulting $\langle p \rangle$ is strictly less than the generalized lengths of all paths in the current candidate set $\mathcal{P}$ then p is added to $\mathcal{P}$. If for at least one demand $d \in \mathcal{D}$ such a path has been added, the resulting (extended by one or more paths) problem PD is resolved, and the pricing problems are reentered.

PP/PD is polynomial for single-link failure scenarios (when links are subject to failures, but only one link can fail at a time), and $\mathcal{NP}$-hard for scenarios admitting states with

simultaneous failures of several links (see [51, Section 3]). In the single-link failure case, the pricing problem can be easily solved by means of any classical shortest path algorithm (as for example the Dijkstra algorithm [22]) for link weights given by $w_a = \xi_a + \gamma_a$, where $\gamma_a = \lambda_d^{\{a\}\star}$ if $\{a\} \in \mathcal{S}$, and $\gamma_a = 0$, otherwise. As shown before PP/PD is $\mathcal{NP}$-hard for multiple-link failures. In fact, it is $\mathcal{NP}$-hard already for double-link failures, $\mathcal{S} = \{\{a, a'\} \in \mathcal{A} \times \mathcal{A} : \ a \neq a'\}$, as demonstrated in [20] for a simpler problem called the minimum-color shortest path problem (assuming $\xi_a = 0, a \in \mathcal{A}$ in (3.7)).

For a multiple-link failure scenario the pricing problem can be formulated as a MIP (mixed-integer program), see problem (14) in [51], and resolved as such by a MIP solver. Still, according to computational experience described in [56,59], such a direct MIP approach is not efficient because the MIP model of the pricing problem does not solve well and fails to deliver solutions in a reasonable time already for networks of very moderate size (10 nodes, say).

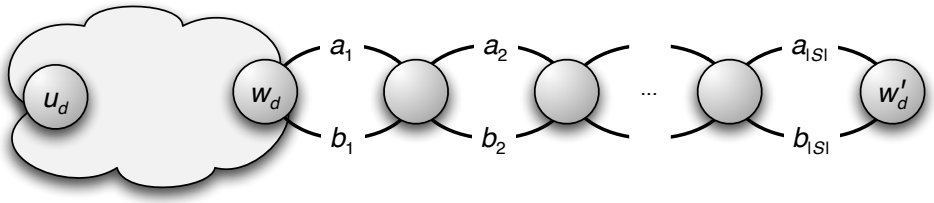


Figure 3.4: Transformation of the PD pricing problem to Shortest Path Problem with Resource Constraints.

Alternatively, the pricing problem can be converted into an instance of the Shortest Path Problem with Resource Constraints (SPPRC) [36]. For a given instance of the pricing problem, the corresponding instance of SPPRC is constructed as in Figure 3.4. The network of the original problem is represented as a cloud in the figure. Each link in this graph has the primal cost equal to ξ_a, as in the pricing problem. For the SPPRC problem instance we introduce $|\mathcal{S}|$ resources (corresponding to the failure states), and assume that link $a \in \mathcal{A}$ consumes one unit of resource $s \in \mathcal{S}$ if a fails in s, and nothing otherwise. The original network is extended by $|\mathcal{S}|$ additional nodes and $2 \cdot |\mathcal{S}|$ links denoted by a^s and b^s, $s \in \mathcal{S}$, as shown in Figure 3.4.

Links a^s have the primal cost $\xi_{a^s} = 0$; these links consume a large amount $M \geqslant |\mathcal{A}|$ of resource s and none of all other resources. Links b^s have the primal cost $\xi_{b^s} = \lambda_d^{s\star}$ and do not consume any resources. Let w'_d denote the rightmost node as depicted in Figure 3.4. The objective of the resulting instance of SPPRC is to find a shortest path (with respect to the primal link costs) from u_d to w'_d in the transformed network, satisfying the constraint

forbidding the path to consume more than M units of any resource on any link. This leads to a routing path with the cost as defined by (3.7): the contribution of the path to the first sum comes from the cloud, and the resource constraints of links a^s make sure that if a path uses state s somewhere in the cloud, link b^s must be used, which adds the value of $\lambda_d^{s\star}$ to the path length. Using this transformation, it is possible to solve the pricing problem using algorithms developed for SPPRC, for example based on dynamic programming (see [36]).

A general label-setting SPPRC algorithm was used for the pricing problem related to another resilient design problem (state-independent restoration without stub-release) in [59]. However, as discussed in [56], it turns out that using general label-setting SPPRC algorithms for the pricing problems of PD considered in this thesis is not effective either (as the MIP approach). Fortunately, such label-setting algorithms can be improved and made work efficiently for practical communication networks instances. The way how to do it for PD is described in [56], where also numerical examples supporting the above observations are discussed.

Chapter 4

Path Restoration Problems with Failure-Independent Restoration

Path restoration (PR) are active mechanisms in the sense that they restore the working flows affected by a failure. Certainly, for achieving that, extra protection capacity is required as compared to capacity supporting normal state only. Still, the amount of extra capacity is typically much smaller for PR than for PP because, as a rule, with PR protection capacity is shared by different demands and different failure situations.

In this chapter it is assumed that failed flows are restored in a failure-independent fashion, i.e., the restoration flow pattern is always the same, and does not depend on the particular failure state that affects a given primary flow. The failure-dependent case is discussed in Chapter 5.

PR mechanisms are considered assuming two cases of capacity use, i.e., with and without stub-release. In the former case, the capacity on surviving parts (stubs) of a failing path can be reused for backup flows. In the latter case, it is reserved exclusively for the primary flows, and cannot be reused in case of any failure.

4.1 FI-nSR: no Stub-Release

First the case without stub-release (nSR) will be considered, i.e., when surviving but unused working capacity cannot be reused for backup flows in failure situations. Notice that nSR is typical for transport layers, e.g., for optical or SONET networks. The generic form of the core optimization problem considered in this section (FI-nSR) is denoted by $\mathbb{P}$ and is as follows.

Problem FI-nSR

$$\text{minimize } F(y) = \sum_{a\in\mathcal{A}} \xi_a(y'_a + y''_a), \tag{4.1a}$$

$$\sum_{r\in\mathcal{T}_d} x_{dr} = 1, \qquad d \in \mathcal{D}, \tag{4.1b}$$

$$\sum_{d\in\mathcal{D}} \sum_{r\in\mathcal{T}'_{ad}} h_d x_{dr} \leqslant y'_a, \qquad a \in \mathcal{A}, \tag{4.1c}$$

$$\sum_{d\in\mathcal{D}} \sum_{r\in\mathcal{R}''_{eds}} h_d x_{dr} \leqslant y''_a, \qquad a \in \mathcal{A},\ s \in \mathcal{S}_a \setminus \{\mathcal{O}\}, \tag{4.1d}$$

$$y''_a \geqslant 0, \qquad a \in \mathcal{A}, \tag{4.1e}$$

$$x_{dr} \geqslant 0, \qquad d \in \mathcal{D},\ r \in \mathcal{T}_d. \tag{4.1f}$$

Problem $\mathbb{P}$ consists in minimizing the cost of primary and protection capacities installed on the links subject to a number of constraints. Constraint (4.1b) assures that all demand volumes are realized (x_{dr} is a fraction of h_d allocated to pair $r = (p, q) \in \mathcal{T}_d$) while (4.1c) states that in the normal state the load of link $a \in \mathcal{A}$ cannot exceed its primary capacity. The flow summation on the left-hand side of (4.1d) is taken over all path-pairs whose backup path contains the considered link a and whose primary path fails in the considered failure state s. Clearly, problem (4.1) assumes that the pool of protection capacity y''_a, $a \in \mathcal{A}$, is shared by the demands in different situations. Note that, $y' \geqslant 0$ for all $a \in \mathcal{A}$ dur to (4.1c).

In the balance of this section $\mathcal{NP}$-hardness of different variants of problem (4.1) will be examined. The variants are distinguished according to the following four criteria:

- **Criterion 1.** (a) Non-bifurcated flow (IP): $x_{dr} \in \{0, 1\}, d \in \mathcal{D}, r \in \mathcal{T}_d$. (b) Bifurcated flow (LP): $x_{dr} \in \mathbb{R}^+, d \in \mathcal{D}, r \in \mathcal{T}_d$.
- **Criterion 2.** (a) Predefined lists of path-pairs: $\mathcal{T}_d$ are given in advance. (b) All possible elementary path-pairs in $\mathcal{T}_d$.
- **Criterion 3.** (a) Single link failure scenario: $|s| = 1, s \in \mathcal{S}\setminus\mathcal{O}$. (b) Arbitrary link failure scenario (failures of more than one link at a time are admitted): $|s| \geqslant 1, s \in \mathcal{S} \setminus \mathcal{O}$.
- **Criterion 4.** (a) Single demand: $D = 1$. (b) Multiple demands: $D \geqslant 1$.

4.1.1 Complexity overview

As indicated in Criterion 1, by IP the version of problem $\mathbb{P}$ that is characterized by binary flow variables (non-bifurcated flows) is denoted, and by LP—its bifurcated counterpart (linear relaxation of IP) with continuous flows.

First observe that the variant specified by Criterion 1b (LP) and Criterion 2a (predefined path-pairs lists) is always polynomial, no matter what Criteria 3 and 4 are. This is because

Table 4.1: Complexity overview ($\mathcal{NPH}$: $\mathcal{NP}$-hard, $\mathcal{P}$: polynomial)

		single failures	multiple failures
$D = 1$	LP, all paths	$\mathcal{NPH}$[1]	$\mathcal{NPH}$[2]
	IP, predefined paths	$\mathcal{P}$[3]	$\mathcal{NPH}$[4]
$D > 1$	LP, all paths	$\mathcal{NPH}$[5]	$\mathcal{NPH}$[5]
	IP, predefined paths	$\mathcal{NPH}$[6]	$\mathcal{NPH}$[5]

in this case formulation (4.1) becomes a compact linear programming problem, and as such can be solved in time which grows polynomially with the size of the problem (see [39]). On the other hand, when LP with all possible (elementary) path-pairs is considered (variant 1b, 2b) then formulation (4.1) becomes non-compact as the number of path-pairs grows exponentially with the size of the graph. As demonstrated later in this section, this variant is $\mathcal{NP}$-hard, no matter what Criteria 3 and 4 are.

The second observation concerns variants assuming Criterion 1a (IP). For the IP case the complexity is already determined by 2a, no matter what variants of Criteria 3 and 4 are considered. When variant 2a is $\mathcal{NP}$-hard, variant 2b is also (automatically) $\mathcal{NP}$-hard. The case when variant 1a can be solved in polynomial time is a bit more tricky. This is in fact variant 1a, 3a, 4a, namely: $|s| = 1$, for all $s \in \mathcal{S} \setminus \mathcal{O}$, $\mathcal{D} = \{d\}$ and $x_{dr} \in \{0, 1\}, r \in \mathcal{T}_d$. Consider Criterion 2a, i.e., that $\mathcal{T}_d$ are given in advance. Then the optimal solution can be found in polynomial time by examining all pairs in $\mathcal{T}_d$, and selecting the shortest one. Now consider the second variant (2b). In this case the problem is just to find a shortest pair of disjoint paths in the network, and allocate the whole demand volume h_d to this pair. This can be efficiently solved using Suurballe algorithm [60]. Hence, both variants 2a and 2b can be solved in polynomial time.

Thus, although according to Criteria 1-4 there are 16 variants of problem (4.1), because of the above remarks, this number can be reduced to 8. These 8 variants are summarized in Table 4.1, where Criterion 2b is assumed for LP, and Criterion 2a is assumed for IP.

When Criterion 1b is assumed instead of 1a so the flow bifurcation is admitted, i.e., vector $x = (x_{dr} \ : \ d \in \mathcal{D}, r = (p, q) \in \mathcal{T}_d)$ is continuous, then a demand can be routed on many different primary/backup path-pairs simultaneously. How different the IP and the LP versions of the problem are can be seen in Figure 4.1 depicting the network consisting of

[1] proved in this section
[2] derived from the proof presented in this section
[3] proved by Suurballe, see [60]
[4] proved by Hu, see [34]
[5] because $D = 1$ is $\mathcal{NP}$-hard
[6] modification of the main proof presented in this section

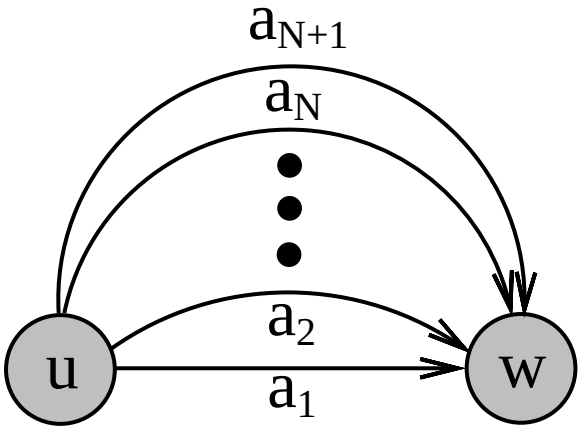


Figure 4.1: Network of two nodes and N+1 links.

two nodes u and w, $N+1$ links $a_1, a_2, \ldots, a_{N+1}$, and one demand d between u and w with $h_d = 1$. Suppose that $\xi_{a_n} = 1$, for $n = 1, 2, \ldots, N+1$. Clearly, any optimal solution of the IP version consists (by definition) of just one primary/backup path-pair, and must cost 2 units. On the contrary, an optimal solution of the LP version uses N different primary paths protected by one common backup path. Such a solution is for example obtained by assigning flow $x_{dr} = \frac{1}{N}$ to each of the N path-pairs (p_n, q), for $n = 1, 2, \ldots, N$, where $p_n = \{a_n\}$ and $q = \{a_{N+1}\}$. Hence, $y'_{a_n} = \frac{1}{N}, y''_{a_n} = 0$, for $n = 1, 2, \ldots, N$ and $y''_{a_{N+1}} = \frac{1}{N}, y'_{a_{N+1}} = 0$, and the backup capacity of link a_{N+1} is shared among the different backup paths in different failure states (each of these failure states affects exactly one link). The cost of this solution is $1 + \frac{1}{N}$, i.e., almost 50% cheaper than the non-bifurcated solution.

Later in this section it is proved that the 1a, 3a (IP, single failures) variant of problem (4.1) is $\mathcal{NP}$-hard. Moreover, the fact that any approximation of the problem better than a $\frac{4}{3}$-approximation is $\mathcal{NP}$-hard is also proved.

Proceed now to the case with multiple demands and with the single failure scenario (3a, 4b). The $\mathcal{NP}$-hardness of the LP case (variant 1b, 3a, 4b) is directly implied by the $\mathcal{NP}$-hardness of the case with only one demand (Criteria 1b, 3a, 4a), since when something is difficult for the single demand case it has to be difficult also for multiple demands. As far as the IP version is concerned (1a, 3a, 4b), it is impossible to follow the same way of reasoning, because the single demand case can be solved in polynomial time. In fact, the version admitting multiple demands is $\mathcal{NP}$-hard. In order to prove this it is possible to use a simple construction presented in this section, and reduce the PARTITION problem to the considered variant of problem $\mathbb{P}$. However, since this construction gives weak results concerning the complexity of the problem's approximations, the main proof of this section

will be extended to cover the IP variant of the problem admitting multiple demands (variant 1a, 3a, 4b).

As far as the multiple failure variant 1b, 3b, 4a is concerned, its complexity can be deduced from [34] where it is proved that finding any pair of failure-disjoint paths when multiple failures are admitted is $\mathcal{NP}$-hard. As each feasible solution to the problem consists of assigning the demand volume to one or more of such pairs, it is clear that finding any of them cannot be completed in polynomial time.

4.1.2 $\mathcal{NP}$-hardness proof (many demands, IP)

Consider the variant of problem $\mathbb{P}$ assuming Criteria 1a, 2a, 3a, and 4b, i.e., IP, limited lists of path-pairs, single failures, and many demands. The $\mathcal{NP}$-hardness of this case can be demonstrated using a reduction of the PARTITION problem to the considered problem. PARTITION was proved to be $\mathcal{NP}$-hard in [29].

Consider a given sequence $\mathcal{H} = (h_1, h_2, \ldots, h_D)$ of positive integer numbers. Denote by $\sum \hat{\mathcal{D}}$ the sum $\sum_{d\in\hat{\mathcal{D}}} h_d$ defined for any subset $\hat{\mathcal{D}}$ of the set of indices $\mathcal{D} = \{1, 2, \ldots, D\}$. Problem PARTITION consists in answering the question whether there is a partition (split) of the set $\mathcal{D}$ into two subsets $\mathcal{D}'$ and $\mathcal{D}''$ ($\mathcal{D}' \cup \mathcal{D}'' = \mathcal{D}, \mathcal{D}' \cap \mathcal{D}'' = \emptyset$), such that $\sum \mathcal{D}'$ is equal to $\sum \mathcal{D}''$. Partition $(\mathcal{D}', \mathcal{D}'')$ will be called a *valid partition*.

Consider an instance $PA_{\mathcal{H}}$ of PARTITION for a given sequence $\mathcal{H}$. The corresponding instance $FI_{\mathcal{H}}$ of problem (4.1) is specified by means of a network that consists of only two nodes, u and w, connected by three parallel links $\mathcal{A} = \{a_1, a_2, a_3\}$ (see Figure 4.1 with $N = 3$). The unit capacity costs are all equal to 1, $\xi_a = 1$ for $a \in \mathcal{A}$. The set of demands is defined as $\mathcal{D} = \{1, 2, \ldots, D\}$, and each demand $d \in \mathcal{D}$ has its source in node u and its sink in node w, its volume is given by h_d. Also, the (single) failure of each of the three links is included into the considered failure scenario, so the scenario contains the normal state $\mathcal{O}$ and the three failure states corresponding to the three links. Let $H = \sum \mathcal{D}$.

Proposition 4.1.1. *The optimal objective of $FI_{\mathcal{H}}$ is $\frac{3}{2}H$ if and only if $PA_{\mathcal{H}}$ forms a valid partition.*

Proof. It will be first proved that the optimal objective $F^\star$ of $FI_{\mathcal{H}}$ is not less than $\frac{3}{2}H$. Indeed, it must be that $y'_{a_1} + y'_{a_2} + y'_{a_3} \geqslant H$ (because the total demand volume to be realized from u to w is equal to H), and that $y''_{a_2} + y''_{a_3} \geqslant y'_{a_1}$, $y''_{a_1} + y''_{a_3} \geqslant y'_{a_2}$ and $y''_{a_1} + y''_{a_2} \geqslant y'_{a_3}$ (because when link a fails then its primary capacity y'_a must be restored on the protection capacity of the two remaining links). Summing up the last three inequalities and using the first inequality we get $2 \cdot (y''_{a_1} + y''_{a_2} + y''_{a_3}) \geqslant y'_{a_1} + y'_{a_2} + y'_{a_3} \geqslant H$, and hence $y''_{a_1} + y''_{a_2} + y''_{a_3} \geqslant \frac{1}{2}H$. Thus, $F^\star \geqslant \frac{3}{2}H$.

Next, it will be shown that a valid partition $(\mathcal{D}', \mathcal{D}'')$ yields an optimal solution of $FI_{\mathcal{H}}$. Such a solution is obtained by assigning the path-pair $r' = (p', q)$ to all demands from $\mathcal{D}'$, and the path-pair $r'' = (p'', q)$ to all demands from $\mathcal{D}''$, where the primary path p' is composed of link a_1, the primary path p'' is composed of link a_2, and the common backup path q is

composed of link a_3. Then $y'_{a_1} = y'_{a_2} = y''_{a_3} = \frac{1}{2}, y'_{a_3} = y''_{a_1} = y''_{a_2} = 0$, and the resulting objective $F(y)$ is $\frac{3}{2}H$, thus optimal.

Finally, it will be proved that any optimal solution of $FI_{\mathcal{H}}$ defines a valid partition of $\mathcal{D}$. Let $y'_a, y''_a, a \in \mathcal{A}$ be such an optimal solution. Introducing the notation $y_a = y'_a + y''_a, a \in \mathcal{A}$ it is possible to write that $y_{a_1} + y_{a_2} + y_{a_3} = \frac{3}{2}H$ (because the solution is optimal and hence $F(y) = \frac{3}{2}H$) and $y_{a_2} + y_{a_3} \geqslant H$, $y_{a_1} + y_{a_3} \geqslant H$ and $y_{a_1} + y_{a_2} \geqslant H$ (because when link a fails then the total demand volume must be carried on the remaining links). The only solution $y = (y_{a_1}, y_{a_2}, y_{a_3})$ fulfilling these conditions is $y_{a_1} = y_{a_2} = y_{a_3} = \frac{1}{2}H$. Now consider any fixed link a. Jointly, the primary capacity y'_a and the backup capacity y''_a of this link carry exactly $\frac{1}{2}\sum_{d\in\mathcal{D}} h_d$ of flow. Define $\mathcal{D}' = \{d \in \mathcal{D} : \exists\ r = (p, q) \in \mathcal{T}_d, x_{dr} = 1 \wedge (a \in p \vee a \in q)\}$ and $\mathcal{D}'' = \mathcal{D} \setminus \mathcal{D}'$. Then, clearly, $\sum \mathcal{D}' = \frac{1}{2}H$ and hence $(\mathcal{D}', \mathcal{D}'')$ is a valid partition of $\mathcal{D}$. □

4.1.3 $\mathcal{NP}$-hardness proof (one demand, LP)

Now consider the linear relaxation of problem $\mathbb{P}$ (i.e., problem (4.1)) admitting all possible path-pairs, with the full single link failure scenario (i.e., all links can fail, one at a time), and with only one demand ($D = 1$). Below, it is proved that this variant (i.e., variant 1b, 2b, 3a, 4a) is also $\mathcal{NP}$-hard by reducing the 2Div-Path problem to it. 2Div-Path has been proved to be $\mathcal{NP}$-hard in [28].

Consider a directed graph and two pairs of its vertices (u_1, w_1) and (u_2, w_2). Assume that all four vertices are different. Problem 2Div-Path consists in answering a question whether there exist two arc-disjoint directed paths, one from u_1 to w_1, and the second from u_2 to w_2, in a given graph. Notice that the problem cannot be solved using an appropriately modified Suurballe algorithm [60], since the algorithm cannot assure that in the two resulting paths u_1 will be connected to w_1 and not to w_2. Moreover, notice that the problem is solvable in polynomial time in undirected graphs [58]. Thus, the proof cannot be applied to undirected cases.

Consider an instance $2D_{\mathcal{G}}$ of 2Div-Path for a given graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ and $u_1, u_2, w_1, w_2 \in \mathcal{V}$. The corresponding instance of problem (4.1) is denoted by $FI_{\mathcal{G}}$, and is modeled by means of a directed network $\mathcal{G}' = (\mathcal{V}', \mathcal{A}')$ depicted in Figure 4.2. The network consists of $8 + |\mathcal{V}|$ nodes and $13+|\mathcal{A}|$ links. The original graph $\mathcal{G}$ forms a subgraph of network $\mathcal{G}'$ and is depicted as a "cloud" in the figure. In the following proof only seven specific links will be used, denoted by $a_a, a_b, a_c, a_d, a_e, a_f, a_g$, and six specific nodes, denoted by u, w, u_1, u_2, w_1, w_2. Most of the unit capacity costs ξ_a are set to 0, and only $\xi_{a_c} = \xi_{a_d} = \xi_{a_e} = 1$. There is only one demand d from node u to node w, and $h_d = 1$.

Proposition 4.1.2. *The optimal objective of $FI_{\mathcal{G}}$ is less or equal to $\frac{3}{2}$ if and only if the answer to $2D_{\mathcal{G}}$ is YES.*

Proof. Consider $2D_{\mathcal{G}}$ and suppose that the two disjoint paths from the question exist. Denote these paths by $u_1 \to w_1$ and $u_2 \to w_2$. Now construct a solution to $FI_{\mathcal{G}}$ with the objective function value equal to $\frac{3}{2}$ by routing the demand using two primary/backup path-pairs. Each of these pairs carries flow $x = \frac{1}{2}$. The first primary flow traverses links a_a, a_c, and

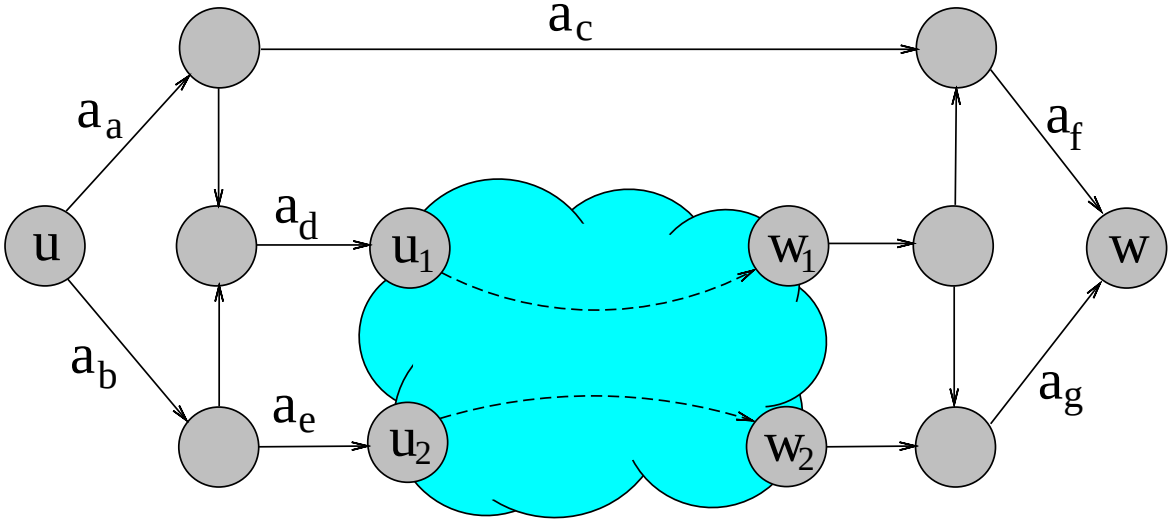


Figure 4.2: Network $\mathcal{N}$ resulting from graph $\mathcal{G}$.

a_f, while its corresponding backup flow is routed through links a_b, a_d, path $u_1 \to w_1$, and link a_g. The second primary flow travels through links a_b, a_e, path $u_2 \to w_2$, and link a_g; its corresponding backup flow passes links a_a, a_d, path $u_1 \to w_1$, and link a_f. Notice that each primary/backup flow pair is arc-disjoint. Moreover, the two primary paths are also arc-disjoint, and both backup flows use link a_d. Therefore, they can share the backup capacity y''_{a_d} on it. Thus, we have to reserve $\frac{1}{2}$ of primary capacity on links a_c and a_e, and $\frac{1}{2}$ of backup capacity on link a_d. It means that the objective of the corresponding solution of $FI_{\mathcal{G}}$ is $\frac{3}{2}$. Observe that this solution would not be feasible, if $u_1 \to w_1$ and $u_2 \to w_2$ were not arc-disjoint.

Now assume that $2D_{\mathcal{G}}$ has no solution, i.e., it is impossible to route two flows, the first from u_1 to w_1 and the second from u_2 to w_2, using arc-disjoint paths. Then, several cases can occur, including the existence of two other arc-disjoint paths: one from u_1 to w_2, and one and from u_2 to w_1. Below it is proved that in all such cases the objective of $FI_{\mathcal{G}}$ cannot be smaller than 2.

Define three sets of links, namely $\mathcal{A} = \{a_a, a_b\}$, $\mathcal{B} = \{a_c, a_d, a_e\}$, and $\mathcal{C} = \{a_f, a_g\}$. Notice that each of these sets cuts the network, so removing any of them results in a situation when a path between nodes u and w does not exist. Moreover, the links in the sets are directed in the way making it impossible to traverse more than one link from any of these sets using only one path. Therefore, each path from $\mathcal{P}_d$ has to pass exactly one link from each of the sets $\mathcal{A}, \mathcal{B}, \mathcal{C}$.

Divide (all possible) paths in $\mathcal{P}_d$ into groups with respect to the links they traverse. In effect, distinguish $|\mathcal{A}| \cdot |\mathcal{B}| \cdot |\mathcal{C}| = 12$ different groups; out of them only 6 are not empty. For instance, there is no path that traverses links a_a, a_c, and a_g. The available groups are shown in Table 4.2. For each of the groups calculate all possible groups of backup paths (certainly, a primary path and its backup path have to be arc-disjoint). The possible backup path groups are also shown in the table. Each group is identified by an appropriate triple. For instance, a group of paths that traverse links a_a, a_c and a_f is denoted by $a - c - f$.

