Shortest Path Routing
Modelling, Infeasibility and Polyhedra

Mikael Call
In memory of my father.
Abstract

The Internet is constantly growing but the available resources, i.e. bandwidth, are limited. Using bandwidth efficiently to provide high quality of service to users is referred to as traffic engineering. This is of utmost importance. Traffic in IP networks is commonly routed along shortest paths with respect to auxiliary link weights, e.g. using the OSPF or IS-IS protocol. Here, shortest path routing is indirectly controlled via the link weights only, and it is therefore crucial to have a profound understanding of the shortest path routing mechanism to solve traffic engineering problems in IP networks. The theoretical aspects of such problems have received little attention.

Traffic engineering in IP networks leads to very difficult optimization problems and the key element in exact methods for such problems is an inverse shortest path routing problem. It is used to answer the fundamental question of whether there exist link weights that reproduce a set of tentative paths. Path sets that cannot be obtained correspond to routing conflicts. Valid inequalities are instrumental to prohibit such routing conflicts.

We analyze the inverse shortest path routing problem thoroughly. We show that the problem is NP-complete, contrary to what is claimed in the literature, and propose a stronger relaxation. Its Farkas system is used to develop a novel and compact formulation based on cycle bases, and to classify and characterize minimal infeasible subsystems. Valid inequalities that prevent routing conflicts are derived and separation algorithms are developed for such inequalities.

We also consider several approaches to modelling traffic engineering problems, especially Dantzig–Wolfe reformulations and extended formulations. We give characterizations of facets for some relaxations of traffic engineering problems.

Our results contribute to the theoretical understanding, modelling and solution of problems related to traffic engineering in IP networks.
**Populärvetenskaplig sammanfattning**


Vår analys resulterar i en matematisk karaktärisering av ruttningskonflikter. Denna leder i sin tur till metoder för att hitta samt förbjudas sådana ruttningskonflikter. Dessa metoder kan sedan integreras med andra metoder som vanligtvis används för att lösa trafikplaneringsproblem utan den komplicerade aspekten att ruttnings- och kortaste vägar.
Acknowledgments

I would like to start by expressing my gratitude towards my supervisor Professor Kaj Holmberg for his support and encouragement. I am very thankful for the opportunity to conduct research studies in combinatorial optimization, and have greatly enjoyed the freedom to investigate my own research ideas.

The collaboration with Professor Andreas Bley and Daniel Karch at TU Berlin have meant a lot to me. Andreas has given me feedback and insights about IP routing in practice when acting as the opponent on my Licentiate thesis and during my short visits at TU Berlin. Daniel has been an invaluable source of discussion and motivation. He also provided me with a nice computational framework in SCIP for experimenting. I believe that our continued collaboration will be very fruitful.

I also want to thank my present and former colleagues. My roommate Åsa Holm has been a great asset, socially and scientifically. I enjoy our discussions and you make me realize when I am wrong — not only when proofreading the hardest parts of this thesis. Professor Torbjörn Larsson have contributed to my understanding of various topics in optimization by many valuable discussions, thank you.

Finally, I want to thank my family and friends for all your love and support. Especially my wife Anna, who has stood by me through the entire process, and given me the best gift imaginable in our daughter Alice.

Linköping, October 31, 2012

Mikael Call
II Inverse Shortest Path Routing 41

4 Complexity of Realizability 43
  4.1 Background ................................................. 43
  4.2 Models for Realizability and Compatibility ............... 45
    4.2.1 Incompatible and Unrealizable SP-graphs .......... 47
  4.3 Complexity of ISPR Problems ............................ 49
  4.4 Conclusion ............................................... 55

5 Relaxations of Realizability 57
  5.1 Background ............................................... 57
    5.1.1 Inverse Shortest Path Routing Problems .......... 58
  5.2 Partial Realizability .................................... 60
    5.2.1 Generalized Shortest Path Graphs ................. 62
    5.2.2 Relation Between ISPR Formulations ............... 63
  5.3 Valid Inequalities from Partial Realizability .......... 64