Notice that any simple non-bifurcated solution of $FI_{\mathcal{G}}$ that uses just one primary/backup path-pair has the objective function equal to 2, as both the primary and the backup path have to traverse one link from set $\mathcal{B}$, and $\xi_a = 1$ only if $a \in \mathcal{B}$. Moreover, it is impossible to reduce

Table 4.2: Possible primary/backup path groups when $F_{\mathcal{G}}$ has no solution

Primary paths	Backup paths
$a-c-f$	$b-d-g$ or $b-e-g$
$a-d-f$	none
$a-d-g$	$b-e-f$
$b-e-f$	$a-d-g$
$b-e-g$	$a-c-f$
$b-d-f$	none
$b-d-g$	$a-c-f$

Table 4.3: Arc-disjoint primary path group pairs and their possible backup path groups

Primary path 1	Primary path 2	Backup path 1	Backup path 2
$a-c-f$	$b-e-g$	$b-d-g$ or $b-e-g$	$a-c-f$
$a-c-f$	$b-d-g$	$b-d-g$ or $b-e-g$	$a-c-f$
$a-d-g$	$b-e-f$	$b-e-f$	$a-d-g$

the cost corresponding to the reserved primary capacity. Therefore, in order to improve the solution, it is inevitable to find a way to reduce the cost corresponding to the reserved backup capacity. It can be done by analyzing all possible arc-disjoint pairs of primary paths, as backup capacity can be shared only by backup flows that belong to primary/backup path-pairs whose primary flows cannot fail simultaneously. In the considered case (single failures) this means that the primary paths have to be arc-disjoint. All possible arc-disjoint primary paths, together with all possible backup paths protecting them, are shown in Table 4.3.

It is clearly visible in the table that when $2D_{\mathcal{G}}$ has no solution it is impossible to find two arc-disjoint primary/backup path-pairs in such a way that the primary paths of these pairs are also arc-disjoint, and their backup paths traverse the same link $a \in \mathcal{B}$. Therefore, when $2D_{\mathcal{G}}$ has no solution (or its solution is not known) the cost of $FI_{\mathcal{G}}$ cannot be lower than 2. □

4.1.4 Complexity of approximation schemes

From the proof of Proposition 4.1.2 it is possible to draw conclusions concerning complexity of various approximation schemes that solve the 1b, 3a, 4a variant (LP, single failures, one demand) of problem $\mathbb{P}$. When a solution to an instance $2D_{\mathcal{G}}$ of 2Div-Path exists (and can be found), the objective of a solution to the corresponding instance $FI_{\mathcal{G}}$ is not greater than $\frac{3}{2}$. On the other hand, when a solution to $2D_{\mathcal{G}}$ is not known, solutions to $FI_{\mathcal{G}}$ cannot use such a path-pair. Therefore, the corresponding objectives must be equal at least to 2. Thus, since there is no way to solve 2Div-Path in polynomial time, any approximation to

the considered variant of $\mathbb{P}$ better than a $\frac{4}{3}$-approximation has to be $\mathcal{NP}$-hard. Note that the same reasoning can be applied also to the 1b, 3a, 4b variant (LP, single failures, many demands).

From the proof of Proposition 4.1.1 (IP, single failures, many demands) it is also possible to draw some conclusions concerning complexity of approximation schemes. Suppose that the answer to the considered instance of the PARTITION problem ($PA_{\mathcal{H}}$) is NO. In such a case, the quality of a solution to the corresponding instance $FI_{\mathcal{H}}$ depends on how precise we can approximate a solution to $PA_{\mathcal{H}}$. Note that PARTITION is equivalent to a special case of the SUBSET SUM problem, for which Kellerer et al. in [38] presented a fully polynomial approximation scheme that solves it within accuracy ε in time $O(\min\{n \cdot \frac{1}{\varepsilon}, n + \frac{1}{\varepsilon^2}\log(\frac{1}{\varepsilon})\})$. Therefore, it is impossible to conclude from the proof of Proposition 4.1.1 that problem $\mathbb{P}$ in the considered variant *cannot* be approximated within a given accuracy ε in polynomial time. That is why, it is worth to show how to modify the proof of Proposition 4.1.2 to cover the version of the problem considered in Proposition 4.1.1.

The modification is simple. Consider set $\mathcal{D}$ of demands instead of just one demand d. Consider a special case when demands from the set $\mathcal{D}$ can be divided into two subsets $\mathcal{D}'$ and $\mathcal{D}''$, such that $\sum_{d\in\mathcal{D}'} h_d = \sum_{d\in\mathcal{D}''} h_d$. Without loss of generality we can assume that $\sum_{d\in\mathcal{D}} h_d = 1$. Now treat the set $\mathcal{D}$ as one demand that can be split into halves. Note that in the proof of Proposition 4.1.2 one demand is equally split into two flows. Therefore, it is possible to apply the reasoning from that proof also to the case considered in Proposition 4.1.1. That is why, all polynomial approximations to the 1a, 3a, 4b variant (IP, single failures, many demands) of problem $\mathbb{P}$ also cannot be better than a $\frac{4}{3}$-approximation.

For variants 3b (multiple failures) of $\mathbb{P}$, the problem is straightforward, as it was proved in [34] that finding a pair of failure-disjoint paths, when multiple failures occur, is $\mathcal{NP}$-hard. Therefore, finding any feasible solution to the considered problem is $\mathcal{NP}$-hard. That is why, all approximation schemes that give any guarantee concerning the quality of a solution have to be $\mathcal{NP}$-hard. That concerns both LP and IP variants.

The results concerning the complexity of approximation schemes that solve different variants of the problem $\mathbb{P}$ are presented in Table 4.4.

4.1.5 Path generation

The pricing problem for the bifurcated FI-nSR (denoted by PRICE-FI-NSR) consists in finding, for each demand $d \in \mathcal{D}$, a pair of failure-disjoint paths $r = (p, q)$ from u_d to w_d

Table 4.4: Overview of the complexity of approximation schemes

		single failures	multiple failures
$D = 1$	LP, all paths	$< \frac{4}{3}$-approximation is $\mathcal{NPH}$	any approximation is $\mathcal{NPH}$
	IP, predefined paths	$\mathcal{P}$	any approximation is $\mathcal{NPH}$
$D > 1$	LP, all paths	$< \frac{4}{3}$-approximation is $\mathcal{NPH}$	any approximation is $\mathcal{NPH}$
	IP, predefined paths	$< \frac{4}{3}$-approximation is $\mathcal{NPH}$	any approximation is $\mathcal{NPH}$

minimizing the quantity (see also [51])

$$\langle r \rangle = \sum_{a \in p} \xi_a + \sum_{a \in q} \Big(\sum_{s \in \bar{\mathcal{S}}_p} \pi_a^{s\star} \Big) \tag{4.2a}$$

where $\pi_a^{s\star}$ are the optimal dual variables corresponding to constraints (4.1d). Notice that in (4.2a) the link metrics for calculating the length of the primary path $p \in \mathcal{P}_d$ are equal to the true unit link costs ξ_a, while the link metrics for calculating the length of the backup path $q \in \mathcal{Q}_p$ are given by the dual cost $\sum_{s \in \bar{\mathcal{S}}_p} \pi_a^{s\star}$. Because the bifurcated version of FI-nSR is $\mathcal{NP}$-hard for all failure scenarios, its pricing problem cannot be polynomial. In fact Price-FI-nSR is $\mathcal{NP}$-hard, as demonstrated (for the single link failure case) in [59] by reduction from 3-Sat (see [29]). Although the pricing problem Price-FI-nSR is $\mathcal{NP}$-hard, FI-nSR can in practice be effectively solved even for large networks by path-pair generation, as discussed for instance in [56].

4.2 FI-SR: Stub-Release

The case considered in this section assumes failure independent flow restoration using stub-release (SR), i.e., when surviving but unused working capacity can be reused for backup flows in failure situations.

The use of SR is typical for traffic layers in communication networks, for example for the MPLS sub-layer in IP networks. The non-bifurcated problem for FI-SR assumes that for each $d \in \mathcal{D}$, its entire demand volume is allocated to a single primary path, and in each failure state that affects the selected primary path, the flow is moved to another, failure-independent single restoration path.

Problem FI-SR

$$\text{minimize } F(y) = \sum_{a \in \mathcal{A}} \xi_a y_a, \tag{4.3a}$$

$$\sum_{r \in \mathcal{T}_d} x_r = 1, \qquad d \in \mathcal{D}, \tag{4.3b}$$

$$\sum_{d \in \mathcal{D}} h_d\Big(\sum_{r \in \mathcal{T}'_{ad} \wedge s \in \mathcal{S}_p} x_r + \sum_{r \in \mathcal{T}''_{ad} \wedge s \in \bar{\mathcal{S}}_p} x_r\Big) \leqslant y_a, \qquad a \in \mathcal{A},\ s \in \mathcal{S}_a, \tag{4.3c}$$

$$x_{dr} \in \{0, 1\}, \qquad d \in \mathcal{D},\ r \in \mathcal{T}_d, \tag{4.3d}$$

$$y_a \geqslant 0, \qquad a \in \mathcal{A}. \tag{4.3e}$$

4.2.1 Complexity overview

The complexity of FI-SR with respect to criteria presented in Section 4.1 is identical to the complexity of FI-nSR. This fact will be proved in the two following sections. An overview of the complexity can be seen in Table 4.1.

4.2.2 Bifurcated flows

The bifurcated version of FI-SR is obtained in the same way as for FI-nSR, leading to a linear relaxation of (4.3), admitting as before, for every $d \in \mathcal{D}$, several non-zero primary flows and several backup flows protecting each primary flow. This version is also $\mathcal{NP}$-hard. To prove that, a construction proving $\mathcal{NP}$-hardness of FI-nSR can be used. It is possible because in fact stub-release cannot be effectively used when only state-independent restoration is allowed, single failures of all links are admitted, and one demand is under consideration.

Lemma 4.2.1. *Stub-release cannot be effectively used in the presented scenario.*

Proof. Consider two flows denoted by α and β consisting of their primary and backup paths denoted by α_p, β_p, α_b, and β_b. Assume now that stub-release is effectively used, and the path α_b uses, through stub-release, the capacity released by the path β_p. Note that the path α_b cannot use the capacity released by the path α_p, as they have to be failure disjoint.

Because of the state-independent restoration assumption, the path β_p has to fail each time the path α_p fails, as it has been assumed that (single) failures of all links occur. In this case, the path β_p has to consist of at least all the links of the path α_p. Since only one demand is considered, flows α and β origin and terminate in the same nodes. Therefore, in order to fulfill the previous observations, the paths α_p and β_p have to be the same.

Because of the state-independent restoration, the paths α_p and α_b have to be disjoint. It means that also the path α_b and the path β_p have to be disjoint, as the paths α_p and β_p are the same.

On the other hand, when the path α_b is supposed to use the capacity released by the path β_p those paths cannot be disjoint. That leads to a contradiction, because they cannot be disjoint and not disjoint simultaneously. Therefore, stub-release cannot be effectively used in the considered scenario. □

This fact allows to use the same construction as in Section 4.1 in order to prove the $\mathcal{NP}$-hardness of the stub-release case.

4.2.3 Non-bifurcated flows

The difficulty of this problem is similar to the difficulty of FI-nSR. For scenarios with multiple failures, the problem is $\mathcal{NP}$-hard already for one demand, as finding a pair of failure disjoint paths is difficult. In the case of multiple demands, problem Partition can be reduced to the considered problem, as demonstrated for FI-nSR in Section 4.1. When only one demand is considered, and only single failures are admitted the problem, as FI-nSR, can be solved in polynomial time. However, this time it is impossible to use the Suurballe algorithm [60] directly, as it is not able to take the stub-release into account. Therefore, another polynomial algorithm is needed. It will be described in this section.

The considered problem (i.e., FI-SR with one demand, single failures, and non-bifurcated flows), denoted by Failure-Disjoint Paths, is a generalization of the Shortest Path problem and the 2 Arc-Disjoint Path problem. In fact polynomial time algorithm solving those two problems (e.g., Dijkstra's algorithm [22] solving Shortest Path, and Suurballe's algorithm [60] solving 2 Arc-Disjoint Path) are used as subroutines in the presented approach solving Failure-Disjoint Paths.

Assume two kinds of arcs: perfectly reliable arcs that do not fail, and unreliable arcs that do fail. Suppose that the two paths can share only the reliable arcs, so a pair of failure-disjoint paths is considered. The Minimum Cost Flow problem can cover the notion of reliability by setting capacities of reliable arcs to infinity and capacities of other arcs to 1. However, in this section under consideration is the problem of finding a shortest pair of failure-disjoint paths, where the cost of any reliable arc that is used by both paths is counted only once and not twice like in the Minimum Cost Flow context.

Consider a graph presented in Figure 4.3 with arc costs as depicted in the picture and reliable arcs denoted by thick arrows. Assume that u is the source, and w is the sink. The optimal solution to 2 Arc-Disjoint Path has the cost equal to 11, and consists of paths (u, v_1, w) and (u, v_3, w).

After adding the notion of reliability, an optimal solution of Minimum Cost Flow is given by sending two units of flow through paths (u, v_1, v_3, w) and (u, v_3, w). The cost is then equal to 10. Finally, the optimal solution to Failure-Disjoint Paths costs 8 and consists of paths (u, v_1, w) and (u, v_1, v_3, w).

If all arcs are reliable then Failure-Disjoint Paths is equivalent to Shortest Path. On the other hand, if the graph does not contain reliable arcs then the it is equivalent to 2 Arc-Disjoint Path. Thus Failure-Disjoint Paths can be seen as a generalization of

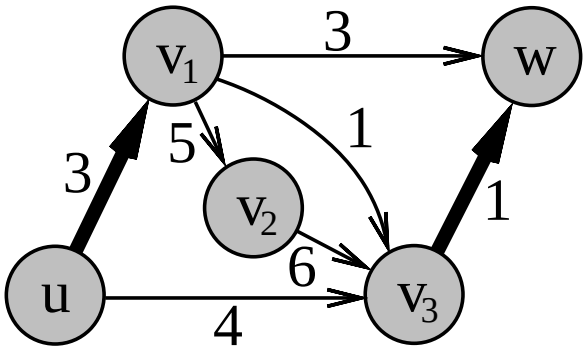


Figure 4.3: Network exposing differences between problems.

those two problems.

Moreover, by the max-flow min-cut theorem, it is easy to see that the problem is equivalent to computing a minimum cost set of arcs inducing a graph containing at least one path from u to w when there are no failures, and also one path for any failure of an unreliable arc.

Lemma 4.2.2. *There exists an optimal solution to* Failure-Disjoint Paths *not containing directed cycles.*

Proof. If the solution contains a directed cycle, then a flow on this cycle can be decreased, and some arcs of the cycle can be eliminated. This does not decrease the total flow carried from u to w, and does not increase the cost. By repeating this operation a solution without directed cycles is obtained. □

Now assume that the paths share L (reliable) arcs. Two ordered collections $\mathcal{L} = (a_1, a_2, \ldots, a_L)$ and $\mathcal{L}' = (a'_1, a'_2, \ldots, a'_L)$ are introduced. They define the order of the shared arcs from the point of view of the first and the second path, respectively.

Lemma 4.2.3. *There exists an optimal solution to* Failure-Disjoint Paths *where* $\mathcal{L} = \mathcal{L}'$.

Proof. Observe that if $\mathcal{L} \neq \mathcal{L}'$ then the solution contains directed cycles. According to Lemma 4.2.2 there is at least one optimal solution without directed cycles, thus at least one with $\mathcal{L} = \mathcal{L}'$ □

From Corollary 4.2.3 we can deduce that there exists an optimal solution containing a sequence $\mathcal{L}$ of reliable arcs shared by both paths, and traversed in the same order. Notice that if $\mathcal{L} = \emptyset$ then the solution is simply a pair of arc-disjoint paths. These arcs are connected with each other, and with u and w, by pairs of arc-disjoint paths. Such an optimal solution can be obtained using the following algorithm.

First, compute a shortest pair of arc-disjoint paths from each vertex i to each vertex j where i is either u or a sink of any reliable arc, and j is either w or a source of any reliable

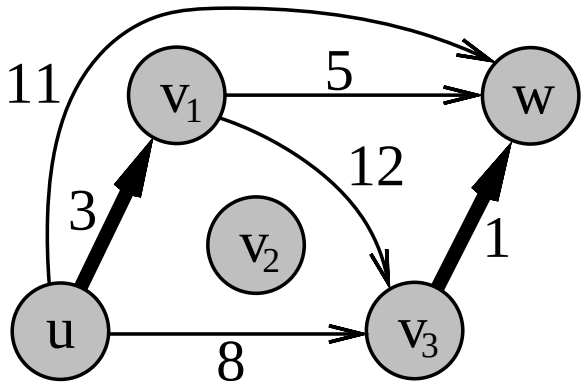


Figure 4.4: Modified graph corresponding to the graph of Figure 4.3.

arc. Next, eliminate all unreliable arcs from the graph. Then, for each computed pair of arc-disjoint paths from the first phase, one arc from the source of the pair to the sink of the pair is added to the graph. The costs ξ_a of those additional arcs are equal to the computed costs of the corresponding pairs of arc-disjoint paths. The resulting graph is called a *modified graph*.

Consider again the graph of Figure 4.3. A corresponding modified graph is depicted in Figure 4.4. Reliable arcs remained unchanged, while all unreliable arcs have been replaced by arcs that correspond to shortest pairs of arc-disjoint paths. For instance, an arc (u, v_3) corresponds to paths (u, v_1, v_3) and (u, v_3), and its cost is equal to a cost of the pair it represents. Notice that there are no pairs of arc-disjoint paths from or to vertex v_2. Therefore, this vertex is isolated in the modified graph.

In the resulting modified graph, which consists of pairs of arc-disjoint paths (represented by newly added arcs) and of reliable arcs a shortest path is computed. In the considered example the shortest path is (u, v_1, w), and it costs 8.

Consider a set $\mathcal{L} = (a_1, a_2, \ldots, a_L)$ (reliable arcs set) consisting of reliable arcs selected by the shortest path algorithm in the previous phase of the algorithm, and a collection of sets $\mathcal{A}_h$ (path pairs sets), where $h \in \mathcal{H} = \{\{u, a_1\}, \{a_1, a_2\}, \ldots, \{a_{L-1}, a_L\}, \{a_L, w\}\}$, consisting of arcs that form pairs of arc-disjoint paths computed earlier and also selected in the previous phase of the algorithm, e.g., a set $\mathcal{A}_{\{u,a_1\}}$ consists of arcs that form the pair of arc-disjoint paths between u and the source of a_1, set $\mathcal{A}_{\{a_1,a_2\}}$ connects the sink of a_1 to the source of a_2, while set $\mathcal{A}_{\{a_L,w\}}$ connects the sink of a_L to w. In the example a reliable arc set is $\mathcal{L} = \{(u, v_1)\}$, and two path pair sets are $\mathcal{A}_{\{u,(u,v_1)\}} = \emptyset$ and $\mathcal{A}_{\{(u,v_1),w\}} = \{(v_1, w), (v_1, v_3), (v_3, w)\}$.

Now consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{A}')$, where $\mathcal{A}' = \bigcup_{h\in\mathcal{H}} \mathcal{A}_h \cup \mathcal{L}$. Define a cost of this graph as $\sum_{a\in\mathcal{A}'} \xi_a$. In the example $\mathcal{A}' = \{(u, v_1), (v_1, w), (v_1, v_3), (v_3, w)\}$, and a cost of $\mathcal{G}$ is 8.

Lemma 4.2.4. *The cost of $\mathcal{G}$ is a lower bound for the cost of the optimal solution of* Failure-Disjoint Paths.

Proof. According to Corollary 4.2.3 there exists an optimal solution that consists of either a pair of arc-disjoint paths from u to w or a set of reliable arcs connected by pairs of arc-disjoint paths. Such a solution cannot be cheaper than the cost of the graph $\mathcal{G}$. □

Lemma 4.2.5. *$\mathcal{G}$ contains a pair of failure-disjoint paths from u to w.*

Proof. Transform the graph $\mathcal{G}$ to a graph $\mathcal{T}$ by adding parallel arcs to those in $\mathcal{L}$. The graph $\mathcal{G}$ consists of the collection $\mathcal{L} = \{a_1, a_2, \ldots, a_L\}$ of shared arcs connected by pairs of arc-disjoint paths (sets $\mathcal{A}_h$). Therefore, we know that in $\mathcal{T}$ the value of the minimum cuts between u and the source of a_1, between the sink of a_i and the source of a_{i+1}, where $i = 1, 2, \ldots, L-1$, and between the sink of a_L and w is not less than 2 (remember that pairs of arc-disjoint paths between those pairs of vertices exist in $\mathcal{T}$). Moreover, the value of the minimum cut in $\mathcal{T}$ between the source of a_i and the sink of a_i, where $i = 1, 2, \ldots, L$, is also not less than 2 (all arcs in $\mathcal{L}$ are reliable, so they are doubled in $\mathcal{T}$). Therefore, it is obvious that the minimum cut in $\mathcal{T}$ between u and w cannot be smaller than 2. Using the max-flow min-cut theorem it is possible to conclude that the maximum flow in $\mathcal{T}$ from u to w cannot be smaller than 2. Since each unit of flow flowing through the network corresponds to a path, we get two paths from u to w. Moreover, only reliable arcs can be utilized by both paths, because only those arcs were doubled during the transformation. Therefore, a pair of failure-disjoint paths in $\mathcal{G}$ from u to w has to exist. □

Consider the graph $\mathcal{T}$ defined in the proof of the previous lemma. Recall that $\mathcal{T}$ is obtained from $\mathcal{G}$ by doubling the arcs of $\mathcal{L}$, where $\mathcal{G}$ and $\mathcal{L}$ are given by the previous phase of the algorithm. By computing a shortest pair of arc-disjoint paths in $\mathcal{T}$, a feasible solution of Failure-Disjoint Paths is obtained. The resulting pair of paths cannot cost more than the graph $\mathcal{G}$ itself, because it cannot use any arc that is not in $\mathcal{G}$. Therefore, according to Corollary 4.2.4, this solution is necessarily an optimal solution of Failure-Disjoint Paths.

Proposition 4.2.6. Failure-Disjoint Paths *can be solved in polynomial time.*

Proof. The problem can be solved using the presented algorithm. It requires, in the first phase, the computation of shortest pairs of arc-disjoint paths between a polynomial number of pairs of vertices. Then, a shortest path is computed to get $\mathcal{G}$ (and also $\mathcal{T}$). Finally, a shortest pair of arc-disjoint paths in $\mathcal{G}$ is computed which gives us the wanted optimal pair of failure-disjoint paths in G. Notice, that the last phase is needed only if some arcs have 0 costs. □

In the first phase of the algorithm, in the worst case, pairs of arc-disjoint paths are computed between all possible pairs of nodes. Then a shortest path is computed once in the second phase. The last phase consists in computing a shortest pair of arc-disjoint paths once. Thus the overall complexity of the algorithm is determined by the complexity of the first phase which is $O(|\mathcal{V}| \cdot |\mathcal{A}| \cdot \log_{(1+|\mathcal{A}|/|\mathcal{V}|)} |\mathcal{V}|)$, i.e., $|\mathcal{V}|$ times the complexity of the algorithm presented in [61] to compute pairs of arc-disjoint paths from one source to all other vertices in a graph.

4.2.4 Generalizations for non-bifurcated flows

Failure-Disjoint Paths can be extended in many ways. This section contains a few of its interesting generalizations.

Pair of failure-disjoint paths with cost depending on flow

Consider a natural generalization of the problem defined, and studied, earlier, where cost depends on flow. If an arc a is reliable, i.e., $a \in \mathcal{A}/\mathcal{S}$, then $\xi_a(0) = 0$, $\xi_a(1) = \xi_a^1$ and $\xi_a(2) = \xi_a^2$. In other words, if a is used by one path, then we have to pay ξ_a^1, and if both paths use a, we should pay ξ_a^2. Assume that $\xi_a^2 \geqslant \xi_a^1$. The unreliable arcs cannot be used by more than one path, so only ξ_a^1 has to be defined for $a \in \mathcal{S}$. The objective function is $\sum_{a\in\mathcal{A}} \xi_a(f_a)$.

Now, it is easy to see that this problem can be solved using the algorithm presented earlier. It is enough to use ξ_a^1 while computing shortest pairs of arc-disjoint paths in the first phase, and ξ_a^2 for reliable arcs while computing the shortest path in the modified graph. Since the correctness proof of the algorithm remains the same, details are skipped.

A general minimum cost flow problem

Consider a generalization of Failure-Disjoint Paths, denoted by General F-D Paths, which is as follows.

General failure-disjoint path problem

$$\text{minimize } F(f) = \sum_{a\in\mathcal{A}} \xi_a(f_a), \tag{4.4a}$$

$$\sum_{a\in\delta^+(i)} f_a - \sum_{a\in\delta^-(i)} f_a = 0, \qquad i \in \mathcal{V}\backslash\{u, w\}, \tag{4.4b}$$

$$\sum_{a\in\delta^+(u)} f_a - \sum_{a\in\delta^-(u)} f_a = k, \tag{4.4c}$$

$$f_a \in \{0, 1, \ldots, c_a\}. \qquad a \in \mathcal{A} \tag{4.4d}$$

Problem (4.4) has to satisfy Kirchhoff's law (4.4b, 4.4c), but k units of flow, instead of just 2, have to leave u. Constraint (4.4d) assures that not more than c_a paths utilize an arc a. Notice that constants c_a can be consider as capacities of arc, and a directed graph with capacities is called a network. Finally, costs of arcs are functions of f_a.

Notice that constraints (4.4d) can be directly integrated in the objective function by defining $\xi_a(f_a)$ in a suitable way, i.e., by setting $\xi_a(f_a) = \infty$ if $f_a \notin \{0, 1, \ldots, c_a\}$. However, it is kept for sake of clearness.

The objective function considered here is quite general. It is generally neither convex nor concave. Readers can refer to, for example, [1] for work on convex flow problems and to,

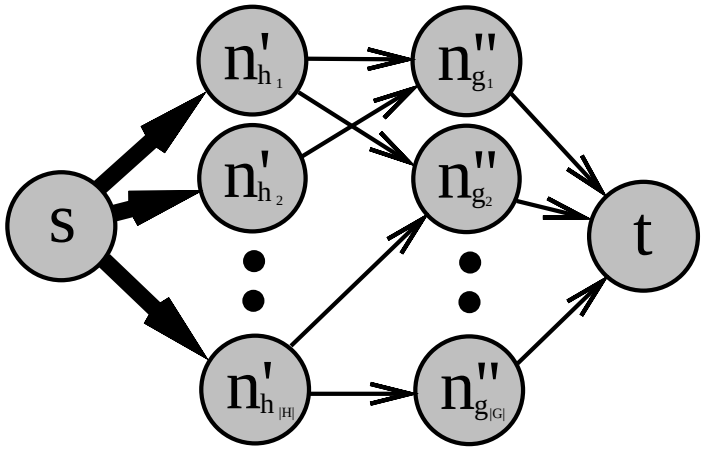


Figure 4.5: Sample instance of Infinite Capacities F-D Paths.

for example, [33] for results related to concave flows. It will be shown that General F-D Paths is $\mathcal{NP}$-hard by proving the $\mathcal{NP}$-hardness of one of its interesting special cases.

Infinite capacities

Consider the problem where all arcs can be either reliable (can accommodate an infinite number of paths) or unreliable (can accommodate only one path). It means that $c_a = 1$, for $a \in \mathcal{S}$, and $c_a \geqslant k$, otherwise. Assume that $\xi_a(f_a)$ is constant and equal to ξ_a when $f_a > 0$, for all $a \in \mathcal{A}$, still $\xi_a(0) = 0$. Denote the problem as Infinite Capacities F-D Paths.

Consider a universe $\mathcal{U}$ and a family $\mathcal{H}$ of subsets of $\mathcal{U}$. A cover is a subfamily $\mathcal{C} \subseteq \mathcal{H}$ of sets whose union is $\mathcal{U}$. In the Set Cover decision problem, the input is a universe $\mathcal{U}$, a family $\mathcal{H}$ and an integer C. The question is whether there is a cover of size C or less.

Consider an instance of Set Cover denoted by $SC_{\mathcal{U}}$. A corresponding instance of Infinite Capacities F-D Paths, denoted by $IC_{\mathcal{U}}$, is a network consisting of a source u, a sink w, $|\mathcal{H}|$ vertices of a first group and $|\mathcal{U}|$ vertices of a second group. Each vertex of the first group, denoted by n'_h, corresponds to one element h from the family $\mathcal{H}$, while each vertex of the second group, denoted by n''_g, corresponds to one element g from the universe $\mathcal{U}$. The source u is connected to all vertices of the first group using reliable arcs of cost 1. A vertex n'_h from the first group is connected, using an unreliable arc of zero cost, to a vertex n''_g from the second group, if $g \in h$. Finally, each vertex of the second group is connected to w using an unreliable arc of zero cost. In this network $|\mathcal{U}|$ paths from u to w are to be found. A sample instance of $IC_{\mathcal{U}}$ can be seen in Figure 4.5.

It is now quite simple to see that a cover of size C exists for $SC_{\mathcal{U}}$ if and only if problem $IC_{\mathcal{U}}$ has a solution whose cost is less than or equal to C. Since the reduction is polynomial,

so Infinite Capacities F-D Paths is $\mathcal{NP}$-hard.

Proposition 4.2.7. General F-D Paths *is $\mathcal{NP}$-hard.*

Note that Infinite Capacities F-D Paths is a special case of General F-D Paths, thus both are $\mathcal{NP}$-hard. Moreover, Set Cover is $\mathcal{NP}$-hard even when the maximum size of elements in $\mathcal{H}$ is 3. It means that General F-D Paths is $\mathcal{NP}$-hard even when capacities of arcs do not exceed 3.

Limited number of reliable arcs

Assume that the number of reliable arcs (reliable arcs can accommodate an infinite number of paths) is fixed and is not a part of the input. Moreover, $\xi_a(f_a)$ is constant and equal to ξ_a, if $f_a > 0$. Still $\xi_a(0) = 0$, for all $a \in \mathcal{A}$. In such a situation the problem is in $\mathcal{P}$, because it can be solved in polynomial time by enumerating all possibilities, i.e., solving Minimum Cost Flow for all possible vectors $x = (x_{a_1}, x_{a_2}, \ldots, x_{a_N})$, where N is the number of reliable arcs and $x_{a_n} = \{0, 1\}$. For each vector x the network is modified as follows. If $x_a = 1$ then $c_a = k$ and ξ_a is reduced to 0. If $x_a = 0$ then arc a is deleted from the graph. After solving Minimum Cost Flow in such a network, the result is increased by $\sum_{i=1}^{N} \xi_{a_i} x_{a_i}$, which is the cost of using (if $x_a = 1$) reliable arcs. The solution with the smallest modified cost is the optimal solution of the considered version of General F-D Paths.

In fact, the problem is still polynomial even if the cost function is general, and the number of arcs that can carry more than one unit of flow is limited. Indeed, if the value of the flow on these N arcs is considered, the total number of possibilities is less than $(k+1)^N$ which is polynomial if N is constant.

Acyclic graphs

Assume that the number of paths k is given, and is not a part of the input. The cost function is the most general one: $F(f) = \sum_{a \in \mathcal{A}} \xi_a(f_a)$. Notice that this function is generally neither convex nor concave. In such a situation General F-D Paths is polynomial for acyclic graphs. It will be proved now.

Assume that $\mathcal{N}$ is acyclic, and at least k units of flow can be carried from u to w. If the last is not possible, then General F-D Paths does not have a solution.

Sort $\mathcal{A}$ in topological order starting from u and ending at w. This sorted sequence will be denoted by $\mathcal{V}' = (v_1, v_2, \ldots, v_{|\mathcal{V}'|})$. Note that $v_1 = u$, $v_{|\mathcal{V}'|} = w$ and $\mathcal{V}' \subseteq \mathcal{V}$ (the sequence does not have to contain all the vertices of the network). Define a *state* as a sorted (according to the topological order $\mathcal{V}'$) multiset of k vertices $\mathcal{M} = (n_1, n_2, \ldots, n_k)$. A number of occurrences of a vertex v in a multiset $\mathcal{M}$ is denoted by $|\mathcal{M}_v|$. Note that $\sum_{v \in \mathcal{M}} |\mathcal{M}_v| = k$. Each state

$\mathcal{M}$ points to an associated optimal solution that routes k paths from u to vertices in $\mathcal{M}$ satisfying the capacity constraints of the considered problem.

Now divide all possible states into $|\mathcal{V}'|$ groups ($\mathcal{V}'$ is the topological order mentioned before) in a way that a state $\mathcal{M}$ belongs to a group $\mathcal{G}_i$, $1 \leqslant i \leqslant |\mathcal{V}'|$, if $v_i \in \mathcal{M}$ and $v_j \notin \mathcal{M}$, for all $j > i$ ($v_i, v_j \in \mathcal{V}'$). Call a state $\mathcal{M}'$ ($\mathcal{M}' \in \mathcal{G}_j$) a *predecessor* of a state $\mathcal{M}$ ($\mathcal{M} \in \mathcal{G}_i$) if:

- $j < i$,
- $|\mathcal{M}'_{v_l}| \geqslant |\mathcal{M}_{v_l}|$, for all $l < i$,
- if $|\mathcal{M}'_{v_l}| > |\mathcal{M}_{v_l}|$ then an arc (v_l, v_i) exists, and $c_{(v_l,v_i)} \geqslant |\mathcal{M}'_{v_l}| - |\mathcal{M}_{v_l}|$, for all $l < i$.

It is possible to build a *state graph* that consists of a set of vertices representing states and a set of arcs representing predeceasing relations, i.e., there is an arc from $\mathcal{M}'$ to $\mathcal{M}$, if $\mathcal{M}'$ is a predecessor of $\mathcal{M}$. The cost of this arc is given by $\xi_{(\mathcal{M}',\mathcal{M})} = \sum_{v \in \mathcal{M}',(v,v_i) \in \mathcal{A}} \xi_{(v,v_i)}(|\mathcal{M}'_v| - |\mathcal{M}_v|)$. Given an optimal solution of state $\mathcal{M}'$, $\xi_{(\mathcal{M}',\mathcal{M})}$ represents the additional cost that is incurred by extending this solution, i.e., by sending $|\mathcal{M}'_v| - |\mathcal{M}_v|$ units of flow on arc $(v, v_i) \in \mathcal{A}$, for each $v \in \mathcal{V}$. Notice that the state graph does not contain directed cycles.

Going back to the example in Figure 4.3 where $k = 2$. The vertices are already sorted there topologically from left to right, i.e., $\mathcal{V}' = (u, v_1, v_2, v_3, w)$. The corresponding state graph can be seen in Figure 4.6. One can see that state (v_2, v_2) does not have any predecessor. This simply means that it is not possible to have 2 paths from u to v_2 satisfying capacity constraints. Moreover, states depicted in grey are irrelevant from the point of view of the algorithm, as there is no path connecting them to a state $(w, w, \ldots, w)$. Therefore, for sake of clearness of the figure, all arcs entering those irrelevant states are also gray, and their costs are not depicted.

Consider now an optimal solution $\mathcal{F}^\star$ of state $\mathcal{M} \in \mathcal{G}_i$, where $i > 1$. Remember that this solution is a set of k paths such that the number of paths ending at $v \in \mathcal{M}$ is exactly $|\mathcal{M}_v|$. By considering predecessors v of v_i, for all $|\mathcal{M}_{v_i}|$ paths reaching v_i and increasing by one values of corresponding $|M_v|$, we get a new state $\mathcal{M}'$ that is a predecessor of a state $\mathcal{M}$. Moreover, a feasible solution of this state is obtained. The cost of this solution is equal to the cost of $\mathcal{F}^\star$ minus $\sum_{v \in \mathcal{M}',(v,v_i) \in \mathcal{A}} \xi_{(v,v_i)}(|\mathcal{M}'_v| - |\mathcal{M}_v|)$. This feasible solution should also be optimal for state $\mathcal{M}'$, since otherwise it is possible to build another solution of $\mathcal{M}$ with a cost less than the cost of $\mathcal{F}^\star$. Said another way, the cost of an optimal solution of $\mathcal{M}$ can be computed by examining optimal costs of the predecessors of $\mathcal{M}$. Consequently, an optimal solution of the problem can be obtained by computing a shortest path from state $(u, \ldots, u)$ to state $(w, \ldots, w)$.

In the example the shortest path is $((u, u), (v_1, v_1), (v_1, v_3), (w, w))$. It costs 8, thus it is optimal.

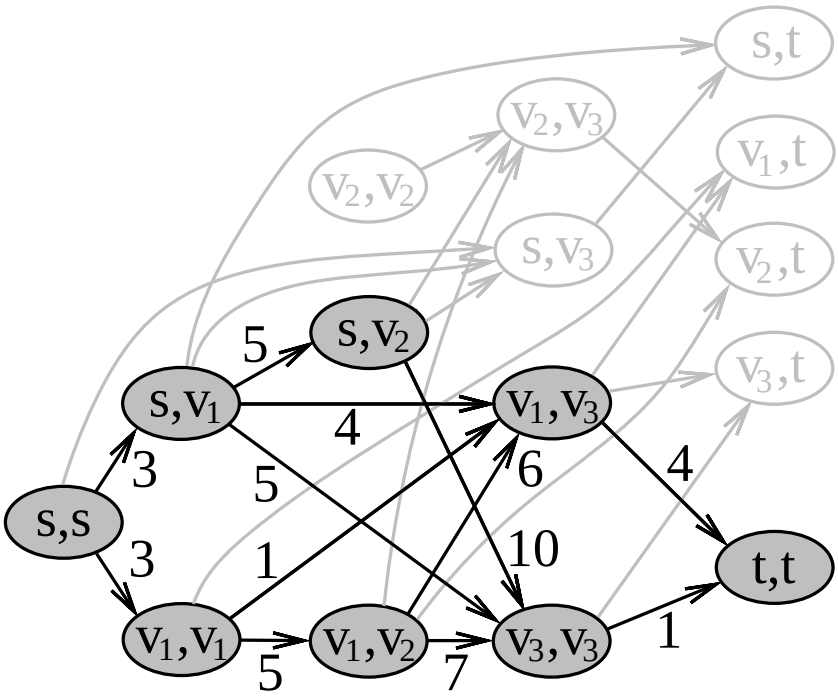


Figure 4.6: State graph.

Proposition 4.2.8. *If the number of paths k is fixed then* General F-D Paths *in acyclic graphs is in $\mathcal{P}$.*

Proof. This version of General F-D Paths can be solved using any shortest path algorithm in acyclic graphs. The number of possible states is $O(|\mathcal{V}|^k)$, and the maximum number of arcs of the state graph is $O(|\mathcal{V}|^{2k})$. Therefore, the overall complexity of the algorithm is $O(|\mathcal{V}|^{2k})$. □

A similar dynamic programming approach for a minimum concave cost network flow problem in acyclic uncapacitated networks was proposed in [16].

Large capacities

Consider General F-D Paths, where $c_a \geqslant \frac{k}{2}$ and $\xi_a(f_a) = \xi_a$, for all $a \in \mathcal{A}$, still $\xi_a(0) = 0$. In such a case the algorithm solving FI-SR presented earlier in this section can be used. This fact will be proved now.

Take any solution to the considered General F-D Paths, and a network $\mathcal{N} = (\mathcal{V}', \mathcal{A}')$ induced by f_a variables of the solution and capacities equal to those variables. Now decrease the capacities in the way that $c_a = 1$ if $f_a < k$, and $c_a = 2$ if $f_a = k$. This modified network will be refereed to as $\mathcal{N}'$.

Lemma 4.2.9. *Two units of flow can be carried from u to w in $\mathcal{N}'$.*

Proof. Using the max-flow min-cut theorem it is possible to deduce that in $\mathcal{N}$ the minimum cut has to consist of at least one arc of capacity k or at least two arcs of any capacity. As all arcs of capacity k are transformed to arcs of capacity 2 in $\mathcal{N}'$, the minimum cut in $\mathcal{N}'$ cannot be smaller than 2. It means that the maximum flow also cannot be smaller than 2. □

Lemma 4.2.10. *There exists an optimal solution to the considered problem that consists of two sets of paths $\mathcal{P}$ and $\mathcal{P}'$, such that $|\mathcal{P}| = \lceil \frac{k}{2} \rceil$, $|\mathcal{P}'| = \lfloor \frac{k}{2} \rfloor$, and paths from the same set are identical.*

Proof. Consider any optimal solution to the problem, and its corresponding $\mathcal{N}'$ network. The cost of this network ($\sum_{a \in \mathcal{A}'} \xi_a$) is equal to the cost of the considered solution. From Lemma 4.2.9 we know that in $\mathcal{N}'$ there exists a pair of paths that can share arcs of capacity k only. By putting $\lceil \frac{k}{2} \rceil - 1$ additional paths identical to the first path, and $\lfloor \frac{k}{2} \rfloor - 1$ additional paths identical to the second, a new solution is obtained. The solution is feasible, as all capacity constraints are satisfied. Moreover, it cannot be more expensive than the previous solution, thus it is also optimal. □

The shortest pair of failure-disjoint set of paths mentioned in Lemma 4.2.10 can be obtained by running the algorithm for Failure-Disjoint Paths in a graph where an arc a belongs to $\mathcal{S}$, if $x_a < k$.

Proposition 4.2.11. General F-D Paths *is in $\mathcal{P}$, if $c_a \geqslant \frac{k}{2}$ and $\xi_a(f_a) = \xi_a$ ($\xi_a(0) = 0$), for all $a \in \mathcal{A}$.*

Proof. An optimal solution to this problem can be obtained by having $\lceil \frac{k}{2} \rceil$ identical paths to the first path returned by the algorithm for Failure-Disjoint Paths and $\lfloor \frac{k}{2} \rfloor$ identical paths to the second path returned by the algorithm. The procedure is polynomial, thus the problem is in $\mathcal{P}$. □

4.2.5 Path generation

For each demand $d \in \mathcal{D}$, the pricing problem Price-FI-SR for the bifurcated version of FI-SR consists in finding a pair of failure-disjoint paths $r = (p, q)$ from u_d to w_d minimizing

$$\langle r \rangle = \sum_{a \in p} \Big(\sum_{s \in \mathcal{S}_p} \pi_a^{s\star} \Big) + \sum_{a \in q} \Big(\sum_{s \in \bar{\mathcal{S}}_p} \pi_a^{s\star} \Big) \tag{4.5a}$$

where optimal dual variables $\pi_a^{s\star}$ correspond to (4.3c). As discussed in [51], already minimizing the first sum under a single link failure scenario is $\mathcal{NP}$-hard; hence, solving the whole problem Price-FI-SR is $\mathcal{NP}$-hard.

Chapter 5

Path Restoration Problems with Failure-Dependent Restoration

In this chapter failure-dependent path restoration (FD) is considered. This means that the backup routing of a demand depends on the particular failure state.

5.1 FD-nSR: no Stub-Release

In the case without stub-release (nSR), the non-compact MIP formulation of the non-bifurcated version of the FD-nSR problem is as follows.

Problem FD-nSR

$$\text{minimize } F(y) = \sum_{a \in \mathcal{A}} \xi_a (y'_a + y''_a), \tag{5.1a}$$

$$\sum_{r \in \mathcal{W}_d} x_{dr} = 1, \qquad d \in \mathcal{D}, \tag{5.1b}$$

$$\sum_{d \in \mathcal{D}} h_d \sum_{r \in \mathcal{W}'_{ad}} x_{dr} \leqslant y'_a, \qquad a \in \mathcal{A}, \tag{5.1c}$$

$$\sum_{d \in \mathcal{D}} h_d \sum_{r \in \mathcal{W}''_{ads} \wedge s \in \bar{\mathcal{S}}_p} x_{dr} \leqslant y''_a, \qquad a \in \mathcal{A},\ s \in \mathcal{S}_a \setminus \{\mathcal{O}\}, \tag{5.1d}$$

$$x_{dr} \in \{0, 1\}, \qquad d \in \mathcal{D}, r \in \mathcal{W}_d, \tag{5.1e}$$

$$y''_a \geqslant 0, \qquad a \in \mathcal{A}. \tag{5.1f}$$

As explained in Chapter 2, the set $\mathcal{W}_d$, defined for each $d \in \mathcal{D}$, contains sequences $r = (p, q_s : s \in \bar{\mathcal{S}}_p)$ of paths from u_d to w_d with the property that path q_s works in state $s \in \bar{\mathcal{S}}_p$. These sequences are predefined. Therefore, the set $\mathcal{W}'_{ad}$ of all sequences $r = (p, q_s : s \in \bar{\mathcal{S}}_p)$ such that $a \in p$, and the set $\mathcal{W}''_{ads}$ of all sequences $r = (p, q_s : s \in \bar{\mathcal{S}}_p)$ such that $a \in q_s$ for $s \in \bar{\mathcal{S}}_p$ are known in advance and are used in constraints (5.1c) and (5.1d), respectively, to compute the load of the links in different states. Due to (5.1b), in an optimal solution for each $d \in \mathcal{D}$

there is exactly one $r \in \mathcal{W}_d$ with $x_{dr} = 1$, and this particular sequence is used to realize the demand d in the non-bifurcated manner. Note that, due to (5.1c), $y'_a \geqslant 0$ for all $a \in \mathcal{A}$.

The non-bifurcated FD-nSR problem can be solved in polynomial time only when single links failures are admitted, and only one demand is considered. In such a case the reasoning of Section 3.2 can be used (showing that HS is a solution to non-bifurcated PD in the single link failure case), so any obtained solution can be transformed to another valid solution of the same or smaller cost that consists of two failure-disjoint paths. The only difference between PD and FD-nSR in this case is that in the latter case link capacities are reduced either to h_d or to $2h_d$ (due to the fact that backup flows can share capacity) before the actual procedure is invoked. Notice that the same reduction is performed for the former case in the first step of the algorithm.

In the case of multiple failures the problem becomes $\mathcal{NP}$-hard, as Vertex Cover (see Section 3.2) can be reduced to it. Also in this case it is possible to directly follow considerations presented for PD. In fact, for the network used in those considerations, problems PD and FD-nSR are equivalent.

When more than one demand is admitted the problem also becomes $\mathcal{NP}$-hard. In this case ideas presented for FI-nSR (Section 4.1) can be utilized, and an appropriate reduction from Partition can be used. The proof uses a network that consists of two nodes and three parallel directed links. In this network each possible simple path consists of only one link. Therefore, there is no difference between FI-nSR and FD-nSR, as each path can fail only in one failure state (single link failures are only considered).

Again, the bifurcated version of FD-nSR is obtained by relaxing the binary flow variables. In the bifurcated version the demand volume h_d, for each demand $d \in \mathcal{D}$, can be realized on several working paths, and each resulting primary flow can be protected by failure-dependent sets of several backup paths. The bifurcated version of FD-nSR is of polynomial complexity for single link failure scenarios, as a compact formulation can be easily written down in the node-link notation (see (3.4)). For multiple failures, however, problem FD-nSR is $\mathcal{NP}$-hard itself. It will be demonstrated in this section.

Proposition 5.1.1. *FD-nSR is $\mathcal{NP}$-hard when multiple failures are admitted.*

The proof uses essentially the same construction of network $\mathcal{N}$ as the proof of Proposition 3.2.1 given in Section 3.2. The only modifications are as follows.

- One additional node w and one additional directed link (w, v_0) are introduced into network $\mathcal{N}$.
- Link (w, v_0) is assigned the unit capacity cost equal to 1 ($\xi_{(w,v_0)} = 1$), and the rest of the links $t_1, t_2, \ldots, t_N, b_1, b_2, \ldots, b_N$ are assigned zero unit capacity costs ($\xi_{t_n} = \xi_{b_n} =$

$0, n = 1, 2, \ldots, N$).

- The considered demand starts in node w and terminates in node v_N.

Consider an undirected graph $\mathcal{G}$ with the set $\mathcal{W} = \{v_1, v_2, \ldots, v_N\}$ of N vertices. Let $\mathcal{E}$ be the family of all independent sets of $\mathcal{G}$. The resulting instance of Fractional Coloring (i.e., of problem FGC (3.5)) will be denoted by $FC_{\mathcal{G}}$.

The instance $FD_{\mathcal{G}}$ of problem FD-nSR (5.1) corresponding to $FC_{\mathcal{G}}$ is modeled by means of the resulting modified directed network $\mathcal{N} = (\mathcal{V}, \mathcal{A})$ and the single demand from node w to node v_N with volume equal to 1. In fact, in this special case problem FD-nSR can be equivalently transformed to the following LP.

Problem FDM (special case of FD-nSR)

$$\text{minimize } G(x) = \max_{s \in \mathcal{S}} \{1 - \sum_{p \in \mathcal{P}^s} x_p\}, \tag{5.2a}$$

$$\sum_{p \in \mathcal{P}} x_p = 1, \tag{5.2b}$$

$$x_p \geqslant 0, \qquad p \in \mathcal{P}. \tag{5.2c}$$

Observe that objective (5.2a) minimizes, over all failure states $s \in \mathcal{S}$, the maximum amount of the total flow failing in state s since this is the actual cost of protection capacity (the cost of primary capacity is always equal to 1). Certainly, for network $\mathcal{N}$ the objective (5.1a) of problem FD-nSR and the objective (5.2a) of problem FDM are related by the equality $F(y) = 1 + G(x)$. In the sequel assume that $FD_{\mathcal{G}}$ is an instance of problem (5.2) rather than (5.1).

Lemma 5.1.2. *The fractional chromatic number $\chi_f(\mathcal{G})$ of graph $\mathcal{G}$ is equal to K if and only if the cost of an optimal solution of problem $FD_{\mathcal{G}}$ is equal to $1 - \frac{1}{K}$.*

Proof. Let $\mathcal{Q}, x = (x_p : p \in \mathcal{Q})$ be an optimal solution of the instance $PDD_{\mathcal{G}}$ of problem PDD (3.3). Due to Lemma 3.2.2, $F(x)$ is equal to $N \cdot K$. Because $F(x) = \sum_{p \in \mathcal{Q}} \xi_p x_p = N \cdot \sum_{p \in \mathcal{Q}} x_p$ (recall that all paths $p \in \mathcal{P}$ have, by construction, the unit cost ξ_p equal to N), we have $\sum_{p \in \mathcal{Q}} x_p = K$. Putting $x'_p = \frac{x_p}{K}$, $p \in \mathcal{Q}$, we obtain a feasible solution of $FD_{\mathcal{G}}$ with $G(x') = \max_{s \in \mathcal{S}}\{1 - \sum_{p \in \mathcal{Q}^s} x'_p\} = 1 - \min_{s \in \mathcal{S}}\{\sum_{p \in \mathcal{Q}^s} \frac{x_p}{K}\} = 1 - \frac{1}{K} \min_{s \in \mathcal{S}}\{\sum_{p \in \mathcal{Q}^s} x_p\} = 1 - \frac{1}{K}$. The last equality follows from the fact that $\min_{s \in \mathcal{S}}\{\sum_{p \in \mathcal{Q}^s} x_p\} = 1$, because in any optimal solution of $PDD_{\mathcal{G}}$ one of the constraints (3.3b) must be active. Thus, $1 - \frac{1}{K}$ is an upper bound for the minimum cost of $FD_{\mathcal{G}}$.

Conversely, suppose that $\mathcal{Q}, x' = (x'_p : p \in \mathcal{Q})$ is an optimal solution of $FD_{\mathcal{G}}$ with $G(x') < 1 - \frac{1}{K}$. Then $\sum_{p \in \mathcal{Q}^s} x'_p > \frac{1}{K}$, for all $s \in \mathcal{S}$, and hence $\sum_{p \in \mathcal{Q}^s} x_p > 1, s \in \mathcal{S}$, where $x_p = K \cdot x'_p, p \in \mathcal{Q}$. This means that x is a feasible solution of $PDD_{\mathcal{G}}$ with $F(x) = N \cdot K$, where $K = \chi_f(\mathcal{G})$. Thus, due to Lemma 3.2.2, the solution x is optimal. This, however, is a contradiction because, again, a solution for which all constraints (3.3b) are not active cannot be optimal. Thus, $1 - \frac{1}{K}$ is also the lower bound for $FD_{\mathcal{G}}$, so $1 - \frac{1}{K}$ is in fact the optimal objective value for this problem. □

As PD, FD-nSR is $\mathcal{NP}$-hard already in the single-demand version.

Path generation

Path generation (and the pricing problem PRICE-FD-NSR), for a given demand $d \in \mathcal{D}$, for the bifurcated version of FD-nSR consists first of all in finding the shortest path, separately for each situation $s \in \mathcal{S} \setminus \{\mathcal{O}\}$, with respect to the optimal dual variables π_a^s corresponding to constraints (5.1d). Then, denoting the length of each such shortest path by $\lambda_d^{s\star}$, we look for the shortest primary path p from u_d to w_d minimizing

$$\sum_{a\in p} \xi_a + \sum_{s\in\bar{\mathcal{S}}_p} \lambda_d^{s\star}. \tag{5.3a}$$

Thus, the pricing problem PRICE-FD-NSR for FD-nSR is essentially the same as the pricing problem for PD, and therefore is polynomial for single link failures (as observed by several authors [50, 65]), and $\mathcal{NP}$-hard for multiple failures.

5.2 FD-SR: Stub-Release

The final case assumes failure dependent (FD) flow restoration using stub-release (SR). The corresponding non-bifurcated problem FD-SR is as follows.

Problem FD-SR

$$\text{minimize } F(y) = \sum_{a\in\mathcal{A}} \xi_a y_a, \tag{5.4a}$$

$$\sum_{r\in\mathcal{W}_d} x_{dr} = 1, \qquad d \in \mathcal{D}, \tag{5.4b}$$

$$\sum_{d\in\mathcal{D}} h_d\Big(\sum_{r\in\mathcal{W}'_{ad}\wedge s\in\mathcal{S}_p} x_{dr} + \sum_{r\in\mathcal{W}''_{ads}\wedge s\in\bar{\mathcal{S}}_p} x_{dr}\Big) \leqslant y_a, \qquad a \in \mathcal{A},\ s \in \mathcal{S}_a \tag{5.4c}$$

$$x_{dr} \in \{0,1\}, \qquad d \in \mathcal{D},\ r \in \mathcal{W}_d, \tag{5.4d}$$

$$y_a \geqslant 0, \qquad a \in \mathcal{A}. \tag{5.4e}$$

For the case of multiple failures and many demands, $\mathcal{NP}$-hardness of FD-SR can be proved using the construction presented in Section 5.1 for FD-nSR . The reason is that in both constructions all possible paths consist of only one link, and in effect stub-release cannot be used, as the only link that can be released in a particular situation fails.

The one demand and single failure case is polynomial, still a little bit more complicated. It can be proved by showing that any feasible solution to this case of FD-SR can be transformed to a solution of FI-SR of the same or lower cost. Follow the considerations for PD, and employ a modified version of the algorithm presented in Section 3.2 showing equivalence

of PD and HS for the non-bifurcated single-link failure case. Having a solution to FD-SR add parallel links to those that do not fail, and are present in the solution. Then, reduce the capacity of the used links to h_d, and exclude from the resulting network the links that do not belong to any of the $2h_d$-cuts (one by one). Finally, erase those added links that have not been erased in the previous phase, and whose counterparts are also still in the network. The resulting network cannot be more expensive than the original solution to FD-SR. Moreover, it is equivalent to a solution to FI-SR. Observe that the optimal solution to FI-SR can be found in polynomial time. The bifurcated relaxation (with continuous flow variables) of FD-SR is most likely $\mathcal{NP}$-hard. Still, the conjecture has not been proved so far.

Path generation

For each demand $d \in \mathcal{D}$, the pricing problem PRICE-FD-SR for the bifurcated version of FD-SR consists essentially in finding a primary path p from u_d to w_d minimizing

$$\langle p \rangle = \sum_{a \in p} \Big(\sum_{s \in \mathcal{S}_p} \pi_a^{s\star} \Big) + \sum_{s \in \bar{\mathcal{S}}_p} \lambda_d^{s\star} \tag{5.5a}$$

where quantities $\pi_a^{s\star}$ are optimal dual variables corresponding to constraints (5.4c), and $\lambda_d^{s\star}$ are the lengths of the shortest paths for situations $s \in \mathcal{S} \setminus \{\mathcal{O}\}$, calculated according to the optimal dual variables $\pi_a^{s\star}$ (as in problem PRICE-FD-NSR, see the previous section). It is demonstrated in [47] (by reduction from the Hamiltonian path problem [29]) and in [50] (by reduction from the max-cut problem [29]) that PRICE-FD-SR is $\mathcal{NP}$-hard already for single-link failures (see also [51]).

Chapter 6

Conclusions

In the first part of the thesis the problem of designing survivable transport networks was considered. Two methods (mechanisms) of providing survivability of transport connections against either single or multiple failures of links were examined: protecting connections with dedicated standby connections, i.e., path protection, and restoring failed connections using protection capacity of links, i.e., path restoration. A number of constraints that might arise in practice was considered: first, that all connections between any pair of nodes must use the same network path, i.e., non-bifurcated routing; next, that each connection must be restored along the same path independently of which failure occurs, i.e., failure-independent restoration; and finally, that the capacity of links occupied by failed connections can be used for restoration, i.e., stub-release-based restoration.

For each combination of the design constraints and the resilience methods the resulting network design problem was analyzed. It was shown how to formulate each of those optimization problems as either a linear or a mixed-integer programming multi-commodity flow problem; the mixed-integer formulations arise in the situations when the non-bifurcated routing of demands is required. In most of the cases the resulting formulation is a non-compact link-path formulation; a compact, node-link formulation is used only in two cases of the bifurcated routing: in the design of path protection with diverse paths and in the design of failure-dependent path restoration routing without stub-release-based restoration.