6 Classes of Infeasible Structures 69
  6.1 Problem Formulation ..................................... 69
    6.1.1 The ISPR Compatibility Model .................... 70
    6.1.2 The ISPR Partial Realizability Model ............ 71
  6.2 Classes of Infeasible Routing Pattern Structures ....... 72
    6.2.1 The General Infeasible Routing Pattern Structure 73
    6.2.2 The Binary, Unitary and Simplicial Structures .... 77
  6.3 Extreme Rays and Generators ........................... 79
    6.3.1 Representation of Polyhedral Cones ............... 79
    6.3.2 Extreme Rays and Generators of the Closure of \( \Theta \) 79
    6.3.3 Irreducible Solutions in the Closure of \( \Theta \) ... 82
  6.4 The Hierarchy of Infeasible Structures .................. 83

7 The Simplicial Structure 91
  7.1 The Valid Cycle Structure ................................ 91
    7.1.1 Valid Cycles with a Single SP-graph .............. 92
    7.1.2 Saturating Solutions with Two SP-Graphs .......... 93
    7.1.3 Non-Saturating Solutions with Two SP-Graphs ...... 94
    7.1.4 Algorithms to Find Generalized Valid Cycles ...... 96
  7.2 A Characterization of the Simplicial Structure ........... 97
    7.2.1 Generating Simplicial Cycle Families .............. 98
  7.3 Dependency Graphs ....................................... 102
  7.4 Characterization of Irreducible Generators ............... 106

8 Cycle Basis Formulations 111
  8.1 Modelling with Cycle Bases .............................. 112
    8.1.1 Oriented Circuits, Circulations and Cycle Bases .... 112
    8.1.2 Fundamental Cycle Bases ............................. 113
    8.1.3 Modelling Circulations with Cycle Bases ........... 114
    8.1.4 The Minimum Cost Circulating Flow Problem ...... 114
12 ECMP Modelling and Relaxations 201
  12.1 Modelling Splitting at a Single Node 202
     12.1.1 Stronger Single Node Formulations 204
     12.1.2 The Single Node Splitting Polytope 207
  12.2 An ECMP Relaxation of an IP Network Problem 212
     12.2.1 A Discretization Approach via ECMP Flow Patterns 213
     12.2.2 Solving the Dantzig–Wolfe Reformulation 215
     12.2.3 Related Dantzig–Wolfe Reformulations 216
  12.3 Solving ECMP Splitting Problems 217
  12.4 Solving the Acyclic Ingraph Problem 226
     12.4.1 The Acyclic Ingraph Polytope 227
     12.4.2 Heuristics for the Acyclic Ingraph Problem 236

13 Separation of SPR Inequalities and Computational Aspects 237
  13.1 Heuristic Separation of Routing Conflicts 238
  13.2 Exact Separation of Routing Conflicts 240
  13.3 Separation of Valid Cycle Inequalities 243
     13.3.1 Directed Cycle and Subpath Inconsistency Separation 244
     13.3.2 Valid Cycle Separation 245
     13.3.3 Valid Cycle Separation in the ECMP Case 247
  13.4 Computational Aspects of SPR Problems 249
     13.4.1 Computation Scheme Components 250

14 Future Research 253

Bibliography 255
1

Introduction

The primary concern of this thesis is the shortest path routing aspect of optimization problems in IP networks. All activities on the Internet require that data packets are sent from a source to a destination. The determination of the path to use is called routing. Since the majority of the traffic on the Internet is directed by shortest path routing, it is important to study this aspect of optimization problems in IP networks. In this thesis, we approach issues related to shortest path routing by mathematical programming.

In fact, the use of the shortest path routing principle is not restricted to routing in IP networks. We use shortest path routing as an umbrella term for decision making processes where the routes used between origins and destinations are determined as shortest paths w.r.t. some arc cost function. A shortest path routing problem is an optimization problem under the presumption that shortest path routing is used.

These problems are naturally described as bilevel programming problems, i.e. two stage optimization problems. In the first stage, a leader decides upon the arc costs in a digraph. Then, in the second stage, the followers, usually corresponding to origin–destination pairs, determine their routes by shortest path routing. Knowing that followers travel along shortest paths, the leader’s objective is to optimize (in some sense) the resulting travelling pattern, i.e. the induced flow. This class of bilevel problems is referred to as bilevel shortest path problems.