Analyzing the complexity of the problems, it is justified to conclude (see Table 6.1) that the majority of problems appear to be $\mathcal{NP}$-hard; only for single failures when only one demand is considered all problems are polynomial. For multiple failures actually all the problems are $\mathcal{NP}$-hard. For single failures the only problems that are not $\mathcal{NP}$-hard are the design of path protection in the case of the non-bifurcated routing and the design of failure-dependent path restoration without stub-release in the case of the bifurcated routing. The former problem can be solved with shortest path algorithms, while the latter, being one of few problems featuring a compact linear programming formulation, by linear programming methods.

Table 6.1: General overview of complexity

	single failures (SF)			multiple failures (MF)		
	bifurcated	non-bifurcated		bifurcated	non-bifurcated	
		$D = 1$	$D > 1$		$D = 1$	$D > 1$
HS	-	$\mathcal{P}$	-	-	$\mathcal{NPH}$	-
PD	$\mathcal{P}$	$\mathcal{P}$	-	$\mathcal{NPH}$	$\mathcal{NPH}$	-
FI-nSR	$\mathcal{NPH}$	$\mathcal{P}$	$\mathcal{NPH}$	$\mathcal{NPH}$	$\mathcal{NPH}$	$\mathcal{NPH}$
FI-SR	$\mathcal{NPH}$	$\mathcal{P}$	$\mathcal{NPH}$	$\mathcal{NPH}$	$\mathcal{NPH}$	$\mathcal{NPH}$
FD-nSR	$\mathcal{P}$	$\mathcal{P}$	$\mathcal{NPH}$	$\mathcal{NPH}$	$\mathcal{NPH}$	$\mathcal{NPH}$
FD-SR	?	$\mathcal{P}$	$\mathcal{NPH}$	?	$\mathcal{NPH}$	$\mathcal{NPH}$

All other design problems in the case of the bifurcated routing of demands are $\mathcal{NP}$-hard, and therefore they cannot have a compact formulation. Still, when the problem is $\mathcal{NP}$-hard then its non-compact LP formulation (in the link-path notation) exists. Certainly, when using such non-compact LP formulations, for exact solutions an appropriate candidate path lists $\mathcal{P}_d, d \in \mathcal{D}$ (or path-pair lists $\mathcal{T}_d, d \in \mathcal{D}$, or path-sequences lists $\mathcal{W}_d, d \in \mathcal{D}$) are needed in order to achieve the true optimum of the problem. As such lists are not known in advance they have to be generated using column generation. This, as already known, is in general $\mathcal{NP}$-hard as it involves $\mathcal{NP}$-hard pricing problems. Nevertheless, according to the experience, the use of non-compact LP formulations together with column generation is the only practically effective general method that yields (in vast majority of cases) optimal solutions to the considered $\mathcal{NP}$-hard resilient network design problems. This issue is discussed below using the results of [56].

In fact, all the $\mathcal{NP}$-hard pricing problems described in the previous chapters of the thesis can be formulated as mixed-integer programming (MIP) problems (such formulations are discussed in [51], see also [56, 59]) and solved by a MIP solver in the column generation process. It turns out, however, that this approach is in general not efficient because of excessive time spent in a MIP solver.

Alternatively, the pricing problems (which are in fact complicated shortest path problems with tricky constraints and multiple link metrics) can be solved using methods for Shortest Path Problem with Resource Constraints (SPPRC) [36]. A general SPPRC algorithm was used for FI-nSR in [59]. Still, as discussed in [56], it turns out that using general label-setting SPPRC algorithms for the considered pricing problems is not effective either. Fortunately, such label-setting algorithms can be properly adjusted. The way how to do it for FI-nSR and FD-SR is described in [56], where numerical examples illustrating the above observations are also given.

The lists of optimal paths obtained for the bifurcated versions of the considered optimization problems can then be used for solving their non-bifurcated counterparts by means of MIP solvers to maximally limit the number of binary variables used in the branch-and-bound process. Certainly, this would be an approximate approach as the optimal paths for the bifurcated problems do not necessarily contain all optimal paths for the non-bifurcated case. To solve the problems exactly, it would actually be required to use branch-and-bound schemes combined with pricing of variables.

The research field presented in the first part of the thesis still contains some interesting problems to solve. The computational complexity of FD-SR with bifurcated flows has been proved neither for the single failure case nor for the multiple failure case. What is more, $\mathcal{NP}$-hardness of FI-nSR and FI-SR with bifurcated flows has been proved by reducing the 2Div-Path problem to it. Notice that 2Div-Path is $\mathcal{NP}$-hard only in directed graphs. Thus, the complexity of both FI-nSR and FI-SR in undirected graphs remains unknown. Finally, the field contains the enormous number of different special cases and generalizations that can change the complexity of base problems.

Part II

Polyhedral model

Chapter 7

Notation and problem formulations

Consider a directed graph $G = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V}$ is a set of nodes and $\mathcal{A}$ is a set of arcs. The graph represents a backbone, and the arcs depict the unidirectional transmission links. For each arc $a \in \mathcal{A}$ an installed capacity $c_a \in \mathbb{R}_+$ and a routing cost $\xi_a \in \mathbb{R}_+$ for one unit of traffic are given.

Let $t = (t_{ij})_{i,j \in \mathcal{V}}$ be a vector of $\mathbb{R}^{|\mathcal{V}|(|\mathcal{V}|-1)}$ that specifies values of traffic demands (or capacity requirements) between pairs of nodes of $\mathcal{V}$. This vector will be called a *traffic matrix*. Note that the size of t can be smaller when it is assumed that $t_{ij} = 0$, for given $i, j \in \mathcal{V}$, and for all $t \in D$. To clarify the notation a demand between nodes i and j is called "demand ij", and its value is denoted by t_{ij}. The traffic matrix is supposed to be variable, and can be any point of a *traffic demand polytope* D. D is generally defined by some linear constraints involving the variables t_{ij}, for $i, j \in \mathcal{V}$. However, it can be also defined by a set of traffic matrices (in this case D is a convex hull of those matrices) or by an oracle.

To express routing problems as mathematical programs the following notation is introduced.

$\mathcal{P}(i,j)$ **:** finite set of acyclic paths of G from i to j $(i, j \in \mathcal{V})$.

x_p^{ij} **:** proportion of a traffic demand from i to j $(i, j \in \mathcal{V})$ carried through a path $p \in \mathcal{P}(i,j)$. Note that $0 \leqslant x_p^{ij} \leqslant 1$. For a current traffic matrix $t \in D$, the traffic carried through p is then given by $t_{ij} \cdot x_p^{ij}$. To fulfill the stability property, the variable x_p^{ij} does not directly depend on the current traffic matrix.

x_a^{ij} **:** proportion of traffic from i to j flowing through an arc $a \in \mathcal{A}$.

f_a **:** maximum amount of traffic carried on an arc $a \in \mathcal{A}$. It depends on the polytope D and the routing pattern. The variable f_a can also be considered as the minimum capacity that has to be reserved on an arc a to satisfy all the constraints. A vector f is called a *reservation vector*.

$F(D)$ **:** routing cost.

7.1 Basic problems

In this section basic routing problems are presented. They will be later enhanced by introducing the partitioning of a traffic demand polytope.

7.1.1 Robust routing

The problem of computing the minimum cost robust stable routing of an uncertainty domain D, denoted by Robust Routing (RR), can be formulated as in (7.1).

Problem RR

$$\text{minimize } F^{RR}(D) = \sum_{a \in \mathcal{A}} \xi_a f_a, \tag{7.1a}$$

$$\sum_{p \in \mathcal{P}(i,j)} x_p^{ij} \geqslant 1, \qquad \forall i, j \in \mathcal{V}, \tag{7.1b}$$

$$\sum_{p \in \mathcal{P}(i,j), p \ni a} x_p^{ij} \leqslant x_a^{ij}, \qquad \forall i, j \in \mathcal{V},\ \forall a \in \mathcal{A}, \tag{7.1c}$$

$$\sum_{i,j \in \mathcal{V}} x_a^{ij} t_{ij} \leqslant f_a, \qquad \forall a \in \mathcal{A},\ \forall t \in D, \tag{7.1d}$$

$$f_a \leqslant c_a, \qquad \forall a \in \mathcal{A}, \tag{7.1e}$$

$$x_p^{ij} \geqslant 0, \qquad \forall p \in \mathcal{P}(i,j),\ \forall i, j \in \mathcal{V}. \tag{7.1f}$$

The objective is to minimize the total routing cost. Inequalities (7.1b) express the fact that traffic demands between every pair of nodes may be split among many paths. Every variable x_a^{ij} is defined by (7.1c). For a given traffic matrix t, a traffic on an arc a is given by the left-hand side of (7.1d). Thus, the capacity f_a that should be reserved on an arc a must be higher than the traffic carried in all the situations. In other words, inequalities (7.1d) must be valid for each traffic matrix t in the polytope D. Inequality (7.1e) indicates that f_a is lower than the capacity of an arc a. Non-negativeness of x_p^{ij} is forced by (7.1f). This leads to non-negativeness of x_a^{ij}, due to (7.1c), and consequently non-negativeness of f_a, due to (7.1d). Note that the variables x_a^{ij} can be eliminated from the formulation. However, according to initial experiments eliminating them from the implementation does not improve running times of algorithms presented in the following sections. Therefore, they are kept for the sake of clarity and simplicity.

Presented in this way, Robust Routing seems to be difficult. First, the number of paths $\mathcal{P}(i,j)$ can be very high. Second, inequalities (7.1d) are non-linear. Finally, they have to be satisfied for each traffic matrix in D, and D is generally an infinite set.

Fortunately, the problem was proved to be easily solvable using an algorithm based on constraint generation (see [8, 9]). Considering the current solution of the relaxation of Robust Routing (only a finite set of traffic matrices is taken into account instead of the

full D), we only have to check wether there is an arc a and a traffic matrix $t \in D$ such that (7.1d) is violated for them. This can be easily done by the following linear program:

$$\max_{t \in D} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V} \setminus \{i\}} x_a^{ij} t_{ij}.$$

If the maximum is larger than f_a the violated inequality is added.

Routing paths can also be generated in an iterative way by solving a shortest path problem for each pair (i, j), where $i, j \in \mathcal{V}$, and link weights are given by the values of appropriate dual variables of (7.1c). If there exists a path whose reduced cost is negative it has to be added to an appropriate $\mathcal{P}(i, j)$.

Another way of solving Robust Routing, based on duality, was proposed in [2]. The dual of the maximization problem above should be written, and the optimum must be always lower than f_a. A similar approach was also used in a previous work [12]. Instead of generating constraints, a compact formulation follows from duality, but the number of variables of the problem increases significantly. Moreover, the approach assumes that a traffic demand polytope is given by a set of inequalities, while the approach presented in [8, 9] is more general, and can also deal with polytopes described by their extreme points or by an oracle.

Note that it is possible to use other cost functions for this problem. For instance it is possible to minimize the congestion. In such a situation the cost function has to be replaced by $F^{RR}(D) = z$, where z is a variable denoting the congestion. Moreover, constraint (7.1e) has to be replaced with $f_a \leqslant z \cdot c_a$. Although the presented algorithms were tested using both cost functions (routing cost and congestion, see Chapter 10) the rest of the models will be presented only for the former one (routing cost).

Notice that the problem difficulty does not change, if either traffic demands or network links are not oriented. It is easy to solve Robust Routing using similar techniques even if either demands or links are undirected.

7.1.2 No sharing routing

Another set of problems emerges when harsher rules on reservation of capacities are imposed, and different demands are forbidden from sharing resources. The core problem of this set is denoted as No Sharing (NS), and is formulated as follows.

Problem NS

$$\text{minimize } F^{NS}(D) = \sum_{a\in\mathcal{A}} \xi_a f_a, \tag{7.2a}$$

$$\sum_{p\in\mathcal{P}(i,j)} x_p^{ij} \geqslant 1, \qquad \forall i,j \in \mathcal{V}, \tag{7.2b}$$

$$\sum_{p\in\mathcal{P}(i,j),p\ni a} x_p^{ij} \leqslant x_a^{ij}, \qquad \forall i,j \in \mathcal{V},\ \forall a \in \mathcal{A}, \tag{7.2c}$$

$$x_a^{ij} t_{ij} \leqslant f_a^{ij}, \qquad \forall i,j \in \mathcal{V},\ \forall a \in \mathcal{A},\ \forall t \in D, \tag{7.2d}$$

$$\sum_{i,j\in\mathcal{V}} f_a^{ij} \leqslant f_a, \qquad \forall a \in \mathcal{A}, \tag{7.2e}$$

$$f_a \leqslant c_a, \qquad \forall a \in \mathcal{A}, \tag{7.2f}$$

$$x_p^{ij} \geqslant 0, \qquad \forall p \in \mathcal{P}(i,j),\ \forall i,j \in \mathcal{V}. \tag{7.2g}$$

In the formulation variables f_a^{ij} denote an amount of capacity reserved on an arc a by traffic generated between nodes i and j. In fact, they [like x_a^{ij} — here and in (7.1)] are obsolete and can be eliminated from the formulation. However, they are kept for the sake of clarity and simplicity. Formulation (7.2) is obtained from (7.1) by substituting (7.2d) and (7.2e) for (7.1d).

Constraint (7.2d) results in $f_a^{ij} = \max_{t\in D} x_a^{ij} t_{ij}$. However, $\max_{t\in D} x_a^{ij} t_{ij} = x_a^{ij} \max_{t\in D} t_{ij}$. Denoting $\max_{t\in D} t_{ij}$ as t_{ij}^{max}, we can write that $f_a^{ij} = x_a^{ij} t_{ij}^{max}$.

Therefore, the basic No Sharing is equivalent to reserving t_{ij}^{max} of flow for each relation (i,j), $i,j \in \mathcal{V}$. Therefore, it is easily solvable by means of an appropriate linear program facilitated by a path generation mechanism. However, it gets more complicated when D is allowed to be partitioned (see Section 7.3).

7.1.3 Dynamic routing

The problem consists in providing routing schemes (possibly different) for all traffic matrices from a traffic demand polytope. In order to formulate the following variables are introduced.

$x_p^{ij,t}$ **:** proportion of a traffic demand from i to j $(i,j \in \mathcal{V})$ for a traffic matrix t carried through a path $p \in \mathcal{P}(i,j)$.

$x_a^{ij,t}$ **:** proportion of traffic from i to j for a traffic matrix t on an arc $a \in \mathcal{A}$.

The core problem of this set is denoted as Dynamic Routing (DR), and can be formulated as follows.

Problem DR

$$\text{minimize } F^{DR}(D) = \sum_{a \in \mathcal{A}} \xi_a f_a, \tag{7.3a}$$

$$\sum_{p \in \mathcal{P}(i,j)} x_p^{ij,t} \geqslant 1, \qquad \forall i,j \in \mathcal{V},\ \forall t \in D, \tag{7.3b}$$

$$\sum_{p \in \mathcal{P}(i,j), p \ni a} x_p^{ij,t} \leqslant x_a^{ij,t}, \qquad \forall i,j \in \mathcal{V},\ \forall a \in \mathcal{A},\ \forall t \in D, \tag{7.3c}$$

$$\sum_{i,j \in \mathcal{V}} x_a^{ij,t} t_{ij} \leqslant f_a, \qquad \forall a \in \mathcal{A},\ \forall t \in D, \tag{7.3d}$$

$$f_a \leqslant c_a, \qquad \forall a \in \mathcal{A}, \tag{7.3e}$$

$$x_p^{ij,t} \geqslant 0, \qquad \forall p \in \mathcal{P}(i,j),\ \forall i,j \in \mathcal{V},\ \forall t \in D. \tag{7.3f}$$

Formulation (7.3) is similar to (7.1). However, it replaces each constraint of (7.1) involving variables x_p^{ij} or x_a^{ij} with an infinite (in general) number of constraints involving variables $x_p^{ij,t}$ and $x_a^{ij,t}$.

Although all the traffic matrices $t \in D$ are present in the formulation it is enough to consider only extreme points of D in order to solve the problem. Having the optimal routings for all the extreme points it is possible to compute optimal routings for all $t \in D$ by calculating appropriate convex combinations of the optimal routings for the extreme points.

The obvious fact is that the problem is solvable in polynomial time if D is defined as a convex hull of a given finite set of traffic matrices. But this is not the only special case when the problem is simple.

Proposition 7.1.1. DYNAMIC ROUTING *can be solved in polynomial time when D is defined by its facets, and the number of demands is less than a known constant K.*

Proof. In such a case the number of extreme points of D is limited by $O(n^K)$, where n is the number of facets, because each extreme point can be explicitly defined by a set of K facets. Note that K is here also the dimension of the traffic demand polytope D. The number of possible different sets of K facets is $\binom{n}{K}$, which is obviously smaller than n^K. Therefore, it is possible to express the problem as a linear program with a polynomially limited number of variables and constraints. □

The problem is co-$\mathcal{NP}$-hard when the number of demands is unlimited [19]. The complexity of the case when the number of demands is limited by K, and the traffic demand polytope is defined by an oracle is still an open question.

7.2 Test cases

In this section two different ways of describing a traffic demand polytope in practice are presented. Both of them were used to evaluate the novel routing strategies described in the thesis.

7.2.1 Hose model

In [27] a hose model was presented. It assumes that the outgoing traffic of each node is limited, i.e., $\sum_{j\in\mathcal{V}} t_{ij} \leqslant A_i$, for each $i \in \mathcal{V}$, where A_i is an upper bound for outgoing traffic from a node i. Moreover, limitations on the incoming traffic can also be considered. In such a case, $\sum_{i\in\mathcal{V}} t_{ij} \leqslant B_j$, for each $j \in \mathcal{V}$, where B_j is an upper bound for incoming traffic to a node j. The traffic matrix then can be any matrix satisfying this kind of constraints. A polytope satisfying those constraints is called a *hose model polytope*, and is formally described as follows.

Hose model polytope

$$\sum_{j\in\mathcal{V}} t_{ij} \leqslant A_i, \qquad \forall i \in \mathcal{V}, \tag{7.4a}$$

$$\sum_{i\in\mathcal{V}} t_{ij} \leqslant B_j, \qquad \forall j \in \mathcal{V}, \tag{7.4b}$$

$$t_{ij} \geqslant 0, \qquad \forall i, j \in \mathcal{V}. \tag{7.4c}$$

In addition it is possible to assume that both the maximum and minimum traffic between each pair of nodes are also constrained, i.e., $t_{ij}^{min} \leqslant t_{ij} \leqslant t_{ij}^{max}$. the obtained polytope is called a *general hose model polytope*, and is formally described as follows.

General hose model polytope

$$\sum_{j\in\mathcal{V}} t_{ij} \leqslant A_i, \qquad \forall i \in \mathcal{V},\ \forall t \in D, \tag{7.5a}$$

$$\sum_{i\in\mathcal{V}} t_{ij} \leqslant B_j, \qquad \forall j \in \mathcal{V},\ \forall t \in D, \tag{7.5b}$$

$$t_{ij} \geqslant t_{ij}^{min}, \qquad \forall i, j \in \mathcal{V},\ \forall t \in D, \tag{7.5c}$$

$$t_{ij} \leqslant t_{ij}^{max}, \qquad \forall i, j \in \mathcal{V},\ \forall t \in D. \tag{7.5d}$$

Note that not all nodes in a network have to generate traffic. Assume that some nodes are transit nodes, and are neither source nor sink nodes of any demand. They are called then *non-active* nodes, while the rest are called *active* nodes.

7.2.2 B-S model

In [13] Bertsimas and Sim presented a traffic demand model in which each demand can assume only either its minimum value t_{ij}^{min} or its maximum value t_{ij}^{max}, and the number of demands that assume their maximum values cannot be greater than k. The model can be modified in a way that all possible traffic demand matrices considered in the original model

are treated as extreme points of D in a new model. A polytope satisfying those constraints is called *B-S model polytope* in this thesis, and is formally described as follows.

B-S model polytope

$$\sum_{i,j\in\mathcal{V}} \frac{t_{ij} - t_{ij}^{min}}{t_{ij}^{max} - t_{ij}^{min}} \leqslant k, \qquad \forall t \in D, \tag{7.6a}$$

$$t_{ij} \geqslant t_{ij}^{min}, \qquad \forall i,j \in \mathcal{V},\ \forall t \in D, \tag{7.6b}$$

$$t_{ij} \leqslant t_{ij}^{max}, \qquad \forall i,j \in \mathcal{V},\ \forall t \in D. \tag{7.6c}$$

Notice that, like in a hose model case, the number of extreme points of this traffic demand polytope is exponential. Moreover, also in this case we can talk about active and non-active nodes. For the sake of simplicity assume that if a node is active, then it generates traffic to all other active nodes.

7.3 Partitioning strategies

Assume that a traffic demand polytope D and a normal vector α that defines a direction of a hyperplane $\alpha \cdot t = \beta$ (β is subject to optimization) are given. Note that $\beta_{min} \leqslant \beta \leqslant \beta_{max}$, where $\beta_{min} = \min_{t\in D} \alpha \cdot t$ and $\beta_{max} = \max_{t\in D} \alpha \cdot t$. Having D and α it is possible to partition D into two subsets $L(D,\beta) = D \cap \{t : \alpha \cdot t \leqslant \beta\}$ and $R(D,\beta) = D \cap \{t : \alpha \cdot t \geqslant \beta\}$, called the *left hand side polytope* (LHSP) and the *right hand side polytope* (RHSP), respectively. Then, we can consider a robust routing for each of them. Said another way, instead of having only one routing scheme, we will have two schemes: if the current traffic matrix belongs to the first (second) subset it uses the first (second) routing scheme.

Depending on restrictions imposed on the reservation vectors two different strategies can be considered. The reservation vectors on opposite sides of the hyperplane can either be required to be identical, or allowed to differ. The former case models a situation when a user is an owner of a network, and his or her task is to minimize the usage of resources. The latter corresponds to a situation when a user rents capacities from other operators, and he or she cannot renegotiate those renting agreements after each change in the routing.

7.3.1 The reservation vectors can be different

The problem is denoted by Different Reservation, and for Robust Routing (DR/RR) is formulated as follows.

Problem DR/RR

$$F^{*DR/RR}(D) = \min_{\beta} F^{DR/RR}(D,\beta),$$

where the cost function is defined as:

$$F^{DR/RR}(D,\beta) = \max\{F^{RR}(L(D,\beta)), F^{RR}(R(D,\beta))\}.$$

The task is to find such a value of β which ensures that the maximum of costs of Robust Routing for LHSP and for RHSP is minimal. Unfortunately, as it is shown in the example below, the function $F^{DR/RR}(D,\beta)$ as a function of β is not continuous.

Example Consider a network depicted in Figure 7.1 with $c_{(s,b)} = 9$, $c_{(s,e)} = 4$ and $\xi_{(s,c)} = 1$ (an arc from node i to node j is denoted by (i,j)). Other capacities are infinite, and other routing costs are set to zero. The traffic demand polytope D is a convex hull of five traffic matrices: $(3,0,3)^t$, $(0,3,3)^t$, $(0,9,1)^t$, $(3,6,1)^t$ and $(3,9,0)^t$, where $(x,y,z)^t$ denotes a traffic matrix characterized as follows: $t_{sa} = x$, $t_{sb} = y$ and $t_{sc} = z$. The hyperplane is characterized as $t_{sa} + t_{sb} = \beta$.

In the considered network the value of the optimal solution is 0, and is obtained for $\beta = 9$. If such a hyperplane is used then for LHSP both demands sa and sb can be routed through (s,b), and the demand sc can utilize a path $\{(s,e),(e,c)\}$. For RHSP the route for the demand sa changes to a path $\{(s,e),(e,a)\}$, but still the demand sc can utilize the same path $\{(s,e),(e,c)\}$ as for LHSP, because $t_{sa} + t_{sc}$ for RHSP is smaller or equal to $c_{(s,e)}$. The cost of the solution is 0, because the expensive arc (s,c) is used neither for LHSP nor for RHSP.

However, when $\beta = 9 + \gamma$ ($\gamma > 0$) the optimal solution is 2 regardless of the actual value of γ. Even a small γ prevents the demand sa from using the path $\{(s,b),(b,a)\}$ for LHSP, and forces it to use an arc (s,e) (note that a matrix $(\gamma, 9, 1-\gamma/3)^t$ belongs to LHSP, so the whole demand sa has to use an arc (s,e)). As a sum $t_{sa} + t_{sc}$ can be as big as 6 in LHSP, the demand sc cannot only utilize the inexpensive path $\{(s,e),(e,c)\}$, and has to also utilize the direct expensive path (s,c). □

Another issue is that obtained results (routings and the hyperplane) have to be implemented in actual network equipment, and these devices can only work with finite precision numbers. Taking this into account, and recalling that $F^{DR/RR}(D,\beta)$ is non-continuous it was decided that β can only assume values of a finite precision, i.e., only values of form $n \cdot \gamma$ are considered, where $n \in \mathbb{Z}$ and γ is the precision.

When the finite precision of β is assumed, Different Reservation for Robust Routing can be optimally solved using a polynomial algorithm presented in [6]. Knowing that $F^{RR}(L(D,\beta))$ is a non-decreasing function of β, and $F^{RR}(R(D,\beta))$ is a non-increasing function of β we can find, using a binary search, $\beta^\star$ such that $F^{DR/RR}(D,\beta^\star)$ is minimized. Notice that the binary search should depend in a logarithmic way on γ. Obviously $\beta^\star$ is a solution to Different Reservation.

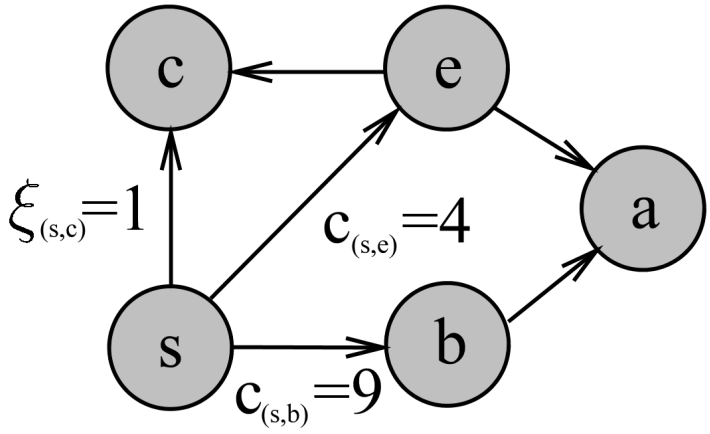


Figure 7.1: Network exposing non-continuity of the cost function.

In order to formulate Different Reservation for No Sharing, F^{NS} have to be substituted for F^{RR}. Note that this variant can be solved using the same algorithm, because $F^{NS}(L(D,\beta))$ [like $F^{RR}(L(D,\beta))$] is also a non-decreasing function of β, and $F^{NS}(R(D,\beta))$ is a non-increasing function of β.

The problem can be also formulated, and solved, in a similar way for Dynamic Routing. It is co-$\mathcal{NP}$-hard in general but remains in $\mathcal{P}$ for some special cases, e.g., when D is defined as a convex hull of a given set of traffic matrices. Dynamic routing partitioning is discussed in details in Section 8.3.

Although $F^{DR/RR}(D,\beta)$ is non-continuous in general, there exist special cases when it is continuous, e.g., when capacities are infinite, and a constraints (7.1e) is eliminated from the appropriate formulation. Before proving that, a new notation and some lemmas have to be introduced.

The Hausdorff distance between two polytopes D_1 and D_2 is defined as follows.

$$d(D_1,D_2)=\max(\sup_{t^1\in D_1}\inf_{t^2\in D_2}||t^1-t^2||,\sup_{t^1\in D_2}\inf_{t^2\in D_1}||t^1-t^2||)$$

where the norm $||t^1-t^2||=\sum_{i,j\in\mathcal{V}}|t^1_{ij}-t^2_{ij}|$. Note that, since D_1 and D_2 are compact, the distance can be also presented in the following way:

$$d(D_1,D_2)=\max(\max_{t^1\in D_1}\min_{t^2\in D_2}||t^1-t^2||,\max_{t^1\in D_2}\min_{t^2\in D_1}||t^1-t^2||).$$

In addition define the polytope $D_\beta=D\cap\{t\geqslant 0:\alpha\cdot t=\beta\}$. Remember also that $\beta_{min}<\beta<\beta^{max}$.