There are several applications that require the solution of a bilevel shortest path problem. For example, there are many applications related to traffic planning. In these problems, it is assumed that travellers use shortest paths w.r.t. some generalized arc cost function that can also take factors like e.g. travel time into account. Another class of applications were mentioned briefly above, optimization problems in IP networks. Routing in IP networks is conducted by shortest path routing protocols, such as OSPF [177] or IS-IS [76], where all routes are determined as shortest paths w.r.t. some auxiliary arc cost function. This is also a kind of traffic planning problem, but the travellers correspond to data packets.

Our main objective is to develop a unified theory of shortest path routing. We strive to
provide theoretical and practical insights that can be used to develop solution methods for shortest path routing problems. The main focus is on methods for optimization problems in IP networks based on cutting planes.

Our approach is heavily based on the following fundamental question: which sets of paths can simultaneously, i.e. w.r.t. the same arc cost function, be realized as shortest paths? A set of paths that can be realized as shortest paths is called a feasible routing pattern, while an unrealizable set of paths is said to induce a routing conflict. We analyze this fundamental question thoroughly via an inverse shortest path routing problem. Based on this inverse problem, we can characterize routing conflicts and analyze the polytopes of simultaneously feasible routing patterns. In particular, valid inequalities are derived together with separation algorithms. To illustrate the concept of a feasible routing pattern and a routing conflict we provide a small example below.

A feature of our unified approach is that it is independent of other complicating problem aspects, e.g. how the arc cost function is determined. This implies that the valid inequalities mentioned above are effective for any bilevel shortest path problem. However, the generality of the approach also becomes its main weakness. In a sense, we focus on feasible routing patterns rather than feasible flow patterns. This is one reason of why the valid inequalities are not necessarily efficient for all bilevel shortest path problems. They do turn out to be efficient for optimization problems in IP networks which is our main concern.

---

**Example 1.1**

Let $G = (N,A)$ be a digraph with four nodes and consider two desired shortest paths $1 - 2 - 4$ and $2 - 3 - 4$. The graph and the associated routing pattern is illustrated in Figure 1.1.

First assume that no path other than $1 - 2 - 4$ is allowed to be a shortest path from node 1 to node 4. In this case, the paths $1 - 2 - 4$ and $2 - 3 - 4$ form a routing conflict since the path $1 - 2 - 3 - 4$ is also an induced shortest $(1,4)$-path. The latter fact follows by Bellman’s principle of optimality applied to shortest paths.

Instead assume that it is allowed that other paths than $1 - 2 - 4$ are shortest paths from 1 to 4. In this case, the paths $1 - 2 - 4$ and $2 - 3 - 4$ no longer form a routing conflict. This can be verified by the arc cost function, $w : A \to \mathbb{Q}$, where

$$w_{12} = 1, w_{23} = 1, w_{24} = 2, w_{34} = 1,$$  \hspace{1cm} (1.1)

since it yields the above paths as shortest paths. Observe that the path $1 - 2 - 3 - 4$ is also a shortest $(1,4)$-path w.r.t. $w$.

**Figure 1.1:** A routing pattern consisting of the two paths, $1 - 2 - 4$ and $2 - 3 - 4$, drawn with solid and dashed arcs, respectively. The feasibility of the routing pattern depends on whether the path $1 - 2 - 3 - 4$ is allowed to be a shortest path or not.
1.1 Thesis Outline

This thesis is divided into three parts. Part I is introductory in nature. It contains introductions to bilevel shortest path problems, optimization problems in IP networks and the inverse shortest path routing problem. Part II consists of a deeper analysis of inverse shortest path routing problems, i.e. the client problem. In particular, several mathematical formulations of this class of problems are given. The solutions to their Farkas’ systems are used to characterize routing conflicts. The focus in Part III is on the solution of shortest path routing problems, i.e. the master problem, by cutting plane approaches. In particular, some polytopes related to feasible routing patterns are analyzed and valid inequalities are derived based on the characterization of routing conflicts in Part II. Separation algorithms are also developed, including very efficient algorithms for some subclasses of inequalities.

Chapter 2, 3 and 9 mainly serve to give an overview of related work and the approach we use. In the remaining chapters, we report on original research. A very brief outline is given.

Chapter 1 The current chapter contains a brief introduction, this outline and a summary of our main contributions.

Part I — Background

Chapter 2 We introduce bilevel shortest path problems. In this framework, followers travel from some origin to some destination along shortest paths w.r.t. arc costs induced by a leader’s decision. Taking this into account, the leader then optimizes w.r.t. some objective. The ordinary shortest path problem and its optimality conditions are fundamental. We properly introduce them. The optimality conditions then allows us to state the inverse shortest path routing problem which will be a key element in our approach. To make bilevel shortest path problems more concrete, we consider two applications that fit well into the framework: routing in IP networks and tariff optimization. Finally, we briefly outline two common solution approaches for bilevel shortest path problems.

Chapter 3 In Part II, the focus is on the inverse shortest path routing problem. Here, we derive the commonly used formulation for this problem based on the optimality conditions for the ordinary shortest path problem.

Part II — Inverse Shortest Path Routing

Chapter 4 This is the first chapter of five dedicated to a thorough analysis of the inverse shortest path routing problem. We state this problem in its full generality and refer to it as the realizability problem. The problem is to decide whether there exist link weights that reproduce a set of tentative paths, without also introducing some undesired paths as shortest paths. A main result is that this problem is NP-complete.

Chapter 5 Since the realizability problem is NP-complete, a relaxation referred to as compatibility is in practice solved instead. Here, we propose a new relaxation. We refer to the relaxation as partial realizability and compare it to realizability and compatibility. We show that our relaxation is stronger than compatibility, and has
additional exploitable structure that will be used throughout the thesis. To demonstrate one descriptive advantage induced by the additional structure, we give some examples of valid inequalities that are easy to derive using partial realizability but require ad hoc arguments if compatibility is considered.

Chapter 6 To derive valid inequalities for bilevel shortest path problems, it is very fruitful to analyze the infeasibility of inverse shortest path routing problems. Our analysis of the Farkas system begins in this chapter. It results in a combinatorial characterization of five classes of infeasible structures. We show that the classes are exhaustive and strictly nested.

Chapter 7 Our analysis of infeasibility is continued. We investigate a comprehensible class of infeasible structures induced by what we refer to as simplicial solutions. This class includes the so called valid cycles that involve at most two SP-graphs. We contribute to the understanding of simplicial solutions by showing how they arise from graph embeddings. In particular, the dual graph encodes a dependency relation between cycles. This is used to derive a main result: a characterization of simplicial structures that correspond to extremal solutions and irreducible solutions. In terms of the inverse shortest path problem, an irreducible solution corresponds to a minimal infeasible subsystem, or a minimal routing conflict.

Chapter 8 We exploit the structure of the Farkas system of the inverse shortest path problem to derive a novel cycle basis formulation. Our model is similar to models based on cycle enumeration and contains only few constraints. Its advantage is that it only contains a polynomial number of variables. The formulation emphasizes the circulation structure, which we use to get practical and theoretical insights that translate to the original partial realizability model and its Farkas system, e.g. under very general conditions a subset of constraints are redundant. The Farkas system of our cycle basis yields a path based formulation equivalent to the partial realizability formulation in Chapter 5. In contrast to other path based formulations in the literature, it has a polynomial number of paths only.

Part III — A Unified Framework for Routing in IP Networks

Chapter 9 This is the first chapter of five that concern problems related to the master problem. It serves as an introduction to shortest path routing problems in the context of telecommunication applications, i.e. traffic engineering problems. We present a core model without complicating side constraints, and give a brief overview of some common modifications of the core problem, e.g. alternative objective functions, link and capacity models, etc.

Chapter 10 The analysis in Chapter 6-7 leads to a description of infeasible routing patterns. Here, we translate this into a combinatorial description of valid inequalities, which results in integer linear formulations of polytopes related to feasible routing patterns. We give some necessary and sufficient conditions for these valid inequalities to be non-dominated, which extends and generalizes the characterization of irreducibility in Chapter 7. Finally, we express the valid inequalities via conflict hypergraphs to address the connection to independence/transitive systems. This
Chapter 11 We propose some Dantzig–Wolfe reformulations for problems related to traffic engineering in IP networks where traffic must be routed along unique shortest paths. This leads to branch-and-price or branch-and-cut-and-price methods. The problem structure results in a small Dantzig–Wolfe master and easy subproblems. We discuss branching rules that preserve the structure of the pricing problem, and how to deal with cutting. We also provide a general method for translating a cut in the original space into a stronger cut in the extended space.