Lemma 7.3.1. *For each bounded polytope D the condition $\lim_{\beta^1\to\beta^2} d(L(D,\beta^1),L(D,\beta^2))=0$ holds.*

Proof. Assume that $\beta^1 < \beta^2$ (an analogous proof can be given for $\beta^1 > \beta^2$), so $L(D,\beta^1) \subseteq L(D,\beta^2)$ and:

$$d(L(D,\beta^1), L(D,\beta^2)) = \max_{t^2 \in L(D,\beta^2)} \min_{t^1 \in L(D,\beta^1)} ||t^1 - t^2||.$$

For given D, β^1, and $t^2 \in L(D,\beta^2)$ define $t(t^2,\beta^1) \in L(D,\beta^1)$, such that $||t(t^2,\beta^1) - t^2||$ is minimized. Note that D is compact, and thus, according to Weierstrass' theorem, $t(t^2,\beta^1)$ exists.

Assume now that the lemma is false. It means that there exists $t^2 \in L(D,\beta^2)$ and $\varepsilon > 0$ such that for each $\beta^1 \in]\beta_{min}, \beta^2]$ the norm $||t(t^2,\beta^1) - t^2|| \geqslant \varepsilon$. Moreover, as D is compact and the norm is continuous, there exists $\beta^\star \in]\beta_{min}, \beta^2]$, such that $||t(t^2,\beta^\star) - t^2|| = \varepsilon$. Take a segment joining $t(t^2,\beta^\star)$ and t^2. The segment belongs to D, because D is convex. Now take t^1 in the middle of the segment, and set β^1 to $\frac{\beta^\star+\beta^2}{2}$. Obviously $||t^1 - t^2|| = \frac{\varepsilon}{2}$. Moreover, t^1 belongs to $L(D,\beta^1)$, thus $||t(t^2,\beta^1) - t^2|| \leqslant \frac{\varepsilon}{2}$. Therefore, the contradiction is reached, as ε was not the value of the minimum norm. □

Lemma 7.3.2. *When capacities are infinite $F^{RR}(L(D,\beta))$ is continuous in variable β.*

Proof. From the formal point of view the proposition states that if capacities are infinite then:

$$\lim_{\beta' \to \beta} F^{RR}(L(D,\beta')) = F^{RR}(L(D,\beta)). \tag{7.7a}$$

Assume that a solution to Robust Routing for $L(D,\beta)$ is given, and values of f_a correspond to this solution. If we assume that $\beta' \geqslant \beta$ then the considered traffic demand polytope, i.e., $L(D,\beta)$, is extended to $L(D,\beta')$. In such a situation some of f_a may have to be increased in order to route new traffic matrices. As capacities are unlimited, it is possible to route those new matrices using the routing corresponding to $L(D,\beta)$. It is obvious that any routing (consisting of loopless paths) cannot cost more than $\sum_{a\in\mathcal{A}} \xi_a$ for each unit of routed flow. Therefore, it is possible to write that:

$$F^{RR}(L(D,\beta)) \leqslant F^{RR}(L(D,\beta')) \leqslant F^{RR}(L(D,\beta)) + \sum_{a\in\mathcal{A}} \xi_a \cdot d(L(D,\beta'), L(D,\beta)).$$

Knowing from Lemma 7.3.1 that $\lim_{\beta'\to\beta} d(L(D,\beta'), L(D,\beta)) = 0$ it is justified to conclude that (7.7) is true. □

Obviously the same fact can be proved for $F^{RR}(R(D,\beta))$. Now it is possible to prove the following proposition.

Proposition 7.3.3. *When capacities are infinite $F^{DR/RR}(D,\beta)$ is continuous in variable β.*

Proof. $F^{DR/RR}(D,\beta)$ for β is just a maximum of two continuous functions, so it also has to be continuous. □

7.3.2 The reservation vectors are identical

Assume that the reservation vectors corresponding to the two uncertainty subsets (LHSP and RHSP) should be the same. The problem will be called Identical Reservation (for Robust Routing or No Sharing). Note that Identical Reservation for Dynamic Routing is meaningless, because partitioning in such conditions does not have any effect.

A version that corresponds to Robust Routing (IR/RR) can be formulated as follows.

Problem IR/RR

$$F^{*IR/RR}(D) = \min_{\beta} F^{IR/RR}(D, \beta),$$

where the cost function is defined as follows.

$$\text{minimize } F^{IR/RR}(D,\beta) = \sum_{a\in\mathcal{A}} \xi_a f_a, \tag{7.8a}$$

$$\sum_{p\in\mathcal{P}(i,j)} x_p^{ij} \geqslant 1, \qquad \forall i,j \in \mathcal{V}, \tag{7.8b}$$

$$\sum_{p\in\mathcal{P}(i,j),p\ni a} x_p^{ij} \leqslant x_a^{ij}, \qquad \forall i,j \in \mathcal{V},\ \forall a \in \mathcal{A}, \tag{7.8c}$$

$$\sum_{i,j\in\mathcal{V}} x_a^{ij} t_{ij} \leqslant f_a, \qquad \forall a \in \mathcal{A},\ \forall t \in L(D,\beta), \tag{7.8d}$$

$$\sum_{p\in\mathcal{P}(i,j)} \overline{x_p^{ij}} \geqslant 1, \qquad \forall i,j \in \mathcal{V}, \tag{7.8e}$$

$$\sum_{p\in\mathcal{P}(i,j),p\ni a} \overline{x_p^{ij}} \leqslant \overline{x_a^{ij}}, \qquad \forall i,j \in \mathcal{V},\ \forall a \in \mathcal{A}, \tag{7.8f}$$

$$\sum_{i,j\in\mathcal{V}} \overline{x_a^{ij}} t_{ij} \leqslant f_a, \qquad \forall a \in \mathcal{A},\ \forall t \in R(D,\beta), \tag{7.8g}$$

$$f_a \leqslant c_a, \qquad \forall a \in \mathcal{A}, \tag{7.8h}$$

$$x_p^{ij}, \overline{x_p^{ij}} \geqslant 0, \qquad \forall p \in \mathcal{P}(i,j),\ \forall i,j \in \mathcal{V}. \tag{7.8i}$$

Formulation (7.8) is an extended version of (7.1). The number of variables and constraints is almost doubled in order to describe two independent routings on both sides of the hyperplane (non-overlined variables correspond to LHSP, and overlined variables correspond to RHSP). As the reservation vectors on both sides of the hyperplane have to be identical it is possible to describe them using only one set of variables, namely f_a, where $a \in \mathcal{A}$. Inequalities (7.8d) and (7.8g) assure that both routings can be realized using the reserved capacities.

Notice that the considered problem, with β fixed, can be solved in polynomial time using the way described in [9], because both constraints and paths can be easily generated in polynomial time.

While dealing with the Identical Reservation version for No Sharing (IR/NS) constraints (7.8d) and (7.8g) have to be replaced with:

$$x_a^{ij} t_{ij} \leqslant f_a^{ij} \qquad \forall i,j \in \mathcal{V},\ \forall a \in \mathcal{A},\ \forall t \in L(D,\beta), \tag{7.9a}$$

$$\overline{x_a^{ij}} t_{ij} \leqslant \overline{f_a^{ij}} \qquad \forall i,j \in \mathcal{V},\ \forall a \in \mathcal{A},\ \forall t \in R(D,\beta), \tag{7.9b}$$

and the following two sets of constraints have to be added.

$$\sum_{i,j\in\mathcal{V}} f_a^{ij} \leqslant f_a \qquad \forall a \in \mathcal{A}, \tag{7.10a}$$

$$\sum_{i,j\in\mathcal{V}} \overline{f_a^{ij}} \leqslant f_a \qquad \forall a \in \mathcal{A}. \tag{7.10b}$$

Moreover, the cost function has to be replaced with $F^{IR/NS}(D,\beta) = \sum_{a\in\mathcal{A}} \xi_a f_a$. Notice that when considering Identical Reservation for No Sharing, if β is fixed, the resulting optimization problem is identical to Dynamic Routing with demand set given as the convex hull of two demand matrices.

$F^{IR/RR}(D,\beta)$, like $F^{DR/RR}(D,\beta)$, is non-continuous in β (in order to prove this it is possible to use the same example as in Section 7.3.1). Therefore, also in this case it was decided that β can only assume values of finite precision, i.e., only values of form $n \cdot \gamma$ are considered, where $n \in \mathbb{Z}$ and γ is the precision.

Chapter 8

Algorithms

In this chapter algorithms solving the previously presented partitioning problems are discussed. In Section 8.1 an algorithm solving Identical Reservation is presented. Another (faster) algorithm is described in Section 8.2. However, it can be applied only to Identical Reservation for No Sharing when a polytope is defined as a convex hull of a given set of traffic matrices. The whole Section 8.3 is devoted to Dynamic Routing. Finally, Section 8.4 consists of implementation details that accelerate the presented algorithms.

8.1 Double binary search algorithm

In this section an algorithm that solves Identical Reservation is presented. It can successfully deal with both viable versions of the problem (Robust Routing and No Sharing) but for the sake of simplicity it concentrates only on the former one. As mentioned earlier Identical Reservation for Dynamic Routing is meaningless.

Computational complexity

The computational complexity of this problem still remains an open question. However, some results dealing with the complexity of its approximation schemes have been already published.

In [6] a polynomial time 2-approximation algorithm for Identical Reservation for Robust Routing was presented. The algorithm works as follows. Let $\beta^\star$ be the optimal solution to Different Reservation, $f^\star$ the optimal reservation vector used for $L(D, \beta^\star)$ and $\overline{f^\star}$ the optimal reservation vector used for $R(D, \beta^\star)$. One can obviously build a new reservation vector defined by the componentwise maximum of $f^\star$ and $\overline{f^\star}$. This vector will allow to carry traffic for each $t \in D$ (keep the routing schemes obtained when Different Reservation is solved). It is clear that the cost of this new vector is less than twice the cost of the optimal solution to Identical Reservation.

The algorithm is polynomial as it requires to solve DIFFERENT RESERVATION (this problem is polynomial, see Section 7.3.1) and perform some simple (of polynomial complexity) operations while building the reservation vector.

Algorithm

Consider a simpler problem that consists in answering the question: does there exist a feasible solution to IDENTICAL RESERVATION of cost smaller than or equal to a given value? This simplified problem will be refereed to as IDENTICAL RESERVATION LIMIT. Obviously, knowing the way of solving IDENTICAL RESERVATION LIMIT it is possible to solve IDENTICAL RESERVATION relatively fast using a binary search.

First upper and lower bounds for the cost $F^{IR/RR}(D, \beta)$, denoted by F_{min} and F_{max} respectively, have to be known. The simplest lower bound F_{min} is 0, while the simplest upper bound is equal to the value of the solution to ROBUST ROUTING or NO SHARING depending on the actual version of the considered problem. Obviously those bounds can be upgraded (see Section 8.4.1).

The question arises how to solve IDENTICAL RESERVATION LIMIT. In this section an algorithm that can do this for both ROBUST ROUTING and NO SHARING is presented. Although its complexity is not polynomial, it performs well, and can solve even medium size problems (see the numerical results in Chapter 10). The key ideas of the approach can be seen in Algorithm 8.1.

Algorithm 8.1 IDENTICAL RESERVATION LIMIT(F_{lim})

$\beta_2 = \beta_{max}$
while $\beta_2 > \beta_{min}$ **do**
  $\beta_1 = \texttt{findMax}(\beta_2, F_{lim})$
  if $\beta_1 = \beta_2$ **then**
    return YES
  else
    $\beta_2 = \beta_1$
  end if
end while
return NO

The algorithm returns YES, if a solution to IDENTICAL RESERVATION that costs no more than F_{lim} exists. Otherwise, it returns NO. It uses constants β_{min} and β_{max} that define an interval of possible values of β, and the function `findMax`. The basic idea of the method is to constantly shrink the interval of possible positions of the hyperplane. At the beginning the hyperplane can be anywhere in $[\beta_{min}, \beta_{max}]$. It means that it is impossible to judge if any $t \in D$ has to belong either to LHSP or to RHSP (assume that the interval is sufficiently large, and $D \subseteq \{t : \alpha \cdot t \geqslant \beta_{min}\} \cap \{t : \alpha \cdot t \leqslant \beta_{max}\}$). Then, in the loop, the interval of possible positions of the hyperplane is limited. More precisely, the set of matrices that have

to belong to RHSP is extended.

Having RHSP defined for a particular loop it is possible, using `findMax` method, to calculate the maximum size of LHSP which satisfies the cost limit. If the whole polytope D can be divided between LHSP and RHSP the problem has been solved. If some matrices cannot be added to LHSP for a given RHSP, it means that they have to belong to RHSP. They are added to RHSP by changing the value of β_2, and the loop is repeated.

Consider now a new problem that will facilitate explanation of `findMax` method. Assume that values β_1 and β_2 are given. The problem, denoted by Two Known Planes, consists in finding the optimal routing when LHSP is defined as $L(D, \beta_1)$ and RHSP is defined as $R(D, \beta_2)$. Note that β_1 and β_2 are constant so the problem can be easily solved using techniques presented in [8, 9], and briefly described in Chapter 7.

The method `findMax` returns the maximum value of β_1 that lets Two Known Planes be solved for a given β_2 and within a given cost limit F_{lim}. If such a value does not exist it returns $-\infty$. Note that β_1 is a variable subject to optimization and β_2 is constant. In such a situation the objective function of Two Known Planes is non-decreasing in β_1, as it is impossible to decrease a cost of a solution by adding new traffic matrices to the problem. Therefore, the method can use (and it does) a binary search. A pseudo-code for `findMax` can be seen in Algorithm 8.2.

Algorithm 8.2 `findMax`(β_2, F_{lim})

$\beta_1 = \beta_{min}$
$\beta_1' = \beta_2$
while $\beta_1' - \beta_1 > \gamma$ **do**
  $\beta_{mid} = \frac{\beta_1 + \beta_1'}{2}$
  if Two Known Planes$(\beta_{mid}, \beta_2) \leqslant F_{lim}$ **then**
    $\beta_1 = \beta_{mid}$
  else
    $\beta_1' = \beta_{mid}$
  end if
end while
return β_1'

Unfortunately, the overall complexity of the whole algorithm is not polynomial. The reason is that the value β_1 found by `findMax` method can be very close to β_2. In fact a difference between β_2 and β_1 can be as small as γ. It means that in the worst case the main loop of the algorithm has to be repeated $m = \lceil \frac{\beta_{max} - \beta_{min}}{\gamma} \rceil$ times. Apparently, the value of m cannot be expressed as a polynomial function of the size of data. In Figure 8.1 an example network that suffers from this drawback is depicted.

Example Assume that a polytope D is a convex hull of only two traffic matrices. Both

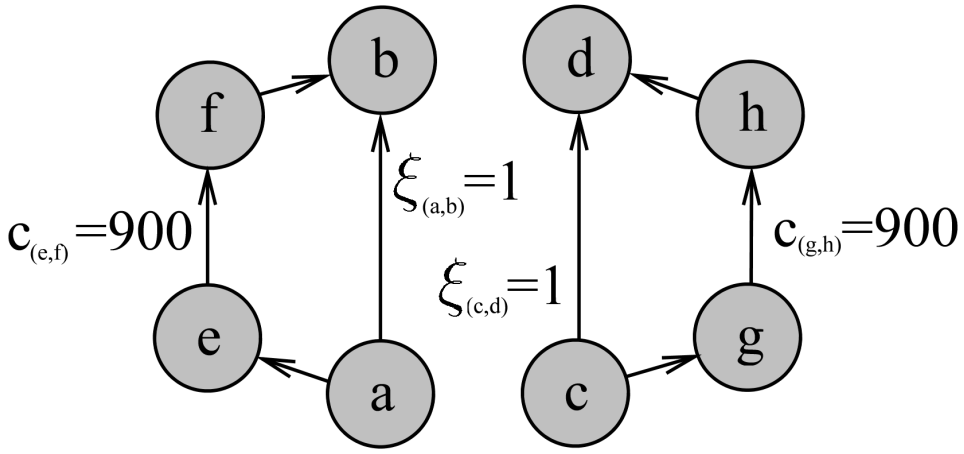


Figure 8.1: Network exposing non-polynomiality of the algorithm.

matrices consist of four demands. The first is as follows: $t_{ab} = 0.1$, $t_{cd} = 0.9$, $t_{ef} = 900$, $t_{gh} = 100$, while the second is: $t_{ab} = 0.9$, $t_{cd} = 0.1$, $t_{ef} = 100$, $t_{gh} = 900$.

Capacities of all arcs but two are unlimited. Only arcs (e, f) and (g, h) have their capacities ($c_{(e,f)}$ and $c_{(g,h)}$, respectively) set to 900. For the majority of arcs, routing costs are set to 0, and only $\xi_{(a,b)} = \xi_{(c,d)} = 1$. Finally, it is assumed that the hyperplane is defined as $t_{ab} = \beta$.

The optimal solution to this instance of Robust Routing is 1.8, because demands ab and cd have to use expensive (direct) paths. They cannot utilize cheap ($\{(a, e), (e, f), (f, b)\}$ and $\{(c, g), (g, h), (h, d)\}$) routes, because arcs (e, f) and (g, h) are fully occupied by demands ef and gh. On the other hand, the optimal solution to Identical Reservation is only 1.0, as D can be divided using the hyperplane $t_{ab} = 0.5$, and different routings on both sides of it can be used. In such a situation the demand ab has to occupy the arc (a, b) only when $t_{ab} \leqslant 0.5$, and the demand cd has to occupy the arc (c, d) only when $t_{cd} \leqslant 0.5$. Note that, this instance of the problem has the infinite number of different optimal solutions that can be obtained for all $\beta \in [\frac{100}{999}, \frac{899}{999}]$.

Assume that the presented algorithm is used, existence of a solution of cost 1.0 has been verified, and now a feasible solution of cost $1.0 - \epsilon$ is of interest. Unluckily, in the considered network a solution to Two Known Planes, when $\frac{100}{999} \leqslant \beta_1 \leqslant \beta_2 \leqslant \frac{899}{999}$, is $1.0-(\beta_2-\beta_1)$. It means that in order to use the algorithm to verify if a feasible solution of cost $1.0 - \epsilon$ exists, Two Known Planes has to be solved almost ϵ^{-1} times, because for $\beta_2 \in [\frac{100}{999} + \epsilon, \frac{899}{999}]$ the value of β_1, resulting from Two Known Planes, will be equal to $\beta_2 - \epsilon$. Note that in the worst case $\epsilon = \gamma$, where γ is the precision for β defined in Section 7.3. □

8.2 Limited number of extreme points (No sharing)

In Section 8.1 an algorithm that solves Identical Reservation for both Robust Routing and No Sharing (Identical Reservation for Dynamic Routing is meaningless) was presented. In this section a special feature of No Sharing is used, and an algorithm that can solve this version of Identical Reservation in polynomial time when D is a convex hull of a given set of traffic matrices is presented. Note that the algorithm *cannot* be used for Robust Routing version of Identical Reservation.

Assume that a polytope D and hyperplanes $\alpha \cdot t = \beta_1$ and $\alpha \cdot t = \beta_2$ ($\beta_1 < \beta_2$) are given, and a *middle polytope* $D_{mid} = R(D, \beta_1) \cap L(D, \beta_2)$ is not empty but does not contain any extreme points of D. Define *left* and right polytope*right polytopes* as $D_{left} = L(D, \beta_1)$ and $D_{right} = R(D, \beta_2)$, respectively. Note that $D_{left} \cup D_{mid} \cup D_{right} = D$.

Lemma 8.2.1. *For any $\beta \in]\beta_1, \beta_2[$ all points $t \in D_\beta$ can be expressed as $t = \lambda t' + (1-\lambda)t''$, where $\lambda = \frac{\beta_2 - \beta}{\beta_2 - \beta_1}$, $t' \in D_{\beta_1}$ and $t'' \in D_{\beta_2}$.*

Proof. Take any point $t \in D_\beta$. As it belongs to D, it can be expressed as a convex combination of extreme points of D. D_{mid} does not contain any extreme points of D, so all of them have to belong either to D_{left} or to D_{right}. It means that a convex combination of only those extreme points that belong to D_{left} can be calculated, and a point in D_{left} (together with its weight) can be obtained. The same operation can be performed on D_{right}. The considered matrix t will be a convex combination of those two points. Obviously, those two points can be connected by a line that is contained in D. What is more, this line also contains one point from D_{β_1} (denoted by t_{β_1}) and one point from D_{β_2} (denoted by t_{β_2}). Certainly, $t = \lambda t_{\beta_1} + (1-\lambda)t_{\beta_2}$, and we can take $t' = t_{\beta_1}$ and $t'' = t_{\beta_2}$. □

It is possible to think about β_2 as a variable, and write the following lemma.

Lemma 8.2.2. *For any $\beta \in]\beta_1, \beta_2[$ all points $t \in D_{mid} \cap \{t : \alpha \cdot t \leqslant \beta\}$ can be expressed as $t = \lambda t' + (1-\lambda)t''$, where $\lambda = \frac{\beta_2 - \beta}{\beta_2 - \beta_1}$, $t' \in D_{\beta_1}$ and $t'' \in D_{mid} \cap \{t : \alpha \cdot t \leqslant \beta_2\}$.*

Proof. Take any point $t \in D_{mid} \cap \{t : \alpha \cdot t \leqslant \beta\}$. Note that $t \in D_{\beta'}$, where $\beta_1 \leqslant \beta' \leqslant \beta$. Now take $\beta_2' = \beta_1 + \frac{\beta' - \beta_1}{\beta - \beta_1}(\beta_2 - \beta_1)$. Obviously, $\beta_1 \leqslant \beta_2' \leqslant \beta_2$, so there are no extreme points of D between D_{β_1} and $D_{\beta_2'}$. What is more, for those β' and β_2' the fraction $\frac{\beta_2' - \beta'}{\beta_2' - \beta_1}$ is equal to λ defined in the lemma. Therefore, it is possible to use Lemma 8.2.1, and construct each matrix $t \in D_{mid} \cap \{t : \alpha \cdot t \leqslant \beta\}$ from matrices $t' \in D_{\beta_1}$ and $t'' \in D_{mid} \cap \{t : \alpha \cdot t \leqslant \beta_2\}$, using the formula $t = \lambda t' + (1-\lambda)t''$. □

Note that $D_{left} = L(D, \beta_1)$ and $D_{left} \subset L(D, \beta_2)$. Therefore, the previous lemma can be extended to also cover points that belong to D_{left}.

Corollary 8.2.3. *For any $\beta \in]\beta_1, \beta_2[$ all points $t \in L(D, \beta)$ can be expressed as $t = \lambda t' + (1-\lambda)t''$, where $\lambda = \frac{\beta_2 - \beta}{\beta_2 - \beta_1}$, $t' \in L(D, \beta_1)$ and $t'' \in L(D, \beta_2)$.*

Obviously, it works also in the opposite direction.

Corollary 8.2.4. *For any $\beta \in]\beta_1, \beta_2[$ all points $t \in R(D, \beta)$ can be expressed as $t = \lambda t' + (1-\lambda)t''$, where $\lambda = \frac{\beta_2 - \beta}{\beta_2 - \beta_1}$, $t' \in R(D, \beta_1)$ and $t'' \in R(D, \beta_2)$.*

Both Corollaries 8.2.3 and 8.2.4 can be rewritten taking into account t_{ij}^{max}. Notice that $\max_{t \in A} t_{ij}$ is denoted by $t_{ij}^{max}(A)$.

Corollary 8.2.5. *For any $\beta \in]\beta_1, \beta_2[$ maximum values of $t_{ij}^{max} = t_{ij}^{max}(L(D,\beta))$ satisfies an inequality:*

$$t_{ij}^{max} \leqslant \lambda t_{ij}^{max\prime} + (1-\lambda) t_{ij}^{max\prime\prime},$$

where $\lambda = \frac{\beta_2 - \beta}{\beta_2 - \beta_1}$, $t_{ij}^{max\prime} = t_{ij}^{max}(L(D,\beta_1))$ and $t_{ij}^{max\prime\prime} = t_{ij}^{max}(L(D,\beta_2))$. Moreover, maximum values of $\overline{t_{ij}^{max}} = t_{ij}^{max}(R(D,\beta))$ satisfies an inequality:

$$\overline{t_{ij}^{max}} \leqslant \lambda \overline{t_{ij}^{max}}' + (1-\lambda) \overline{t_{ij}^{max}}'',$$

where $\lambda = \frac{\beta_2 - \beta}{\beta_2 - \beta_1}$, $\overline{t_{ij}^{max}}' = t_{ij}^{max}(R(D,\beta_1))$ and $\overline{t_{ij}^{max}}'' = t_{ij}^{max}(R(D,\beta_2))$.

Assume now that the optimal solutions of Identical Reservation (for No Sharing) for given β_1 and β_2 are provided, and a solution for $\beta \in]\beta_1, \beta_2[$ is of interest. Variables of the solution for β will be denoted as described in Chapter 7. Variables of the solution for β_1 will be additionally followed by a prime, e.g., f_a' or $x_p^{ij\prime}$, while those corresponding to the solution for β_2 will be followed by a double prime, e.g., f_a''.

Construct now a routing to the solution for β in the following way.

$$x_p^{ij} = (\lambda x_p^{ij\prime} t_{ij}^{max\prime} + (1-\lambda) x_p^{ij\prime\prime} t_{ij}^{max\prime\prime}) / t_{ij}^{max}, \tag{8.1a}$$

$$\overline{x_p^{ij}} = (\lambda \overline{x_p^{ij}}' \overline{t_{ij}^{max}}' + (1-\lambda) \overline{x_p^{ij}}'' \overline{t_{ij}^{max}}'') / \overline{t_{ij}^{max}}. \tag{8.1b}$$

Lemma 8.2.6. *The routing constructed using (8.1) is feasible, i.e., it satisfies (7.8b) and (7.8e).*

Proof. Focus on the first equality. Use the first inequality from Corollary 8.2.5, substitute u for $\lambda t_{ij}^{max\prime}$, and substitute v for $(1-\lambda) t_{ij}^{max\prime\prime}$. Summing the obtained inequalities for all $p \in \mathcal{P}(i,j)$ the following formula is obtained.

$$\sum_{p \in \mathcal{P}(i,j)} x_p^{ij} \geqslant \frac{u}{u+v} \sum_{p \in \mathcal{P}(i,j)} x_p^{ij\prime} + \frac{v}{u+v} \sum_{p \in \mathcal{P}(i,j)} x_p^{ij\prime\prime}.$$

Variables $x_p^{ij\prime}$ and $x_p^{ij\prime\prime}$ satisfy (7.8b), so x_p^{ij} also has to satisfy it. Similar proof can be used to show that $\overline{x_p^{ij}}$ satisfy (7.8e). □

Knowing that the routing expressed by (8.1) is feasible, and using (7.9) and (7.8c) in order to substitute f_a^{ij} for $\sum_{p \ni a} x_p^{ij} t_{ij}$ the following corollary is obtained.

Corollary 8.2.7. *If a routing of a solution for β satisfies (8.1) then $f_a^{ij} \leqslant \lambda f_a^{ij\prime} + (1-\lambda) f_a^{ij\prime\prime}$ and $\overline{f_a^{ij}} \leqslant \lambda \overline{f_a^{ij}}' + (1-\lambda) \overline{f_a^{ij}}''$, for all $a \in \mathcal{A}$ and $i, j \in \mathcal{V}$.*

Note that, according to (7.8), the cost function $F^{IR}(D,\beta)$ depends only on f_a. What is more, according to (7.10), f_a depends on f_a^{ij} only. Therefore, from Corollary 8.2.7 the following can be deduced.

Corollary 8.2.8. *Function $F^{IR/NS}(D,\beta)$ is convex in $[\beta_1,\beta_2]$, if $D_{mid} = R(D,\beta_1)\cap L(D,\beta_2)$ does not contain extreme points of D.*

Returning to the algorithm, it is simple to calculate a list of increasing values of β that describes all hyperplanes which contain extreme points of D (note that the polytope is a convex hull of a given set of traffic matrices). Define the list as $\mathcal{B} = \{\beta_1, \beta_2, \ldots, \beta_n\}$ such that $\beta_i < \beta_{i+1}$, for $i = 1, 2, \ldots, n-1$, and each extreme point of D belongs to $\bigcup_{i=1}^{n} D_{\beta_i}$. Knowing from Corollary 8.2.8 that function $F^{IR/NS}(D,\beta)$ is convex in $[\beta_i,\beta_{i+1}]$, for $i = 1, 2, \ldots, n-1$, it is possible to calculate optimal $F^{\star IR/NS}_{[\beta_i,\beta_{i+1}]}(D)$, for $i = 1, 2, \ldots, n-1$, using any known algorithm that can find the minimum of a convex function in polynomial time (e.g., golden ratio or Fibonacci search). Obviously, optimal $F^{\star IR/NS}_{[\beta_{min},\beta_{max}]}(D) = \min_{i=1,2,\ldots,n-1} F^{\star IR/NS}_{[\beta_i,\beta_{i+1}]}(D)$. Therefore, the following proposition can be written.