Chapter 12 When shortest paths are not required to be unique in the traffic engineering context, it is usually assumed that traffic is split according to the so called ECMP principle. We consider the modelling aspect of the ECMP principle. The convex hull of the ECMP splitting polytope for a single node is described via an extended formulation. A problem with ECMP splitting similar to a problem in Chapter 11 is approached via Dantzig–Wolfe reformulation. The resulting pricing problem yields a very large LP. We propose a reformulation of the problem that leads to a dual that resembles the dual of the shortest path problem, but has an exponential number of constraints. Using the shortest path analogy, we develop efficient dynamic programming algorithms to solve the pricing problem, also for quite general branching rules. Finally, the acyclic ingraph problem related to routing with ECMP splitting is considered. Several classes of facets are derived for the associated polytope.

Chapter 13 Solution methods based on cutting planes require repeated solution of separation problems. We derive a heuristic and an exact method for separating fractional solutions from valid inequalities based on general routing conflicts. The most important class of routing conflicts is associated with valid cycles. We develop several efficient separation algorithms for inequalities based on routing conflicts related to valid cycles. Finally, a computational scheme for traffic engineering problems is outlined, and we discuss how the pieces in Part III relates to each other and how they are intended to be used in computations.

Chapter 14 We conclude by giving some directions for future research.

1.2 Contributions

The theoretical aspects of traffic engineering problems in IP networks have received little attention. We make some contributions in this direction.

We divide the main contributions presented here into two categories; Contributions in Part II relates to the client problem, i.e. the inverse shortest path routing problem and contributions in Part III relates to the master problem, e.g. a traffic engineering problem in an IP network.
1.2.1 Contributions Related to the Client Problem

The key element in exact methods for shortest path routing problems is an inverse shortest path routing problem. We contribute to the understanding of the shortest path routing mechanism by analyzing this problem. Our main contributions are:

1. We introduce the realizability problem. Earlier, the compatibility version of the inverse shortest path routing problem was used to decide whether there exist arc costs that reproduce a set of tentative paths as shortest paths when some arcs are not allowed to be in some shortest paths. It was believed that compatibility is necessary and sufficient for realizability. We show that it is only necessary and that deciding if a set of tentative paths are realizable is NP-complete.

   We develop the partial realizability problem to obtain a stronger necessary condition for realizability. Besides being a stronger relaxation than compatibility, the partial realizability problem is also more structured. We relate the three problems and give some sufficient conditions for their equivalence.

   These results are published in [74].

2. We thoroughly analyze the inverse shortest path routing problem. The structure of the Farkas system of the partial realizability problem allows us to characterize its solutions. This leads to both a combinatorial description of solutions and a classification of routing conflicts.

   We derive properties of the solutions to the Farkas system of theoretical and practical significance; for instance, all constraints are under very general conditions binding which can be exploited to remove constraints and reduce degeneracy. We also give combinatorial characterizations of the extremal solutions to the Farkas system and of irreducible routing conflicts, i.e. minimal infeasible subsystems to the inverse shortest path routing problem.

3. We propose a novel modelling approach for the partial realizability problem based on fundamental cycle bases. We utilize the multicommodity structure of the partial realizability problem. This yields a compact model. Our cycle basis formulation is similar to a Dantzig–Wolfe reformulation, or a cycle enumeration approach, and results only in aggregated capacity constraints and variable bounds. In contrast to models based on cycle enumeration, it has only a polynomial number of variables. Via duality, this also yields the first path based formulation of inverse shortest path problems with a polynomial number of paths.

   A related formulation is published in [75].

1.2.2 Contributions Related to the Master Problem

Our contributions to the traffic engineering problem are of polyhedral character or related to modelling. The main contributions are:

1. The combinatorial description of infeasibility from Part II is translated to a combinatorial description of valid inequalities that prohibit routing conflicts. We develop
1.2 Contributions

separation algorithms for these inequalities. For the practically most important class of routing conflicts, i.e., valid cycles, we are able to derive very efficient separation algorithms. Further, the characterization of irreducible routing conflicts leads to criteria for domination among valid inequalities. We describe all necessary routing conflicts in terms of conflict hypergraphs to facilitate their structure and the connection to independence systems and transitive systems.

2. We consider Dantzig–Wolfe reformulations of some traffic engineering problems and develop efficient branch-and-price schemes. We propose branching rules that either preserves the structure of the pricing problem or can be handled efficiently. For the so called ECMP case, we develop an exponential reformulation of the pricing problem. This reformulation can be solved efficiently by dynamic programming, and can incorporate very general branching rules.

3. We consider systems of inequalities arising in traffic engineering problems and propose alternative modelling approaches. For the single node splitting polytope arising in the ECMP case, we develop an extended formulation that projects to the convex hull of the original polytope. For the acyclic ingraph polytope, also relevant in the ECMP case, we derive several classes of facets and some efficient separation algorithms.

In conclusion, our results contribute to the theoretical understanding, modelling and solution of problems related to traffic engineering in IP networks.
Part I

Background
Bilevel Programming is a well established paradigm in mathematical programming. A bilevel program is characterized by a two stage decision process where first a leader makes a decision, and then a set of followers react upon the leaders decision.

In our setting, the leader must (possibly implicitly) decide upon arc costs in a strongly connected digraph. Every follower is associated with an origin–destination (OD) pair in this graph as well as a demand function. Being rational, followers travel from their associated origin to their destination along the shortest path(s) w.r.t. the arc costs induced by the leaders decision, i.e. an amount of flow equal to the demand is distributed on these paths. Taking into account the rational behavior of followers, the leader’s objective is to (in some sense) optimize the resulting travel pattern. This class of bilevel problems will be referred to as bilevel shortest path (BSP) problems.

The main objective in the current chapter is to describe the context and give a general model for BSP problems. We also present the standard techniques to reformulate the general model as a mixed integer linear program (MILP). To make this more concrete, two applications are modelled as specific BSP problems: optimization problems in IP networks and a tariff optimization problems.

A prerequisite to modelling and solving BSPs is a solid knowledge of the classical shortest path problem and its optimality conditions. Based on these, the inverse shortest path routing (ISPR) problem can be formulated. Feasibility of the latter constitutes a main ingredient of the approach taken in this thesis.

Outline The shortest path problem is introduced in Section 2.1. A basic variant of the ISPR problem is presented in Section 2.2. Then, the general model for bilevel shortest path problems is described and reformulated as a bilinear single-level problem in Section 2.3. Two applications that are naturally described as BSP problems are considered in Section 2.4. Finally, two solution approaches are outlined in Section 2.5.
2.1 The Shortest Path Problem

The shortest path problem may be the most fundamental problem in combinatorial optimization. Given a digraph, \( G = (N, A) \), and an arc cost function, \( c : A \rightarrow \mathbb{Q} \), the basic, or single-pair, version of the problem is to find some shortest path w.r.t. the arc cost function, \( c \), from an origin node, \( s \in N \), to a destination node, \( t \in N \). It is well known that this problem is NP-hard in general but polynomially solvable when the arc cost function does not induce a directed negative cost cycle in \( G \).

In this thesis, we will only consider the polynomially solvable case. In fact, for our purposes it often suffices to assume that \( c \) is non-negative, i.e. \( c : A \rightarrow \mathbb{Q}_+ \). Further, it will be assumed that \( G \) is strongly connected, i.e. there is a path between every pair of nodes in \( G \).

Some well known properties of the shortest path problem will be required later in this thesis. These results are presented below from a linear programming (LP) perspective. The standard formulation of the single-pair shortest path problem is based on a minimum cost flow problem formulation, see e.g. [4]. The variable \( x_a \) can be seen as the amount of flow on arc \( a \in A \) or as a binary indicator of whether the arc is on the shortest path. This yields,

\[
\begin{align*}
\text{minimize} & \quad \sum_{a \in A} c_a x_a \\
\text{subject to} & \quad \sum_{a \in \delta^+(i)} x_a - \sum_{a \in \delta^-(i)} x_a = b_i, \quad i \in N, \\
& \quad x_a \geq 0, \quad a \in A,
\end{align*}
\]

(2.1)

where the node balance vector, \( b_i \), in the single-pair variant is defined as

\[
b_i := \begin{cases} 
1, & \text{if } i = s, \\
-1, & \text{if } i = t, \\
0, & \text{otherwise.}
\end{cases}
\]

(2.2)

Besides the single-pair version of the shortest path problem, there are two other variants that are often considered: the single-source (or single-destination) and the all-pairs shortest path problems. In the former problem, the shortest paths from the origin, \( s \), to all destinations are sought, and in the latter, the shortest paths between all pairs of nodes are sought.