Proposition 8.2.9. Identical Reservation *for* No Sharing *can be solved in polynomial time if the traffic demand polytope is given by a set of its extreme points.*

8.3 Partitioning and dynamic routing

In this section Different Reservation for Dynamic Routing is considered. The problem can be solved by means of the method of Section 7.3.1. It is co-$\mathcal{NP}$-hard in general but belongs to $\mathcal{P}$ when D is defined as a convex hull of a given set of traffic matrices. The former fact can be concluded from the proof presented in [19] showing that Dynamic Routing alone is co-$\mathcal{NP}$-hard. The latter fact will be proved in this section. But first some lemmas have to be discussed.

Consider a traffic demand polytope D divided by a hyperplane $\alpha \cdot t = \beta$ into a right hand side polytope (RHSP), defined as $R(D,\beta)$, and a left hand side polytope (LHSP), defined as $L(D,\beta)$. A strict combination of some points $t \in \mathcal{T}$ is defined as $\sum_{t\in\mathcal{T}} \lambda_t t$, where $\sum_{t\in\mathcal{T}} \lambda_t = 1$ and $\lambda_t > 0$ for all $t \in \mathcal{T}$.

Lemma 8.3.1. *If a traffic matrix t^r can be expressed as a strict convex combination of more than three affinely independent points of D it can be also expressed as a strict convex combination of exactly three affinely independent points of D.*

Proof. Assume that t^r is a strict convex combination of K affinely independent points from D.

$$t^r = \sum_{k=1}^{K} \lambda_k t^k.$$

It can be transformed into:

$$t^r = \lambda_1 t^1 + \lambda_2 t^2 + M \sum_{k=3}^{K} \frac{\lambda_k t^k}{M},$$

where:

$$M = \sum_{k=3}^{K} \lambda_k.$$

This is obviously a strict convex combination of three affinely independent points belonging to D. □

Obviously the same lemma can be proved for a strict convex combination of more than W affinely independent points, where $W \geqslant 3$. However, for the purpose of this section, the proof for $W = 3$ is essential.

Lemma 8.3.2. *If a traffic matrix $t^r \in D$ belonging to the hyperplane $\alpha \cdot t^r = \beta$ has to be expressed as a strict convex combination of three affinely independent points $t^a, t^b, t^c \in D$ then t^r is an extreme point of neither LHSP nor RHSP.*

Proof. First notice that at least one of the points has to belong to LHSP, and at least one has to belong to RHSP (points on the hyperplane belong to both LHSP and RHSP). Assume that t^a is in RHSP, and both t^b and t^c are in LHSP. Note that if all three points belong to LHSP they also have to belong to RHSP (they have to be on the hyperplane). In such a situation t^r cannot be an extreme point of LHSP, because it is a convex combination of points belonging to LHSP. The same claim holds for RHSP.

Define D_1 as a convex hull of t^a, t^b, and t^c. Then $t^r \in D_2 = D_1 \cap \{t : \alpha \cdot t = \beta\}$. D_2 can be also expressed as a convex hull of its two extreme points (first — an intersection of a line connecting t^a to t^b and the hyperplane, second — an intersection of a line connecting t^a to t^c and the hyperplane). t^r cannot be any of these extreme points so it is an extreme point of neither RHSP nor LHSP. □

Now the main claims of this section can be presented.

Corollary 8.3.3. *Each extreme point of either LHSP or RHSP belonging to the hyperplane $\alpha \cdot t = \beta$ has to be either an extreme point of D or a strict convex combination of one extreme point of D belonging to LHSP and one extreme point of D belonging to RHSP.*

Proof. Combining Lemma 8.3.2 and Lemma 8.3.1 it is possible to conclude that any point that has to be a strict convex combination of more than two affinely independent points of D is an extreme point of either LHSP or RHSP. It means that the only way to obtain an extreme point of either RHSP or LHSP is to take either an extreme point of D or a strict convex combination of two extreme points of D. Obviously those two points cannot belong to the same half-space defined by the hyperplane. Therefore, one of them has to belong to RHSP, and the other has to belong to LHSP. □

The corollary implies that the number of extreme points of both LHSP and RHSP is limited by $O(n^2)$, where n is the number of extreme points of D. Therefore, the following corollary can be written.

Corollary 8.3.4. Different Reservation *for* Dynamic Routing, *for given β, can be solved in polynomial time if the traffic demand polytope D is defined as a convex hull of a given set of traffic matrices.*

Proof. The problem can be expressed in form of a linear program with a polynomially limited number of variables and constraints. □

Finally, the following proposition can be written.

Proposition 8.3.5. DIFFERENT RESERVATION *for* DYNAMIC ROUTING *can be solved in polynomial time if the traffic demand polytope D is defined as a convex hull of a given set of traffic matrices.*

Proof. Knowing from Corollary 8.3.4 that DIFFERENT RESERVATION for DYNAMIC ROUTING, for given β, can be solved in polynomial time, and that $F^{DR}(L(D,\beta_1)) \leqslant F^{DR}(L(D,\beta_2))$, when $\beta_1 < \beta_2$, it is possible to conclude that the polynomial time algorithm of Section 7.3.1 based on a binary search can be used in order to solve DIFFERENT RESERVATION for DYNAMIC ROUTING with β as a variable if the traffic demand polytope D is defined as a convex hull of a given set of traffic matrices. □

8.4 Implementation details

In this section some enhancements to the presented algorithms are presented. They are not crucial from the point of view of the theoretical complexity of the algorithms. However, they significantly reduce running times of the implementations.

8.4.1 Cost bounds

In Section 8.1 it was assumed that $F_{min} = 0$ and F_{max} is equal to a cost of either ROBUST ROUTING or NO SHARING solution, depending on the version of the problem. Note that those bounds can be easily upgraded.

Consider the lower bound first. Obviously, $F_{min} = 0$ is valid. However, a better lower bound for a cost of a IDENTICAL RESERVATION solution is a cost of a DIFFERENT RESERVATION solution. The bound can be relatively easy obtained using the method of Section 7.3.1.

As far as the upper bound is concerned there are at least two different approaches that result in a bound of higher quality than the current upper bound. The straightforward one requires to solve IDENTICAL RESERVATION for any $\beta \in]\beta_{min}, \beta_{max}[$. Although the objective function $F^{IR}(D,\beta)$, for $\beta \in [\beta_{min}, \beta_{max}]$, does not have to be concave, it reaches its maximum for both β_{min} and β_{max}.

The second approach takes advantage of a fact that a solution to DIFFERENT RESERVATION provides a 2-approximation to IDENTICAL RESERVATION (see Section 8.1). In the implementation a method that hybridizes those two approaches has been used. First the optimal value of $\beta^\star$ for DIFFERENT RESERVATION was found, and then IDENTICAL RESERVATION was solved for this particular value of $\beta^\star$.

8.4.2 Golden ratio

In the algorithm of Section 8.2 the golden ratio search is used. This method finds the extremum of an unimodal function. The basic idea of the method is to evaluate values of

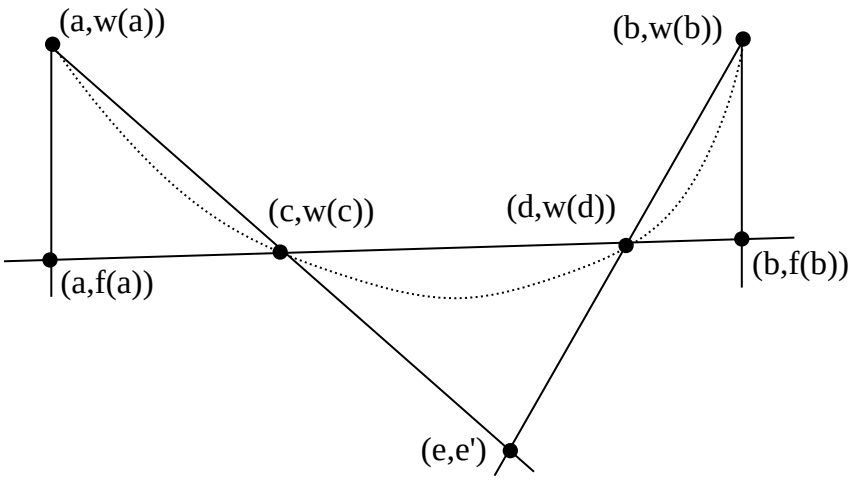


Figure 8.2: Evaluation of the lower bound of a convex function.

a considered function at endpoints of a considered interval (denoted by a and b) and for two values of β inside the interval (denoted by c and d). Assume that $a < c < d < b$. If $F(c) < F(d)$ it is possible, in the next step, to consider an interval $[a, d[$ only. Otherwise, an interval $]c, b]$ should be considered.

In Corollary 8.2.8 it is stated that $F^{IR/NS}(D, \beta)$ is convex in β, and it does not imply its unimodality (the function can be constant). Luckily, a fact if the function is unimodal can be verified in the first step of the golden ratio search, i.e., if results of the first four evaluations of the function are equal to each other the function is constant.

What is more, the lower bound of the function can be calculated each time it is evaluated for the four values of β. To simplify the notation use $F(\beta)$ instead of $F^{IR/NS}(D, \beta)$. Having points $(a, F(a))$, $(c, F(c))$, $(d, F(d))$, $(b, F(b))$ (see Figure 8.2) it is possible to define points (e, e'), $(a, f(a))$ and $(b, f(b))$. Point (e, e') is an intersection of two lines: first containing points $(a, F(a))$ and $(c, F(c))$, second containing points $(d, F(d))$ and $(b, F(b))$. Values $f(a)$ and $f(b)$ are evaluations of a linear function $f(\beta)$ containing points $(c, F(c))$ and $(d, F(d))$, for $\beta = a$ and $\beta = b$, respectively. If the considered function is convex, and this is the case with $F^{IR/NS}(D, \beta)$, $\min(e', f(a), f(b))$ is the lower bound for it.

Note that in the algorithm of Section 8.2 we look for all local minima of the cost function, and select the smallest one. In order to accelerate the procedure the following enhancement can be applied. If a lower bound for a local minimum in a considered interval is greater than a value of one of local minima, which has been already found, the consider interval should be abandoned, and the next interval should be considered instead.

Another enhancement is tightly connected to the previous one. The procedure can be additionally accelerated, if the true optimum is found as soon as possible. Having the true

optimum, and using the previously presented enhancement it is possible to perform less steps of the golden ratio search. In other words, the order in which intervals are considered matters. As in the numerical experiments used polytopes consisted of small number of extreme points, any sophisticated way of ordering the intervals was not applied. Therefore, it still remains an open question, how to arrange the intervals in order to accelerate the algorithm.

8.4.3 Column and constraint generation

The algorithms presented in Sections 8.1 and 8.2 consist of steps (subproblems) in which simpler variants of the considered problems are solved. Those subproblems can be solved independently. However, as they are usually strongly related to each other, information obtained during solving one of them should be used in order to facilitate solving another similar subproblem.

While solving the subproblems a column generation is used in order to obtain a set of promising paths. Consider two instances of Identical Reservation. They are identical as far as: network topology, demand polytope, and characteristics of arcs are concerned. The only difference between those two instances is a position of the hyperplane that divides a demand polytope into LHSP and RHSP. In such a situation it is highly probable that sets of promising paths obtained for those instances will be similar (or even identical). Therefore, in the implementation all paths obtained in the process of a column generation are collected, and used as a starting point for all successive subproblems.

Another piece of information obtained during solving the subproblems is a set of vital extreme points of the demand polytope. In this case a slightly different strategy is applied, and not all obtained extreme points are stored. The reason is that not all of them are true extreme points of D, because some of them can be created by the hyperplane, and are extreme points of both LHSP and RHSP but not D. Those extreme points are obsolete when the hyperplane changes its position, because they are either a convex combination of two extreme points of D or a convex combination of an extreme point of D and an extreme point of either the new LHSP or the new RHSP. Therefore, only extreme points that are not on the hyperplane should be stored.

Chapter 9

Volume oriented strategies

The major problem with the partitioning approaches presented in the previous chapters is that a routing can significantly differ between subsets, and implementing those changes can result in serious traffic fluctuations. Therefore, it is worth considering something that will make the routing changes less abrupt. Another drawback of the partitioning strategies is their centralized nature. They require a central node that has complete and up-to-date knowledge about all demands in a network. Moreover, this central node has to immediately inform all other nodes about routings they are supposed to execute. Those two disadvantages are addressed in this chapter by introducing a novel routing paradigm called the volume oriented strategy.

First, in Section 9.1, unrestricted volume oriented routing is discussed. The complexity of this approach is $\mathcal{NP}$-hard in general. However, it encompasses some special cases that are polynomial even when more that one hyperplane is taken into account. Those special cases, namely simplified volume oriented routing and general volume oriented routing, are presented in Sections 9.2 and 9.3, respectively.

9.1 Volume oriented routing

The first strategy, denoted by VO Routing, is an extension of Robust Routing in which demands can be routed differently depending on their actual volumes. In this case the solution consists of two different routings for each demand, and a set of thresholds. If a volume of a demand is smaller than a corresponding threshold the whole demand is routed using the first routing. If the volume is greater than the threshold a part of the demand equal to the threshold is sent using the first routing, while the rest of the demand uses the second routing. Note that the approach can be considered as a modification of Identical Reservation presented in previous sections. Instead of changing the routing for the whole demand after reaching the threshold only the amount of traffic exceeding the threshold is routed differently.

9.1.1 Notation

In order to formulate the problem the notation presented in Chapter 7 has to be extended by the following variables.

h_{ij} **:** threshold for a demand ij.

t'_{ij} **:** volume of a demand ij corresponding to a traffic demand matrix t that has to be routed using the first routing scheme.

t''_{ij} **:** volume of a demand ij corresponding to a traffic demand matrix t that has to be routed using the second routing scheme.

$\overline{x_p^{ij}}$ **:** routing scheme used by a demand ij to route the volume exceeding h_{ij} (second routing scheme).

$\overline{x_a^{ij}}$ **:** proportion of the volume exceeding h_{ij} flowing through an arc a.

9.1.2 Problem formulation

The problem of computing the minimum cost volume oriented routing of an uncertainty domain D, denoted by VO Routing (VO), can be formulated as follows.

Problem VO

$$\text{minimize } F^{VO}(D) = \sum_{a\in\mathcal{A}} \xi_a f_a, \tag{9.1a}$$

$$\sum_{p\in\mathcal{P}(i,j)} x_p^{ij} \geqslant 1, \qquad \forall i,j \in \mathcal{V}, \tag{9.1b}$$

$$\sum_{p\in\mathcal{P}(i,j)} \overline{x_p^{ij}} \geqslant 1, \qquad \forall i,j \in \mathcal{V}, \tag{9.1c}$$

$$\sum_{p\in\mathcal{P}(i,j),p\ni a} x_p^{ij} \leqslant x_a^{ij}, \qquad \forall i,j \in \mathcal{V},\ \forall a \in \mathcal{A}, \tag{9.1d}$$

$$\sum_{p\in\mathcal{P}(i,j),p\ni a} \overline{x_p^{ij}} \leqslant \overline{x_a^{ij}}, \qquad \forall i,j \in \mathcal{V},\ \forall a \in \mathcal{A}, \tag{9.1e}$$

$$\min(t_{ij}, h_{ij}) = t'_{ij}, \qquad \forall i,j \in \mathcal{V},\ \forall t \in D, \tag{9.1f}$$

$$\max(t_{ij} - h_{ij}, 0) = t''_{ij}, \qquad \forall i,j \in \mathcal{V},\ \forall t \in D, \tag{9.1g}$$

$$\sum_{i,j\in\mathcal{V}} (x_a^{ij} t'_{ij} + \overline{x_a^{ij}} t''_{ij}) \leqslant f_a, \qquad \forall a \in \mathcal{A},\ \forall t \in D, \tag{9.1h}$$

$$f_a \leqslant c_a, \qquad \forall a \in \mathcal{A}, \tag{9.1i}$$

$$h_{ij} \geqslant 0, \qquad \forall i,j \in \mathcal{V}, \tag{9.1j}$$

$$x_p^{ij},\ \overline{x_p^{ij}} \geqslant 0, \qquad \forall p \in \mathcal{P}(i,j),\ \forall i,j \in \mathcal{V}. \tag{9.1k}$$

In Problem (9.1) two different routings are considered for each demand (one used for the volume below the threshold and one used for the volume above the thresholds). Therefore, inequalities (7.1b) and (7.1c) have to be doubled into (9.1b), (9.1c) and (9.1d), (9.1e), respectively. The overall routing is expressed using (9.1f-9.1h). The equation (9.1f) defines the volume that has to be routed using the first routing scheme (variables x_p^{ij} and x_a^{ij}), the equation (9.1g) defines the volume that has to be routed using the second routing scheme (variables $\overline{x_p^{ij}}$ and $\overline{x_a^{ij}}$), while (9.1h) makes those volumes be sent with appropriate routings. Note that, Problems (9.1) and (7.1) are equivalent when $h_{ij} = 0$, for all $i, j \in \mathcal{V}$.

9.1.3 Computational complexity

Unfortunately a decision version of the problem is co-$\mathcal{NP}$-complete, so it is impossible to verify in polynomial time if a given solution is feasible. It will be proved in this section by reducing Subset Sum to the problem of verifying if a given solution is feasible. But first some lemmas will be proved.

Assume that a collection of numbers is given. Subset Sum consists in answering the question: is there a subset of those numbers whose sum is equal to a given limit. Subset Sum is $\mathcal{NP}$-complete and can be considered as a special case of the knapsack problem. [37]

Consider a collection of integer numbers $\mathcal{H} = \{l_1, l_2, \ldots, l_N\}$ and an integer limit $L \geqslant 1$. The instance of Subset Sum is denoted by $SS_{\mathcal{H}}$.

An instance $VO_{\mathcal{H}}$ of VO Routing corresponding to $SS_{\mathcal{L}}$ is modeled by means of a graph presented in Figure 9.1. Capacities of all links but one are unlimited. Only the capacity of a link u is limited, and is equal to $\epsilon \cdot (L - 0.5)$, where ϵ is a sufficiently small number, i.e., $0 < \epsilon < 0.5$. Routing costs are set to 0 (in fact those values are irrelevant, because the feasibility of a solution is being verified, and not its actual cost). The traffic demand polytope D describes possible volumes of N different demands st_i, where $i = 1, 2, \ldots, N$, each corresponding to one number from $\mathcal{H}$. The inequalities describing D are as follows.

$$t_{st_i} \leqslant l_i, \qquad \forall i \in \{1, 2, \ldots, N\}, \tag{9.2a}$$

$$\sum_{i=1}^{N} t_{st_i} \leqslant L. \tag{9.2b}$$

The following solution is considered. The thresholds h_{st_i}, for each demand st_i, where $i = 1, 2, \ldots, N$, are set to $l_i \cdot (1 - \epsilon)$. The first routing scheme (the one used for the volume below the threshold) for each demand uses link d, while the second routing scheme for each demand uses link u.

Let $\beta = (\beta_{ij})_{i,j \in \mathcal{V}}$, where $\beta_{ij} = 1$ or $\beta_{ij} = -1$ be any vector. Define D_β as follows: $D_\beta = D \cap \{t : \forall_{i,j \in \mathcal{V}} \beta_{ij}(t_{ij} - h_{ij}) \leqslant 0\}$. It is clear that $D = \bigcup_{\beta \in \{-1,1\}^{|\mathcal{V}|(|\mathcal{V}|-1)}} D_\beta$. The solution

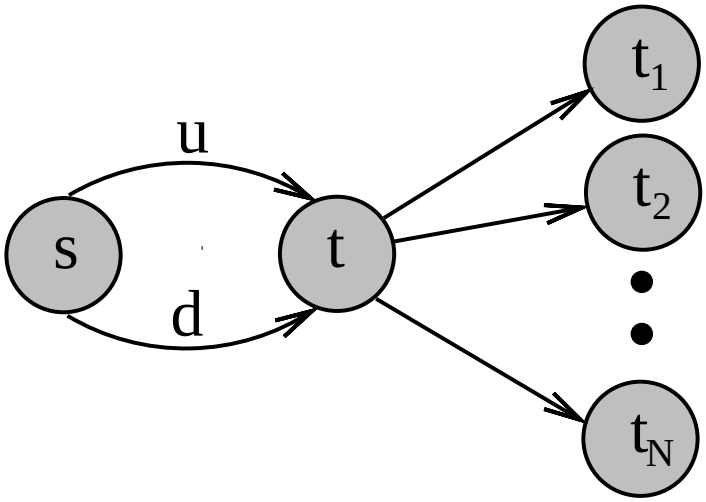


Figure 9.1: Graph proving that a decision version of VO Routing is co-$\mathcal{NP}$-complete.

considered above is not feasible if and only if there exists at least one $\beta \in \{-1,1\}^{|\mathcal{V}|(|\mathcal{V}|-1)}$ and a matrix $t \in D_\beta$ for which the routing is not feasible. Given any $\beta \in \{-1,1\}^{|\mathcal{V}|(|\mathcal{V}|-1)}$, the function $\max(t_{ij} - h_{ij}, 0))$ can be replaced by $t_{ij} - h_{ij}$ if $\beta_{ij} = -1$, or by 0 if $\beta_{ij} = 1$, while function $\min(t_{ij}, h_{ij})$ is equal to t_{ij} when $\beta_{ij} = 1$, and to h_{ij} in the other case. In other words, constraint (9.1h) is linear in t, and it has the same form for all matrices of D_β. Consequently, the existence of a matrix $t \in D_\beta$ for which the solution above is not feasible is equivalent to the existence of an extreme point of D_β for which the same holds. To check feasibility, it is then possible to consider only extreme points of the family of polytopes D_β where $\beta \in \{-1,1\}^{|\mathcal{V}|(|\mathcal{V}|-1)}$. This observation will be useful since the extreme points of D_β have a very simple structure.

Lemma 9.1.1. *The given solution to $VO_\mathcal{H}$ is feasible if and only if the answer to $SS_\mathcal{H}$ is NO.*

Proof. Suppose that the answer to $SS_\mathcal{H}$ is YES. Then the traffic demand polytope D contains at least one extreme point such that $\forall_{i=1,2,\ldots,N} t_{st_i} \in \{0, l_i\}$, and $\sum_{i=1}^{N} t_{st_i} = L$. Such an extreme point requires $\epsilon \cdot L$ of capacity at the link u, which is $0.5 \cdot \epsilon$ more than the available capacity. Therefore, the given solution to $VO_\mathcal{H}$ is not feasible.

Now assume that the solution is unfeasible. It will be proved that in such a case the answer to $SS_\mathcal{H}$ is YES. There exists a traffic demand matrix $t \in D$ and a routing corresponding to this matrix such that together they require more than $\epsilon \cdot (L - 0.5)$ of capacity on the link u. According to the observation above, the feasibility has to be only checked for the extreme points of the family of polytopes D_β. Each D_β is described by sets of constraints but only one of those constraints contains more than one variable. Therefore, for each extreme point of any D_β, an equation $t_{st_i} = \{l_i, l_i \cdot (1-\epsilon), 0\}$ can be not satisfied only for one t_{st_i}, where $i = 1, 2, \ldots, N$. Note that $l_i \cdot (1-\epsilon)$ is the value of the threshold for a demand st_i.

Assume that $t_{st_1} = l_1 \cdot (1-\epsilon)$. Note that in this case $t_{st_1} = h_{st_1}$ so the whole t_{st_1} is routed using the link d. The rest of the available demands' volume, i.e., at most $L - l_1 \cdot (1-\epsilon)$ [as

a result of (9.2b)], has to be routed in such a way that it will use more than $\epsilon \cdot (L - 0.5)$ of the capacity of the link u. But $\max_{t_{st_i} \in [0, l_i]} \frac{t_{st_i} - h_{st_i}}{t_{st_i}} = \epsilon$, for $i = 1, 2, \ldots, N$. Therefore, the remaining volume cannot use more than $\epsilon \cdot (L - l_1 + \epsilon \cdot l_1)$ of the capacity of the link u, and it is less than $\epsilon \cdot (L - 0.5)$, because it was assumed that l_1 is an integral number and $\epsilon < 0.5$.

Knowing that $\forall_{i=1,2,\ldots,N} t_{st_i} \neq l_i \cdot (1 - \epsilon)$, it is possible to conclude that for any extreme point of any D_β for which the routing solution is not feasible, an equation $t_{st_i} = \{l_i, 0\}$ has to be satisfied for at least all but one t_{st_i}. Assume that the equation is not satisfied for $i = N$. Then constraint (9.2b) is necessarily saturated. In other terms, we have $M + t_{st_N} = L$, where M is the sum of volumes of the other demands. Notice that this implies that t_{st_N} is integer. The demands other than t_{st_N} consume at most $\epsilon \cdot (L - t_{st_N})$ of the capacity of the link u, so at least $\epsilon \cdot (t_{st_N} - 0.5)$ has to be consumed by the demand st_N. Using (9.2a) and knowing that $h_{st_N} = l_N \cdot (1 - \epsilon)$ it is possible to write that $(t_{st_N} - h_{st_N}) \geqslant \epsilon \cdot (t_{st_N} - 0.5)$. In other words, we have $l_N \geqslant t_{st_N} \geqslant l_N - \frac{0.5 \cdot \epsilon}{1 - \epsilon}$. The facts that t_{st_N} is integer and $\frac{0.5 \cdot \epsilon}{1 - \epsilon} < 1$ imply that $t_{st_N} = l_N$. It means that the answer to $SS_{\mathcal{H}}$ is YES.

To finish the proof consider the case where $t_{st_i} = \{l_i, 0\}$ is satisfied for each traffic component. If the constraint (9.2b) is saturated, then the answer to $SS_{\mathcal{H}}$ is still YES. Suppose that $\sum_i t_{st_i} < L$. Since all t_{st_i} are integer in this case (either equal to 0 or l_i), the previous inequality becomes $\sum_i t_{st_i} \leqslant L - 1$. Then the traffic sent on the link u is less than $\epsilon \cdot (L - 1)$. However, the routing solution is not feasible implying that this traffic is more than $\epsilon \cdot (L - 0.5)$. It is clearly a contradiction. Said another way, the answer to $SS_{\mathcal{H}}$ is always YES whenever the proposed solution to $VO_{\mathcal{H}}$ is not feasible. □

Proposition 9.1.2. *A decision version of* VO Routing *is co-$\mathcal{NP}$-complete.*

Proof. The feasibility problem is in co-$\mathcal{NP}$ since a certificate of a negative response is given by any matrix that cannot be routed according to the proposed thresholds and routing schemes. It is easy to check whether a given matrix can be routing according to some given routing schemes. The reduction proposed in Lemma 9.1.1 ends the proof. □

As VO Routing is difficult, it will not be easy to solve it for real world networks in satisfying time. Fortunately, one of its special cases presented in the following section is polynomial. Moreover, according to Chapter 10, the special case (and its modification presented in Section 9.3) is very efficient.

9.2 Simplified volume oriented routing

Simplified volume oriented routing is a special case of VO Routing. In the latter case positions of the thresholds are subject to optimization, while in the former case those positions are set to the minimum values of the corresponding demands' volumes. The change makes the problem polynomial. On the other hand, the strategy is not efficient when there are not dominated traffic demand matrices in D that contain zero components.

9.2.1 Notation

In order to formulate the simplified volume oriented routing problem an additional set of constants is introduced.

t_{ij}^{min} **:** defined as $t_{ij}^{min} = \min_{t \in D} t_{ij}$.

9.2.2 Problem formulation

The problem of computing the minimum cost simplified volume oriented routing of an uncertainty domain D, denoted by SVO Routing (SVO), can be formulated as in (9.3).

Problem SVO

$$\text{minimize } F^{SVO}(D) = \sum_{a \in \mathcal{A}} \xi_a f_a, \tag{9.3a}$$

$$\sum_{p \in \mathcal{P}(i,j)} x_p^{ij} \geqslant 1, \qquad \forall i,j \in \mathcal{V}, \tag{9.3b}$$

$$\sum_{p \in \mathcal{P}(i,j)} \overline{x_p^{ij}} \geqslant 1, \qquad \forall i,j \in \mathcal{V}, \tag{9.3c}$$

$$\sum_{p \in \mathcal{P}(i,j), p \ni a} x_p^{ij} \leqslant x_a^{ij}, \qquad \forall i,j \in \mathcal{V},\ \forall a \in \mathcal{A}, \tag{9.3d}$$

$$\sum_{p \in \mathcal{P}(i,j), p \ni a} \overline{x_p^{ij}} \leqslant \overline{x_a^{ij}}, \qquad \forall i,j \in \mathcal{V},\ \forall a \in \mathcal{A}, \tag{9.3e}$$

$$t_{ij}^{min} = t'_{ij}, \qquad \forall i,j \in \mathcal{V},\ \forall t \in D, \tag{9.3f}$$

$$t_{ij} - t_{ij}^{min} = t''_{ij}, \qquad \forall i,j \in \mathcal{V},\ \forall t \in D, \tag{9.3g}$$

$$\sum_{i,j \in \mathcal{V}} (x_a^{ij} t'_{ij} + \overline{x_a^{ij}} t''_{ij}) \leqslant f_a, \qquad \forall a \in \mathcal{A},\ \forall t \in D, \tag{9.3h}$$

$$f_a \leqslant c_a, \qquad \forall a \in \mathcal{A}, \tag{9.3i}$$

$$x_p^{ij},\ \overline{x_p^{ij}} \geqslant 0, \qquad \forall p \in \mathcal{P}(i,j),\ \forall i,j \in \mathcal{V}. \tag{9.3j}$$

The formulation (9.3) modifies (9.1) by simplifying definitions of t'_{ij} and t''_{ij}, i.e., it replaces (9.1f) with (9.3f), and (9.1g) with (9.3g). Moreover, as threshold variables h_{ij} are not used, inequalities (9.1j) are obsolete in (9.3). Notice that, if $t_{ij}^{min} = 0$, for all $i,j \in \mathcal{V}$, then SVO Routing is equivalent to Robust Routing.

9.2.3 Computational complexity

The problem is polynomial, and can be easily solved using techniques presented in Chapter 7, i.e., using an algorithm based on constraint generation (see [8, 9]), and generating paths in an iterative way.

9.3 General volume oriented routing

General volume oriented routing can be seen as an extension of SVO Routing, with modified (more general) division of demands between routings. In fact, it is a generalization of

SVO Routing, so it cannot be less efficient. On the other hand, it is more complicated to implement.

9.3.1 Notation

In order to formulate the problem an additional set of constants has to be introduced.

t_{ij}^{max} **:** defined as $t_{ij}^{max} = \max_{t \in D} t_{ij}$.

9.3.2 Problem formulation

The formulation of the problem of computing the minimum cost general volume oriented routing of an uncertainty domain D, denoted by GVO Routing (GVO), is alike the formulation of SVO Routing, thus alike the formulation of VO Routing. The only differences are in definitions of t'_{ij} and t''_{ij}, i.e., constraint (9.3f-9.3g) are replaced by (9.4f-9.4g).

Problem GVO

$$\text{minimize } F^{GVO}(D) = \sum_{a \in \mathcal{A}} \xi_a f_a, \tag{9.4a}$$

$$\sum_{p \in \mathcal{P}(i,j)} x_p^{ij} \geqslant 1, \qquad \forall i,j \in \mathcal{V}, \tag{9.4b}$$

$$\sum_{p \in \mathcal{P}(i,j)} \overline{x_p^{ij}} \geqslant 1, \qquad \forall i,j \in \mathcal{V}, \tag{9.4c}$$

$$\sum_{p \in \mathcal{P}(i,j), p \ni a} x_p^{ij} \leqslant x_a^{ij}, \qquad \forall i,j \in \mathcal{V},\ \forall a \in \mathcal{A}, \tag{9.4d}$$

$$\sum_{p \in \mathcal{P}(i,j), p \ni a} \overline{x_p^{ij}} \leqslant \overline{x_a^{ij}}, \qquad \forall i,j \in \mathcal{V},\ \forall a \in \mathcal{A}, \tag{9.4e}$$

$$\alpha_{ij} t_{ij} + \beta_{ij} = t'_{ij}, \qquad \forall i,j \in \mathcal{V},\ \forall t \in D, \tag{9.4f}$$

$$(1 - \alpha_{ij}) t_{ij} - \beta_{ij} = t''_{ij}, \qquad \forall i,j \in \mathcal{V},\ \forall t \in D, \tag{9.4g}$$

$$\sum_{i,j \in \mathcal{V}} (x_a^{ij} t'_{ij} + \overline{x_a^{ij}} t''_{ij}) \leqslant f_a, \qquad \forall a \in \mathcal{A},\ \forall t \in D, \tag{9.4h}$$

$$f_a \leqslant c_a, \qquad \forall a \in \mathcal{A}, \tag{9.4i}$$

$$x_p^{ij},\ \overline{x_p^{ij}} \geqslant 0, \qquad \forall p \in \mathcal{P}(i,j),\ \forall i,j \in \mathcal{V}. \tag{9.4j}$$

Where vectors α and β are given, and satisfy the following set of constraints.

$$\alpha_{ij} t_{ij}^{min} + \beta_{ij} \geqslant 0, \qquad \forall i,j \in \mathcal{V}, \tag{9.5a}$$

$$(1 - \alpha_{ij}) t_{ij}^{min} - \beta_{ij} \geqslant 0, \qquad \forall i,j \in \mathcal{V}, \tag{9.5b}$$

$$\alpha_{ij} t_{ij}^{max} + \beta_{ij} \geqslant 0, \qquad \forall i,j \in \mathcal{V}, \tag{9.5c}$$

$$(1 - \alpha_{ij}) t_{ij}^{max} - \beta_{ij} \geqslant 0, \qquad \forall i,j \in \mathcal{V}. \tag{9.5d}$$

It is obvious that SVO Routing is a special case of GVO Routing, where $\alpha_{ij} = 0$ and $\beta_{ij} = t_{ij}^{min}$, for each $i, j \in \mathcal{V}$. An interesting observation is that for each D there exists a simple way to obtain an optimal pair of vectors α and β.

Proposition 9.3.1. *For a given D vectors α and β defined in* (9.6) *yield an optimal solution, i.e., there are no cheaper solutions to* GVO Routing *for a given D and any other α and β.*

$$\alpha_{ij} = \frac{-t_{ij}^{min}}{t_{ij}^{max} - t_{ij}^{min}}, \qquad \forall i, j \in \mathcal{V}, \tag{9.6a}$$

$$\beta_{ij} = \frac{t_{ij}^{max} t_{ij}^{min}}{t_{ij}^{max} - t_{ij}^{min}}, \qquad \forall i, j \in \mathcal{V}. \tag{9.6b}$$

Proof. Consider a capacity consumed by only one demand ij. The constraints defining the used capacity on a link a for GVO Routing can be combined into:

$$x_a^{ij}(\alpha_{ij} t_{ij} + \beta_{ij}) + \overline{x_a^{ij}}[(1 - \alpha_{ij})t_{ij} - \beta_{ij}],$$

and for α and β defined as (9.6) into:

$$z_a^{ij} t_{ij}^{min} \frac{t_{ij}^{max} - t_{ij}}{t_{ij}^{max} - t_{ij}^{min}} + \overline{z_a^{ij}} t_{ij}^{max} \frac{t_{ij} - t_{ij}^{min}}{t_{ij}^{max} - t_{ij}^{min}}.$$

For the sake of clearness, in the latter case, z_a^{ij} was substituted for x_a^{ij} and $\overline{z_a^{ij}}$ for $\overline{x_a^{ij}}$. Assume that the routing for GVO Routing (with α and β denoted by α' and β')is known and expressed using vectors x and $\overline{x}$. In order to obtain the same results for all $t \in D$ and suggested α and β take:

$$z_a^{ij} = (\alpha'_{ij} + \frac{\beta'_{ij}}{t_{ij}^{min}}) x_a^{ij} + (1 - \alpha'_{ij} - \frac{\beta'_{ij}}{t_{ij}^{min}}) \overline{x_a^{ij}},$$

$$\overline{z_a^{ij}} = (\alpha'_{ij} + \frac{\beta'_{ij}}{t_{ij}^{max}}) x_a^{ij} + (1 - \alpha'_{ij} - \frac{\beta'_{ij}}{t_{ij}^{max}}) \overline{x_a^{ij}}.$$

It is an easy exercise to check that the obtained loads on each link will be exactly like for GVO Routing. Moreover, the routing is feasible, because both z_a^{ij} and $\overline{z_a^{ij}}$ are convex combinations of x_a^{ij} and $\overline{x_a^{ij}}$, for $a \in \mathcal{A}$ and $i, j \in \mathcal{V}$. Note that $0 \leqslant \alpha'_{ij} + \frac{\beta'_{ij}}{t_{ij}^{min}} \leqslant 1$, and $0 \leqslant \alpha'_{ij} + \frac{\beta'_{ij}}{t_{ij}^{max}} \leqslant 1$, for all $i, j \in \mathcal{V}$, because of (9.5). □

The proof contains an observation that will be interesting in the following section of the thesis, and now will be stated as a corollary.

Corollary 9.3.2. *Each solution to* GVO Routing *can be transformed to another solution to* GVO Routing *with the vectors α and β defined like in* (9.6)*. The solution will be of the same cost, and it will require the same amount of capacity at each link.*

9.3.3 Computational complexity

Alike SVO Routing, GVO Routing is polynomial, and can be easily solved using techniques presented in Section 7, i.e., using an algorithm based on constraint generation (see [8, 9]), and generating paths in an iterative way.

Chapter 10

Numerical results

In this chapter numerical results are presented. They show applicability of the algorithms and strategies presented in the second part of the thesis. The algorithms were implemented using Visual C++ and tested on a one core Intel 2.4 GHz CPU with 3.25 GB RAM using a linear programming solver CPLEX 11.0 [35].

10.1 Test cases

Example cases were built using real world networks available in SNDlib [52]. The following topologies were used: *atlanta*, *france*, and *cost266*. All were tested using two different set of active nodes $\mathcal{V}^{act}$, and three to five different traffic demand polytopes defined for each of the set. The first polytope for each set satisfied restrictions of the hose model presented in [23], and described in Section 7.2.1. The sets of active nodes, and the bounds for incoming and outgoing traffic, were chosen at random. In the first part of the experiments standard hose model polytopes were used, while in the second part a general hose model was considered. In the latter case the minimum traffic between a pair of node i and j, where $i, j \in \mathcal{V}^{act}$, was set to $\frac{\min(A_i, B_j)}{2 \cdot |\mathcal{V}^{act}|}$, while the maximum traffic was set to $\frac{\min(A_i, B_j)}{2}$, where A_i and B_j are defined as in Section 7.2.1. The last two polytopes were built using the B-S polytope model presented in Section 7.2.2. The bounds were set at random, while k was set to 10% or 20% of its maximum value, i.e., 0.1 or 0.2 times $|\mathcal{V}^{act}| \cdot (|\mathcal{V}^{act}| - 1)$, depending on the test case. Note that B-S model polytopes were used only in the second part of the experiments. The resulting test cases can be seen in Table 10.1. It consists of six columns.

: identification number of a test case.

topology : name of the topology in SNDLib.

$\mathcal{V}$: number of nodes in a network.

$\mathcal{A}$: number of undirected links in a network.

dim. D **:** dimension of a traffic demand polytope, i.e., $|\mathcal{V}^{act}| \cdot (|\mathcal{V}^{act}| - 1)$.

variant : way a traffic demand polytope was created, hose — full hose model polytope, 3 pts. (10 pts.) — convex hull of three (ten) random extreme points of a full hose model polytope, B-S 10% (20%) — B-S polytope model with $k = \max(1, \lfloor K \cdot |\mathcal{V}^{act}| \cdot (|\mathcal{V}^{act}| - 1) \rfloor)$, where $K = 0.1(0.2)$.

Note that neither the full hose model polytopes nor the B-S model polytopes were considered for *cost266* topology. The reason is that the complexity of those polytopes is enormous, and it was impossible to solve the simplest Robust Routing (without any partitioning) for them in reasonable time. Note that in the thesis the hose models with bounds on both incoming and outgoing traffic for each active node are considered (in [8, 9] only bounds on outgoing traffic were considered). The number of extreme points of those polytopes was also the reason not to consider them for Dynamic Routing.

10.2 Partitioning strategies

In the first phase of the experiments the partitioning strategies (Different Reservation and Identical Reservation for Robust Routing, No Sharing, and Dynamic Routing) are considered. First the congestion for different traffic demand polytopes and different basic problems (Robust Routing, No Sharing and Dynamic Routing) is computed. Note that in this phase the polytopes are not partitioned in any way. The results are presented in Table 10.2. It consists of nine columns. The first six are identical to the columns in Table 10.1. The following three columns present the congestion for each test case and each basic problem. Note that the congestion is scaled in such a way that $z = 1.0$ is obtained for Robust Routing and the traffic demand polytope that is a convex hull of three extreme points of the full hose polytope. Therefore, they are comparable only within a single configuration, and not between different configurations (a configuration is characterized by the same network and the same number of demands). Also notice that in this part of the experiments B-S model polytopes were not considered. What is more, considered hose model polytopes are standard, and not general like in the case of volume oriented strategies presented in the next part of the experiments.

As expected, and clearly shown in Table 10.2, implementing No Sharing results in the greatest congestion, while an implementation of Dynamic Routing is the least expensive. An interesting observation is that the difference between the congestions for Robust Routing and No Sharing increases with the complexity of a traffic demand polytope. On the other hand, the difference between Robust Routing and Dynamic Routing does not

Table 10.1: Test cases

#	topology	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{A}\rvert$	dim. D	variant
1	*atlanta*	15	44	12	3 pts.
2					10 pts.
3					hose
4					B-S 10%
5					B-S 20%
6				30	3 pts.
7					10 pts.
8					hose
9					B-S 10%
10					B-S 20%
11	*france*	25	90	20	3 pts.
12					10 pts.
13					hose
14					B-S 10%
15					B-S 20%
16				56	3 pts.
17					10 pts.
18					hose
19					B-S 10%
20					B-S 20%
21	*cost266*	37	114	90	3 pts.
22					10 pts.
23				156	3 pts.
24					10 pts.

Table 10.2: Congestion

network						congestion z		
#	topology	$\|\mathcal{V}\|$	$\|\mathcal{A}\|$	dim. D	variant	Robust	No sharing	Dynamic
1					3 pts.	1.00	1.48	0.87
2				12	10 pts.	1.06	1.48	0.87
3	*atlanta*	15	44		hose	1.06	1.48	-
6					3 pts.	1.00	1.56	0.97
7				30	10 pts.	1.20	2.54	1.20
8					hose	1.20	2.66	-
11					3 pts.	1.00	1.80	0.96
12				20	10 pts.	1.05	2.38	1.05
13	*france*	25	90		hose	1.05	2.51	-
16					3 pts.	1.00	2.20	1.00
17				56	10 pts.	1.20	3.86	1.20
18					hose	1.98	3.91	-
21				90	3 pts.	1.00	1.92	1.00
22	*cost266*	37	114		10 pts.	1.43	3.97	1.43
23				156	3 pts.	1.00	2.38	1.00
24					10 pts.	1.10	4.57	1.10

seem to be significant. The observations justify introducing the partitioning of the traffic demand polytope for No Sharing, and questions the significance of introducing partitioning for Robust Routing (results for Dynamic Routing are a lower bound for Identical Reservation for Robust Routing).

Although the partitioning can be meaningless for Robust Routing when minimizing the congestion, it is profitable when minimizing routing cost (see further experiments in Table 10.3), especially when appropriate conditions occur, i.e., networks are congested.

Having the congestion the efficiency of different partitioning strategies was tested. Moreover, running times of algorithms capable of solving the considered problems were also checked. In this phase routing costs were used as cost functions.

In order to compare the strategies on equal terms the traffic demand polytopes were scaled in such a way that the congestion z for each of them would be 1.1 if those scaled polytopes had been used in the previous phase of the experiments. For instance, for the *france* network with 56 demands while solving different partitioning problems for Robust

Routing all traffic matrices from the traffic demand polytope which is a full hose model polytope (test case #18) are multiplied by $\frac{1.1}{1.98}$, while all traffic matrices from the traffic demand polytope which is a convex hull of 10 extreme points of the full hose polytope (test case #17) are multiplied by $\frac{1.1}{1.20}$. For the same test cases and No Sharing the traffic demand polytope is even more diminished, and all the traffic matrices are multiplied by $\frac{1.1}{3.91}$ and $\frac{1.1}{3.86}$, respectively.

In order to handle the insolvability of the test cases (they are built in such a way that all networks are congested) additional uncapacitated arcs are introduced between all pairs of active nodes. Routing costs of these additional arcs are set in such a way that it is approximately twenty times more expensive to use them than paths consisting of arcs in the original network. Those arcs can be considered as possibilities to rent capacity from other operators or as links set up using different, and more expensive, technology, e.g., radio links or satellite links. Note that, those additional expensive links are not included in Table 10.1 (column $\mathcal{V}$).

For each topology and each polytope up to five different directions α of the hyperplane were used. The presented result correspond to the most profitable direction of the hyperplane.

In Table 10.3 the first part of the results is presented. The first column (#) contains identification numbers of test cases, and is the same as in Table 10.1. The following five columns present results obtained for five different variants of the problem, i.e., Identical Reservation and Different Reservation both for Robust Routing and No Sharing, and Different Reservation for Dynamic Routing. Assume that u is a cost of an optimal solution to one of the basic problems without any partitioning, and v is a cost of an optimal solution to the same problem with partitioning (either Identical Reservation or Different Reservation). In the table gains are presented. They are understood as $\frac{u-v}{u} \cdot 100\%$.

Another table, namely Table 10.4, presents running times of the presented algorithms that solve various partitioning problems. Its construction is similar to the construction of Table 10.3.

All versions of Different Reservation were solved by means of the method of Section 7.3.1. Identical Reservation for Robust Routing was solved using the double binary search algorithm of Section 8.1, while Identical Reservation for No Sharing was solved using both the double binary search method and the algorithm of Section 8.2. That is why the execution times for this version of the problem are presented as: a/b, where a is time used by the double binary search algorithm, and b is time used by the algorithm of Section 8.2. Note that the algorithm of Section 8.2 cannot be used for the hose model polytopes.

Table 10.3: Efficiency of the partitioning algorithms—gain [%]

	Robust		No sharing		Dynamic
#	Different	Identical	Different	Identical	Different
1	9.7	5.6	48.3	44.3	1.3
2	23.6	15.9	52.4	44.8	0.8
3	19.4	15.9	47.0	44.8	-
6	15.4	12.8	50.0	46.0	6.2
7	10.4	0.8	52.5	47.2	8.6
8	4.6	2.2 - 2.5	29.2	27.4	-
11	11.5	7.9	52.0	45.7	3.2
12	2.8	1.1	43.1	40.3	1.5
13	3.8	0.4	41.5	40.0	-
16	15.5	12.7	50.3	46.3	8.2
17	15.3	8.2	41.3	39.2	7.3
18	0.0	0.0	31.2	30.2	-
21	10.8	5.4	54.8	47.7	4.7
22	1.1	0.2-0.7	54.5	51.2	2.1
23	7.5	1.1	66.6	61.1	7.5
24	2.9	0.0-2.9	53.4	49.5	4.2

The 9000-second time limit has been hit three times. Identical Reservation for Robust Routing has not been solved to optimality for the test cases #8, #22, and #24. In those cases, the gains are presented as: $a - b$, where a is the gain for the best solution found, and b is the upper bound. In other words, the optimal gain belongs to $[a, b]$. Note that time limits were not used for Dynamic Routing.

Notice that the gain for No Sharing is significant, while the gain for Dynamic Routing is rather small. The reason is that No Sharing in its nature is very restrictive, and any mechanism that can relax its resource reservation rules is of great importance. On the other hand, Dynamic Routing is very flexible. Therefore, there is no place for significant improvements.

As shown in Table 10.3, Different Reservation almost always outperforms Identical Reservation. The only case when these two strategies provide similar results is test case #18 for Robust Routing (gain for both is 0). In other cases the difference in gains seems to be stable, and it oscillates near 5%.

Table 10.4: Running times of the partitioning algorithms [s]

	Robust		No sharing		Dynamic
#	Different	Identical	Different	Identical	Different
1	0.4	5.1	0.3	0.5 / 0.2	1.3
2	0.5	7.0	0.3	0.6 / 0.3	19.4
3	0.6	8.9	0.4	0.7	-
6	1.3	20.4	0.8	2.8 / 1.0	6.2
7	6.2	302.8	0.8	3.4 / 1.5	45.6
8	31.8	9000.0	1.5	3.6	-
11	4.3	31.1	1.4	4.0 / 2.0	5.3
12	9.8	180.4	2.9	5.6 / 3.7	26.0
13	19.3	1132.9	1.7	2.8	-
16	13.3	111.6	10.2	17.8 / 8.6	26.7
17	98.1	2873.3	9.3	17.2 / 11.0	983.0
18	3016.4	3016.4	10.2	18.8	-
21	46.1	354.0	13.7	64.4 / 45.3	106.4
22	517.7	9000.0	39.5	127.7 / 79.5	6808.9
23	148.3	1771.1	37.3	124.7 / 132.0	669.1
24	4024.3	9000.0	137.4	594.7 / 350.7	12019.3

It is also worth to notice that the running times of the double binary search algorithm presented in Section 8.1 are not significantly larger than the running times of the polynomial algorithm of Section 8.2 (see Table 10.4, Identical Reservation for No Sharing). It justifies the claim that the double binary search algorithm, although non-polynomial, is practical, and can be applied even to medium size networks.

10.3 Volume oriented strategies

In the second part of the experiments volume oriented strategies are considered, i.e, SVO Routing and GVO Routing. Notice that VO Routing was not tested because of its complexity.

An evaluation strategy used in this section is similar to the one used for the partitioning strategies. First, the polytopes were scaled in such a way that the congestion, using Robust Routing, obtained for the most loaded link was 110%. Then, in order to handle the

insolvability of the test cases additional uncapacitated arcs between all pairs of active nodes were introduced. Routing costs of these additional arcs were set in such a way that it was approximately twenty times more expensive to use them than paths consisting of arcs in the original network.

In this part of the experiments all test cases were used (including B-S polytope model). As far as the hose model is concerned, the general hose model was considered. The reason is that both SVO Routing and GVO Routing require lower bounds on demands' volumes in order to work appropriately.

The result of this part of the experiments are presented in Table 10.5. It consists of nine columns. The first contains identification numbers of test cases, while the rest show execution times and costs for the following cases.

Robust : strategy described in Section 7.1.1. It is the simplest routing strategy and an upper bound on costs of all other strategies.

SVO Routing : strategy presented in Section 9.2.

GVO Routing : strategy presented in Section 9.3.

It is clearly seen that both volume oriented strategies are efficient, and they outperform a corresponding partitioning strategy, i.e., Identical Reservation for Robust Routing. However, they take advantage of a special structure of traffic demand polytopes (lack of non-dominated traffic demand matrices with zero components), and they cannot outperform the partitioning strategies in general test cases. It is also worth to notice that GVO Routing provides better results than SVO Routing only in five cases, and only in two of them (#6 and #17) the difference is not negligible.

What is more, computational times are also very good. They prove that the volume oriented strategies can be successfully implemented in medium size networks. The 9000 second time limit was hit only three times (twice for #18 and once for #20), and in all those cases feasible solutions to the problem had been first obtained (that is a reason why the results are presented using a sign $\leqslant$).

Table 10.5: Efficiency and running times of the volume oriented strategies

	Robust		SVO Routing			GVO Routing		
#	cost	time[s]	cost	gain[%]	time[s]	cost	gain[%]	time[s]
1	723.0	0.1	717.5	0.8	0.4	717.5	0.8	0.2
2	447.5	0.2	332.0	25.8	0.4	332.0	25.8	0.3
3	442.3	0.3	380.3	14.0	0.5	380.3	14.0	0.5
4	361.1	0.1	287.9	20.3	0.2	287.9	20.3	0.2
5	343.0	0.2	291.8	14.9	0.3	291.8	14.9	0.5
6	410.7	0.6	395.2	3.8	0.7	393.5	4.2	0.9
7	448.8	0.7	420.5	6.3	1.6	420.4	6.3	2.5
8	443.5	3.1	402.2	9.3	7.0	402.1	9.3	9.9
9	625.7	1.0	350.6	44.0	6.1	350.6	44.0	10.5
10	506.1	1.4	448.0	11.5	11.0	448.0	11.5	25.9
11	332.4	0.6	317.8	4.4	0.7	317.8	4.4	0.8
12	353.6	0.8	324.1	8.3	0.8	324.1	8.3	1.6
13	357.0	1.6	328.9	7.9	1.9	328.9	7.9	3.0
14	394.5	1.4	371.5	5.8	2.7	371.5	5.8	4.3
15	483.6	1.7	477.9	1.2	4.9	477.9	1.2	8.8
16	652.3	3.6	619.6	5.0	3.7	619.3	5.0	3.4
17	695.4	8.6	574.2	17.4	22.7	565.1	18.7	37.3
18	649.2	495.0	525.4	19.1	1336.4	525.4	19.1	3401.0
19	854.5	33.2	802.8	6.1	177.4	802.8	6.1	765.1
20	518.7	11.5	516.3	0.0	605.7	$\leqslant$516.3	$\geqslant$0.0	7200.0

Chapter 11

Conclusions

In the thesis two different partitioning strategies were presented, i.e., Identical Reservation and Different Reservation. They were applied to three different basic models, i.e., Robust Routing, No Sharing, and Dynamic Routing. Moreover, three different volume oriented strategies were presented, i.e., VO Routing, SVO Routing, and GVO Routing. They were applied to one basic model, i.e., Robust Routing.

The computational complexity of different combinations of these basic models, partitioning strategies, and volume oriented strategies were discussed, e.g., a special case of Dynamic Routing was presented, which is polynomial even when Different Reservation is considered for it.

For the partitioning problems two algorithms were proposed: the double binary search algorithm that solves Identical Reservation, and another (polynomial time) algorithm that solves Identical Reservation for No Sharing when the traffic demand polytope is given by a set of its extreme points. The presented algorithms (and a method that solves Different Reservation introduced earlier in the literature) were thoroughly tested, and proved to be applicable to networks of practical sizes. Differences in network utilization's efficiency of the three basic problems were shown. What is more, possible gains that result from the implementation of different partitioning strategies were also presented.

In the thesis the direction of the hyperplane used for partitioning is assumed to be known (not subject to optimization), and in the experiments hyperplanes of a particular form were used, i.e., traffic between a given pair of nodes is a hyperplane's direction. But, in general, it is possible to use any hyperplane's direction, even one that is not directly connected to traffic, e.g., time.

The presented algorithms can be easily extended to solve a problem when the traffic demand polytope is divided by many hyperplanes of the same direction. However, a more general problem that consists in providing an optimal set of hyperplanes of *unknown* directions still remains a challenge.

As far as the volume oriented strategies are concerned, it was shown in the thesis that

the problem of providing an optimal solution satisfying VO Routing is difficult in general. However, a way to cope with this difficulty was also presented, i.e., SVO Routing and GVO Routing were introduced, which are polynomially solvable modifications of VO Routing. It was shown that SVO Routing cannot outperform GVO Routing as far as cost is concerned, but on the other hand, it can be implemented easier. Moreover, those strategies can be equally effective when some special polytopes are considered, e.g., B-S polytope model. Finally, numerical results proving the applicability and efficiency of the introduced volume oriented approaches were presented.

In the thesis different strategies that were to merge the efficiency of the dynamic routing with the simplicity of the robust stable routing were presented. Although the obtained results are satisfactory, there is still a lot of space for improvements. The only theoretical bound for the efficiency of novel strategies to be proposed is the efficiency of the dynamic routing. The bound has not been reached by any solution that is solvable in polynomial time. Thus, the research field is by no mean exhausted.

Index

2-disjoint paths, 29

active nodes, 85

backup capacity, 2, 10
backup flow, 10
backup paths, 11
basic capacity, 10
Bertsimas-Sim model, 4, 59, 85
bottom link, 19

column generation, 3
congestion, 88
CPLEX, 3, 85

demand-wise shared protection, 13
different reservation vectors, 60
diversification, 13
double binary search algorithm, 66
DWDM, 1
dynamic routing, 4, 57, 72

failure scenario, 10
failure state, 10
failure-dependent restoration, 2, 11, 45
failure-disjoint, 11, 17
failure-independent restoration, 2, 24
fractional graph coloring, 18

general volume oriented routing, 82
golden ratio, 74

Hausdorff distance, 62
hose model, 4, 59, 85
hot-standby, 1, 12

identical reservation vectors, 63
IP, 1

left hand side polytope, 60
left polytope, 70

middle polytope, 70
MIP, 12, 22, 45
multiple failures, 1
multiple-link failure scenario, 10

no sharing routing, 56
normal state, 2, 10

partitioning, 28, 46
path diversity, 13
path protection, 1, 12
path restoration, 2, 24, 45
polyhedral model, 4
pricing problem, 3, 13, 21, 32, 48
primary capacity, 10
primary flow, 10
primary path, 11
protection capacity, 2, 10
protection path, 11

restoration flow, 10
restoration path, 11
right hand side polytope, 60
robust optimization, 4
robust stable routing, 4, 55

set cover, 40
set splitting, 13

shared risk resource groups, 2
simplified volume oriented routing, 81
single-link failure scenario, 10
SPPRC, 22
stochastic programming, 4
stub-release, 2, 24, 33, 48
subset sum problem, 79
Suurballe algorithm, 13, 26

top link, 19
traffic demand polytope, 5, 54
traffic matrix, 54

volume oriented routing, 5, 77, 91

working capacity, 10
working flow, 10
working path, 11

Bibliography

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

[2] A. Altın, E. Amaldi, P. Belotti, and M.Ç. Pınar. Provisioning virtual private networks under traffic uncertainty. *Networks*, 49(1):100–115, 2007.

[3] G.R. Ash. *Dynamic routing in telecommunications networks*. McGraw-Hill Professional, 1997.

[4] A. Bashllari and D. Nace. A study on two new protection strategies. In *Proceedings of the IEEE International Workshop on IP Operations and Management*, 2008.

[5] A. Bashllari, D. Nace, E. Gourdin, and O. Klopfenstein. The MMF rerouting computation problem. In *Proceedings of the International Network Optimization Conference*, 2007.

[6] W. Ben-Ameur. Between fully dynamic routing and robust stable routing. In *Proceedings of the International Workshop on the Design of Reliable Communication Networks*, 2007.

[7] W. Ben-Ameur and H. Kerivin. Routing of uncertain demands. In *Proceedings of the INFORMS Telecommunications Conference*, 2001.

[8] W. Ben-Ameur and H. Kerivin. New economical virtual private networks. *Communications of the ACM*, 46(6):69–73, 2003.

[9] W. Ben-Ameur and H. Kerivin. Routing of uncertain traffic demands. *Optimization and Engineering*, 6(3):283–313, 2005.

[10] W. Ben-Ameur and T.T.L. Pham. Design of survivable networks based on end-to-end rerouting. In *Proceedings of the International Workshop on the Design of Reliable Communication Networks*, 2001.

[11] W. Ben-Ameur and M. Żotkiewicz. Robust routing and optimal partitioning of a traffic demand polytope. *International Transactions in Operational Research*. Available in Early View.

[12] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25(1):1–13, 1999.

[13] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical Programming*, 98(1-3):49–71, 2003.

[14] R. Bhandari. *Survivable Networks – Algorithms for Diverse Routing*. Kluwer, 1999.

[15] G. Brightwell, G. Oriolo, and F.B. Shepherd. Reserving resilient capacity in a network. *SIAM Journal on Discrete Mathematics*, 14(4):524–539, 2001.

[16] R.E. Burkard, H. Dollani, and P.T. Thach. Linear approximations in a dynamic programming approach for the uncapacitated single-source minimum concave cost network flow problem in acyclic networks. *Journal of Global Optimization*, 19(2):121–139, 2001.

[17] T. Carpenter, D.P. Heyman, and I. Saniee. Studies of random demands on network costs. *Telecommunication Systems*, 10(3-4):409–421, 1998.

[18] C. Chekuri. Routing and network design with robustness to changing or uncertain traffic demands. *SIGACT News*, 38(3):106–129, 2007.

[19] C. Chekuri, F.B. Shepherd, G. Oriolo, and M.G. Scutellá. Hardness of robust network design. *Networks*, 50(1):50–54, 2007.

[20] D. Coudert, P. Datta, S. Perennes, H. Rivano, and M-E. Voge. Shared risk resource group complexity and approximability issues. *Parallel Processing Letters*, 17(2):169–184, 2007.

[21] G. Dahl and M. Stoer. A cutting plane algorithm for multicommodity survivable network design problems. *INFORMS Journal on Computing*, 10(1):1–11, 1998.

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

[23] N.G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K.K. Ramakrishnan, and J.E. Van Der Merwe. A flexible model for resource management in virtual private networks. In *Proceedings of the ACM SIGCOMM*, 1999.

[24] J. Edmonds and R.M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972.

[25] A. Feldmann, A.C. Gilbert, P. Huang, and W. Willinger. Dynamics of IP traffic: A study of the role of variability and the impact of control. In *Proceedings of the ACM SIGCOMM*, 1999.

[26] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: Methodology and experience. *IEEE/ACM Transactions on Networking*, 9(3):265–279, 2001.

[27] J.A. Fingerhut, S. Suri, and J.S. Turner. Designing least-cost nonblocking broadband networks. *Journal of Algorithms*, 24(2):287–309, 1997.

[28] S. Fortune, J.E. Hopcroft, and J.C. Wyllie. The directed subgraph homeomorphism problem. Technical report, Cornell University, 1978.

[29] M.R. Garey and D.S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, 1979.

[30] N. Goyal, N. Olver, and F.B. Shepherd. Dynamic vs. oblivious routing in network design. In *Proceedings of the European Symposia on Algorithms*, 2009.

[31] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[32] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer Verlag, 1988.

[33] G.M. Guisewite and P.M. Pardalos. Minimum concave-cost network flow problems: applications, complexity, and algorithms. *Annals of Operations Research*, 25(1-4):75–100, 1990.

[34] J.Q. Hu. Diverse routing in optical mesh networks. *IEEE Transactions on Communications*, 51(3):489–494, 2003.

[35] ILOG. *CPLEX 11.0 User's Manual*. ILOG, 2007.

[36] S. Irnich and G. Desaulniers. Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosier, and M.M. Solomon, editors, *Column Generation*, pages 33–65. Springer, 2005.

[37] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[38] H. Kellerer, R. Mansini, U. Pferschy, and M.G. Speranza. An efficient fully polynomial approximation scheme for the subset-sum problem. *Journal of Computer and System Sciences*, 66(2):349–370, 2003.

[39] L.G. Khachiyan. A polynomial algorithm for linear programming. *Soviet Mathematics Doklady*, 244(5):191–194, 1979.

[40] A.M.C.A. Koster and A. Zymolka. Demand-wise shared protection and multiple failures. In *Proceedings of the International Network Optimization Conference*, 2007.

[41] A.M.C.A. Koster, A. Zymolka, M. Jäger, and R. Hülsermann. Demand-wise shared protection for meshed optical networks. *Journal of Network and Systems Management*, 13(1):35–55, 2005.

[42] P. Kouvelis and G. Yu. *Robust discrete optimization and its applications*. Kluwer Academic, 1997.

[43] L. Lasdon. *Optimization Theory for Large Systems*. MacMillan, 1970.

[44] A. Lisser, A. Ouorou, and J-P. Vial. Capacity planning under uncertain demand in telecommunications networks. Technical report, France Telecom R&D, NT/CNET/6570, 1999.

[45] D.H. Lorenz and A. Orda. QoS routing in networks with uncertain parameters. *IEEE/ACM Transactions on Networking*, 6(6):768–778, 1998.

[46] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41(5):960–981, 1994.

[47] J-F. Maurras and S. Vanier. Network synthesis under survivability constraints. *4OR*, 2(1):53–67, 2004.

[48] M. Minoux. *Mathematical Programming: Theory and Algorithms*. John Wiley & Sons, 1986.

[49] M. Nabe and M.M.H. Miyahara. Analysis and modeling of world wide web traffic for capacity dimensioning of internet access lines. *Performance Evaluation*, 34(4):249–271, 1998.

[50] S. Orlowski. Local and global restoration of node and link failures in telecommunication networks. M.Sc. thesis, Technische Universität Berlin, 2003.

[51] S. Orlowski and M. Pióro. On the complexity of column generation in survivable network design. Technical report, Zuse Institut Berlin and Warsaw University of Technology, 2008.

[52] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly. SNDlib 1.0–Survivable Network Design Library. *Networks*, 55(3):276–286, 2010.

[53] A. Ouorou. Robust capacity assignment in telecommunications. *Computational Management Science*, 3(4):285–305, 2006.

[54] G. Petrou, C. Lemaréchal, and A. Ouorou. Robust network design in telecommunications. In *Proceedings of the International Network Optimization Conference*, 2005.

[55] M. Pióro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan-Kaufmann Publishers, 2004.

[56] M. Pióro, T. Śliwiński, M. Zagożdżon, M. Dzida, and W. Ogryczak. Path generation issues for survivable network design. In *Proceedings of the International Conference on Computational Science and Its Applications, Part II*, 2008.

[57] S. Sen, R.D. Doverspike, and S. Cosares. Network planning with random demand. *Telecommunication Systems*, 3(1):11–30, 1994.

[58] P.D. Seymour. Disjoint paths in graphs. *Discrete Mathematics*, 306(10-11):979–991, 2006.

[59] T. Stidsen, B. Petersen, K.B. Rasmussen, S. Spoorendonk, M. Zachariasen, F. Rambach, and M. Kiese. Optimal routing with single backup path protection. In *Proceedings of the International Network Optimization Conference*, 2007.

[60] J.W. Suurballe. Disjoint paths in a network. *Networks*, 4(2):125–145, 1974.

[61] J.W. Suurballe and R.E. Tarjan. A quick method for finding shortest pairs of disjoint paths. *Networks*, 14(2):325–336, 1984.

[62] R.E. Tarjan. *Data structures and network algorithms*. Society for Industrial and Applied Mathematics, 1983.

[63] K. Thompson, G.J. Miller, and R. Wilder. Wide-area internet traffic patterns and characteristics. *IEEE Network*, 11(6):10–23, 1997.

[64] A. Tomaszewski, M. Pióro, and M. Żotkiewicz. On the complexity of resilient network design. *Networks*, 55(2):108–118, 2010.

[65] R. Wessäly. *Dimensioning Survivable Capacitated NETworks*. PhD thesis, Technische Universität Berlin, 2000.

[66] R. Wessäly, S. Orlowski, A. Zymolka, A.M.C.A. Koster, and C. Gruber. Demand-wise shared protection revisited: A new model for survivable network design. In *Proceedings of the International Network Optimization Conference*, 2005.

[67] M. Żotkiewicz and W. Ben-Ameur. Adding dynamism to robust stable routing. In *Proceedings of the Polish Teletraffic Symposium*, 2009.

[68] M. Żotkiewicz and W. Ben-Ameur. More adaptive robust stable routing. In *Proceedings of the IEEE Global Communications Conference*, 2009.

[69] M. Żotkiewicz and W. Ben-Ameur. Volume oriented routing. In *Proceedings of the International Telecommunications Network Strategy and Planning Symposium*, 2010.

[70] M. Żotkiewicz, W. Ben-Ameur, and M. Pióro. Failure disjoint paths. In *Proceedings of the International Symposium on Combinatorial Optimization*, 2010.

[71] M. Żotkiewicz, W. Ben-Ameur, and M. Pióro. Finding failure-disjoint paths for path diversity protection in communication networks. *IEEE Communications Letters*, 14(8):776–778, 2010.

[72] M. Żotkiewicz, M. Pióro, and A. Tomaszewski. Complexity of resilient network optimization. In *Proceedings of the Polish-German Teletraffic Symposium*, 2008.

[73] M. Żotkiewicz, M. Pióro, and A. Tomaszewski. Complexity of resilient network optimization. *European Transactions on Telecommunications*, 20(7):701–709, 2009.

Appendix A

Résumé de thèse

Dans les grands réseaux de transport, certains éléments du réseau peuvent être responsables du traitement d'importants volumes de trafic. Cela rend ces réseaux vulnérables aux pannes telles que les coupures de câbles. Des mécanismes appropriés pour le recouvrement du trafic doivent être mis en œuvre pour éviter les ruptures de service. Une des meilleures techniques pour protéger les réseaux de transport consiste à prévoir des mécanismes de restauration au niveau de la couche transport elle-même afin que chaque opérateur de transport puisse sécuriser son propre réseau et offrir un service de transport fiable aux autres acteurs tels que les opérateurs IP. D'autres mécanismes de protection pourront alors être déployés aux niveaux supérieurs sans interférences avec la restauration au niveau transport.

Outre les pannes pouvant touchers ses composantes, un réseau doit aussi faire face à l'incertitude de la matrice de trafic qu'on chercher à acheminer dans le réseau. Cette incertitude est une conséquence de la multiplication des applications et services faisant appel au réseau. La mobilité des usagers ainsi que les pannes touchant le réseau contribuent également à cette incertitude.

La thèse se découpe donc en deux parties. Dans la première partie, nous nous intéressons à la complexité des différents mécanismes de sécurisation des réseaux. Dans la seconde partie, nous nous intéressons à l'incertitude de la matrice de trafic et notamment au modèle polyédral.

A.1 Première partie: Complexité

L'un des mécanismes les plus courants pour la sécurisation des réseaux de transport est la protection de chemin (*path protection*, PP). Un exemple est le mécanisme de protection 1+1, appelé *Hot Standby* (HS) dans lequel on définit un chemin primaire utilisé en temps normal et un chemin secondaire disjoint du chemin primaire et portant le même signal. Lorsque le chemin primaire tombe en panne, on reçoit les données à travers le chemin secondaire. Le mécanisme 1+1 est efficace en termes de temps de restauration, mais coûteux en termes de capacités. En effet, les capacités de protection ne sont pas partagées entre les différents chemins primaires et doivent donc être au moins égales aux capacités utilisées en temps normal. Certaines variantes de la protection de chemin (comme la protection n : m et la diversité des chemins) peuvent réduire le coût de la protection d'une manière limitée.

Pour réduire les coûts d'une manière plus sensible, des mécanismes actifs de restauration de chemins (PR) sont envisagés. Bien que PR est assez complexe à mettre en œuvre et peut ralentir les temps de restauration, il permet de mutualiser les ressources et de réduire les coûts. La technique PR consiste à prévoir des chemins de routage utilisés en régime normal (absence de pannes) et d'autres qui sont utilisés en cas de panne. Lorsqu'une panne se produit, les connexions qui utilisaient des chemins affectés par la panne sont reroutés sur des chemins valides. On considère plusieurs variantes de PR:

- Les capacités utilisées pour la sécurisation peuvent provenir des capacités libérées par les chemins qui étaient utilisés en régime nominal et qui ne le sont plus à cause d'une panne donnée. On est alors dans le cas dit avec *Stub-Release* (SR).

- Les capacités utilisées pour la protection sont disjointes de celles utilisées en régime normal. On est alors dans le cas sans *Stub-Release* (nSR).

Dans les deux cas, une optimisation fine est nécessaire pour mutualiser les ressources au

mieux en tenant compte des possibilités techniques offertes pour chaque variante. Dans les deux cas, il est important que la capacité de protection soit partagée entre les flux restaurés suite à des pannes différentes.

La protection peut:

- Soit dépendre de la panne (*failure-dependent*, FD). Les chemins de routage utilisés pour restaurer un flux primaire donné peuvent alors être différents en cas de pannes différentes.

- Soit être complètement indépendante de la panne (*failure-independant*, FI). C'est-à-dire qu'il faut définir des chemins de routage qui soient valides quelle que soit la panne qui touche les chemins utilisés en régime normal.

Dans la première partie de la thèse, nous analysons la complexité de différents mécanismes de sécurisation des réseaux. La complexité des problèmes d'optimisation sous-jacents dépend également des scénarios de panne considérés. Dans certains cas, on considère uniquement des pannes simples et non-simultanées de quelques éléments du réseau. Dans d'autres cas, on peut avoir des pannes multiples où plusieurs composantes du réseau sont en panne. Le fait de considérer des pannes multiples rend les problèmes d'optimisation plus difficiles.

Il est important de connaître la complexité des problèmes d'optimisation liés à la conception des réseaux résilients. En effet, connaître cette complexité aide à choisir les bonnes méthodes de résolution pour ces problèmes

Notre principale contribution dans ce cadre est le résultat suivant.

La plupart des problèmes de réseaux résilients sont $\mathcal{NP}$-difficiles, même lorsque l'on autorise la bifurcation des flux.

Les problèmes considérés sont en réalité des variantes des problèmes de multi-flots liés aux mécanismes PP et PR. Nous avons établis que tous les problèmes sont $\mathcal{NP}$-difficiles lorsque des pannes multiples peuvent se produire simultanément, même lorsqu'on autorise

de partager une demande de trafic sur plusieurs chemins (routage fractionnaire). Les preuves de complexité sont non-triviales et sont présentées dans le manuscrit.

Dans la littérature, plusieurs auteurs se sont intéressés à la complexité des problèmes de conception de réseaux résilients. En réalité, la plupart de ces résultats ne traitent pas de la complexité des problèmes d'optimisation de base, mais sont plutôt consacrés à la complexité du problème dit de pricing, requis pour la génération de colonnes dans l'écriture en formulation chemins du modèle linéaire correspondant. Malgré l'importance pratique de ces résultats, ils ne permettent pas en général d'établir la complexité des problèmes de base, à laquelle est dédiée la première partie de la thèse.

Dans la littérature on retrouve également l'étude du cas d'une seule panne et le calcul de bornes inférieures de bonne qualité pour notre problème.

Le premier problème que nous analysons est la protection de chemin (*path protection*, PP). Il s'agit de mécanismes passifs dans le sens où aucun flux n'est restauré, et donc le flux résiduel dans un état de panne doit être suffisant pour satisfaire les demandes. Dans la thèse, nous présentons le mécanisme de *hot-standby* PP (HS), ainsi que son extension appelée *path diversity* (PD). Avec HS, c'est à dire la protection 1+1, chaque demande est envoyée sur un seul chemin protégé par un chemin unique dédié. Les deux chemins sont disjoints en termes de pannes, c'est-à-dire que les deux chemins ne sont jamais en panne simultanément.

L'inconvénient majeur de HS est qu'il nécessite au moins deux fois la capacité d'un réseau non protégé. Par conséquent, on s'intéresse à de meilleurs mécanismes de PP moins gourmands en capacités. Un des mécanisme de PP qui est sans doute parmi les plus économiques en terme de capacité est la diversité des chemin: *path diversity* (PD). PD protège le trafic contre des pannes de composants du réseau, par sur-dimensionnement, c'est-à-dire en acheminant plus de trafic que spécifié par la valeur de la demande dans l'état normal, et en veillant à ce que au moins la quantité prévue de volume survive dans chaque scénario de défaillance,

sans reroutage. Le concept de la PD a été étudié dans la littérature sous des noms différents, on retrouve par exemple les termes de diversification et de protection partagée en fonction de la demande. La complexité des problèmes de HS-PP et PD est étudiée dans le début de la première partie.

La suite de la première partie est consacrée à l'étude de la complexité de la restauration de chemin (PR). Il s'agit de mécanismes actifs, dans le sens où ils restaurent le fonctionnement des flux détruits par une panne. Certes, pour garantir une bonne restauration, des capacités de protection supplémentaires sont nécessaires par rapport à un réseau conçu uniquement pour fonctionner sans panne. Mais le coût des capacités supplémentaires est généralement beaucoup plus faible pour PR que pour les PP, puisque, par définition, les capacités de protection PR sont partagées par différentes demandes et pour des pannes différentes. Dans un premier temps, nous supposons que les flux défaillants sont restaurés de façon indépendante de la panne (FI). Le cas de la restauration dépendant des pannes (FD) est traité dans un deuxième temps.

Les mécanismes PR sont étudiés sous deux hypothèses différentes: avec ou sans stub-release. Avec Stub-Release (SR), la capacité sur les parties survivantes (stubs) d'un chemin de routage qui a été coupé par une panne peut être réutilisée pour la sécurisation. Sans stub-release (nSR), cette capacité est exclusivement réservée pour le routage en absence de panne, et ne peuvent pas être réutilisés en cas de défaillance. Notez que nSR est typique pour les couches de transport, par exemple, pour les réseaux optiques, ou les réseaux SONET.

Selon les paramètres considérés, on obtient ainsi quatre problèmes différents de restauration de chemins, à savoir: FI-nSR, FI-SR, FD-nSR, et FD-SR. La variante non-bifurquée (mono-routage) du problème de FI-SR, suppose que chaque demande est routée sur un seul chemin en absence de panne. A chaque panne affectant le chemin primaire choisi, la demande

est déplacée vers un autre chemin, sans fractionnement de la demande. La complexité algorithmique de tous ces cas est discutée dans la thèse. On retrouve également des résultats sur la $\mathcal{NP}$-complétude de différentes variantes du problème FI-nSR.

Les différentes variantes ont été étudiées. Nous proposons des formulations en programmes linéaires, ou en programmes linéaires à variables mixtes (MIP). Dans la plupart des cas, la formulation est une formulation non-compacte du type arc-chemin. La formulation compacte sommet-arc est utilisée uniquement dans deux cas de routage bifurqué à savoir: PP-PD (protection de chemin avec diversité), et la FI-nSR PR (restauration de chemin indépendante des pannes sans stub-release).

Certains des résultats de cette première partie de la thèse ont été présentés au Polish-German Teletraffic Symposium (PGTS), Berlin, Allemand, en Septembre 2008, et par la suite dans European Transactions on Telecommunications en 2009 et le journal Networks en 2010. Enfin, certains résultats ont été présentés au Colloque International Symposium on Combinatorial Optimization (ISCO), Hammamet, Tunisie en Mars 2010, et publiés dans IEEE Communications Letters en 2010.

A.2 Deuxième partie: Modèle polyédral

Dans la deuxième partie de la thèse, nous nous intéressons à un autre type d'incertitude: l'incertitude de la demande de trafic. Nous nous focalisons sur le modèle polyédral.

Les réseaux de télécommunication modernes doivent acheminer du trafic généré par une variété d'applications différentes, et fournir des services à un grand nombre d'utilisateurs. Cela rend difficile la prédiction du trafic. En outre, l'introduction de nouveaux services et la mobilité des clients rendent la tâche encore plus difficile.

Certains modèles ont été proposés dans le passé pour faire face à l'incertitude du trafic. La première approche consiste à construire une matrice de trafic fondée sur le pire cas pour

chaque composante du trafic. Le routage est alors calculé sur la base de cette matrice. Cette approche a l'avantage d'être simple, mais elle peut fournir des solutions onéreuses. En effet, elle n'autorise pas le partage des ressources par des composantes de trafic qui n'atteignent pas le pire cas simultanément.

La deuxième approche est basée sur une modélisation probabiliste des variations de trafic. Après avoir spécifié un modèle probabiliste pour la prévision du trafic, on peut chercher le routage qui optimise un certain critère probabiliste: le débit moyen, le retard moyen, la probabilité de blocage, etc. La solution obtenue de cette manière est bonne en moyenne, mais peut être très mauvaise dans certains cas. En outre, cette méthode nécessite la connaissance des modèles de probabilité, qui sont généralement difficiles à obtenir. Ce type d'approche est généralement appelé programmation stochastique. Une approche classique différente, l'optimisation robuste, prend en compte un nombre fini de situations possibles. Dans ce cas, on cherche une solution qui prend en charge toutes les situations envisagées. Dans le contexte des réseaux, il s'agit alors de déterminer un schéma de routage, de telle sorte que chacune des matrices de trafic appartenant à un ensemble fini donné de matrices de trafic peut être satisfaite par le réseau. Un autre modèle robuste suppose que le trafic sortant de chaque nœud est borné (la limitation sur le trafic entrant peut également être prise en compte). La matrice de trafic peut être alors n'importe quelle matrice satisfaisant ces contraintes. Ce modèle avec incertitude est appelé *hose model*. Plusieurs problèmes de conception de réseau basé sur ce modèle ont reçu une attention considérable dans la littérature.

Certains auteurs proposent un modèle différent, en supposant que tous les nœuds ne peuvent pas produire la quantité maximale de trafic simultanément. En fait, ce modèle, ainsi que le *hose model* présenté ci-dessus, sont des cas particuliers d'un modèle polyédral plus général. Le modèle suppose que chaque matrice de trafic possible appartient à un polytope. Dans la littérature un algorithme en temps polynomial a été proposé pour calculer un plan de

routage qui résout ce problème. Le routage est robuste et stable (robuste, car il est compatible avec toutes les matrices, et stable, car il ne change pas lorsque des changements de matrice se produisent). Bien que le modèle ait été défini pour les matrices de trafic appartenant à un polytope, il peut également traiter avec succès les problèmes avec des matrices appartenant à une union de polytopes.

Une approche différente face à l'incertitude consiste à permettre à un réseau de changer de routage de façon dynamique, quand il y a un changement significatif en termes de matrice de trafic. Ce problème a été intensément étudié dans le cadre des réseaux commutés. Des règles différentes peuvent être appliquées pour connecter des appels en fonction de la situation actuelle, comme par exemple, le routage dynamique alternatif, le routage séquentiel, etc. Bien que ce type de routage présente de nombreux avantages, il est généralement difficile à mettre en œuvre. De plus il n'est pas optimal car les règles utilisées sont fixées à l'avance. Cependant, il peut être utilisé comme point de référence (borne inférieure) pour évaluer les approches présentées plus tôt. Dernièrement, il a été prouvé que le coût optimal d'un réseau fondé sur les modèles de routage robustes et stables (fractionnel ou intégral) peut être un facteur de $(\log n)$ plus grand que le coût requis pour utiliser le routage dynamique.

Dans le passé, la question suivante a été posée: étant donné un polytope de demande de trafic avec incertitude, est-il en théorie facile de calculer un routage complètement dynamique (système de routage qui dépend du trafic de la matrice courante)? Récemment, il a été prouvé que ce problème est en général difficile: il est co-$\mathcal{NP}$-difficile de décider si un réseau donné avec des capacités connue est capable de transporter chaque matrice de trafic dans le polytope, lorsque le routage est dynamique. En outre, pour des raisons techniques, il est très difficile de mettre en œuvre ce type de routage. Par conséquent, il est intéressant de considérer un intermédiaire entre le routage stable et robuste et le routage entièrement dynamique. Au lieu de calculer une solution robuste, il est possible de partitionner le polytope de la demande

de trafic en plusieurs sous-ensembles, et de calculer un routage robuste pour chacun d'eux. On peut par exemple faire ce partitionnement à l'aide d'un hyperplan. On suppose que la direction d'un hyperplan divisant le polytope des matrices de trafic est connue, mais sa position est sujette à optimisation. Par exemple, une quantité de trafic généré entre une paire de nœuds, ou une quantité de trafic généré par un nœud spécifique peuvent être pris comme une direction de l'hyperplan. Dans une telle situation, une position d'un seuil qui déclenche des changements dans le routage est optimisée. En d'autres termes, deux stratégies de routage sont spécifiées, la première est utilisée si le volume du trafic, qui a été défini comme une direction de l'hyperplan, est inférieur au seuil, sinon la seconde stratégie de routage est utilisée. Une autre possibilité est de considérer le temps comme une direction de l'hyperplan. Dans ce cas, le temps est représenté par une dimension supplémentaire de l'ensemble de l'incertitude, et le moment où le routage doit être changé est optimisé.

En plus du routage robuste stable et le routage dynamique, nous étudions également le routage sans partage. Dans ce mode de routage, les ressources sont réservées par demandes et aucune mutualisation entre les demandes de trafic n'est tolérée.

Parmi les différents modes de routage et les différents problèmes sous-jacents, nous avons identifié lesquels de ces problèmes étaient polynomiaux, et comment ils peuvent être résolus avec succès.

Nous distinguons par exemple le cas où les capacités sont réservées une seule fois pour l'ensemble du polytope de trafic (i.e., même si le routage change, les capacités réservées ne changent pas). Nous étudions également le cas où les capacités peuvent être re-réservées à chaque modification de routage. Toutes les variantes sont étudiées et comparées.

Le problème majeur avec ce type d'approche est que les routages peuvent différer considérablement entre les sous-ensembles obtenus en partitionnant le polytope, et la mise en œuvre de ces changements peut alors entraîner des pertes graves de trafic (en raison de

l'instabilité). Il est donc nécessaire de trouver une stratégie rendant les changements de routage moins abrupts. Pour cela, dans la deuxième partie, la thèse suivante est vérifiée.

Il existe des mécanismes de routage qui sont décentralisés et dont la mise en œuvre est simple capables de faire face à la demande de trafic polyédrale.

Nous proposons un nouveau mécanisme de routage, appelé routage guidé par le volume (et ses variantes polynomiales: routage guidé par volume simplifié et routage guidé par volume généralisé). Ce mécanisme est l'une de nos contributions majeures. Il tire parti de la simplicité du routage stable robuste et de l'efficacité du routage dynamique. En outre, il n'implique pas de changements brusques des flux dans le réseau.

Ce nouveau mécanisme de routage implique une division d'un polytope de la demande de trafic. Toutefois, dans ce cas, seule une partie du débit qui dépasse un seuil est acheminé à l'aide d'un ensemble de chemins différents. Trouver une stratégie optimale est généralement $\mathcal{NP}$-difficile. Cependant, nous montrons qu'une autre variante du routage guidé par le volume peut être traitée en temps polynomial. Nous proposons également une généralisation du routage guidé par le volume qui est aussi facile à calculer.

L'ensemble des modes de routage et toutes les variantes ont été implémentés. Les résultats numériques montrent l'applicabilité des algorithmes.

Les résultats concernant les stratégies de partitionnement étudiées dans la thèse ont d'abord été présentées par l'auteur lors du 6th Polish Teletraffic Symposium, Łódź, Pologne, en Septembre 2009, puis à IEEE Global Communications Conference (GLOBECOM), Honolulu, États-Unis, en décembre 2009. Ils sont également publiés dans un article dans le journal International Transactions in Operational Research. Les résultats concernant les stratégies guidées par volume ont été présentées lors du 14th International Telecommunications Network Strategy and Planning Symposium, Varsovie, Pologne, en Septembre 2010.