In theory, the single-source shortest path problem is not harder than the single-pair version. Several algorithms actually implicitly determine a shortest path to all destinations, e.g. Dijkstra’s algorithm [91]. Model (2.1) can be adapted to the single-source problem by changing the node balances in (2.2) to

\[
b_i := \begin{cases} 
n - 1, & \text{if } i = s, \\
-1, & \text{otherwise.}
\end{cases}
\]

(2.3)

Remark 2.1. Note that the change of node balances from (2.2) to (2.3) implies that the interpretation of the variable \( x_a \) as a shortest path indicator is no longer accurate, but the
flow interpretation is. Also, the interpretation of the objective changes. The disaggregated node balance formulation, (2.2), is often preferred when (2.1) is part of a larger model since it typically yields much stronger (but larger) formulations. The aggregated node balances, (2.3), will be used when the optimality condition aspect is considered, in particular when ISPR problems are modelled.

Several algorithms exist for solving the shortest path problem efficiently. The most well known may be Dijkstra’s algorithm, first presented in [91], which implicitly also finds a shortest path from the origin, s, to all destinations. Using Fibonacci heaps, Dijkstra’s algorithm can be implemented to run in $O(m + n \log n)$ [113], where as usual $n = |N|$ and $m = |A|$. A survey of running time bounds for the shortest path problem is given in [205, Chapter 7.5], see also [4, Chapter 4].

When there is only one destination, the $A^*$ algorithm, initially given in [134], can be used. It improves upon Dijkstra’s algorithm by adding information via an optimistic heuristic (referred to as an admissible heuristic in this context) that underestimates the distance to the destination to reduce the number of unnecessary node expansions.

The all-pairs shortest path problem can be solved by Floyd–Warshall’s algorithm or Johnson’s algorithm, introduced in [105] and [146], respectively. Floyd–Warshall’s algorithm is a dynamic programming algorithm which is very easy to implement, but runs in $O(n^3)$, while Johnson’s algorithm runs in $O(n^2 \log n + nm)$, which is asymptotically superior for sparse graphs.

Textbook descriptions of the above and related algorithms are given in [201], [205, Chapter 7-8] and [4, Chapter 4-5]. The latter also provides discussions of implementation issues and more detailed complexity analyzes.

The properties of the polyhedron formed by the set of feasible solutions to (2.1), using node balances (2.2) or (2.3), will be considered as common knowledge. Like all polyhedra, the polyhedron can be decomposed into a polytope and a pointed cone. By assumption, there is no negative cycle, hence there exists an optimal solution within the polytope. For an extremal point of the polytope it follows by total unimodularity that $x$ is integral. Since an extremal point corresponds to a basic feasible solution, the linear independence implies that the arcs corresponding to basic variables form a shortest path tree rooted at $s$. When (2.3) is used, the tree arcs are just the non-zero arcs.

Duality is crucial in combinatorial optimization. As for many problems, the theory and solution methods for shortest path problems rely on duality theory, e.g. Dijkstra’s algorithm rather solves the LP dual of (2.1). In particular, several results in this thesis are based on the LP complementarity slackness optimality conditions adapted to (2.1). The LP dual of (2.1) using the node balances in (2.2) is

\[
\text{maximize} \quad \pi_s - \pi_t \\
\text{subject to} \quad c_a + \pi_i - \pi_j \geq 0, \quad a := (i, j) \in A, \quad (2.4a)
\]

where the dual variables, $\pi \in \mathbb{Q}^N$, associated with the flow conservation constraints (2.1a) are also referred to as node potentials. These variables are also often called distance labels. If $\pi_s$ is set to zero, then if the node balances in (2.3) are used, the value of $\pi_i$ in the unique optimal solution is indeed the distance from $s$ to $i$. When (2.2) is used, there
are typically multiple optimal solutions and some \( \pi_i \) may only be a valid lower bound on the actual distance from \( s \) to \( i \).

**Definition 2.1**
A node potential, \( \pi \in \mathbb{Q}^N \), is feasible if

\[
c_a + \pi_i - \pi_j \geq 0, \quad a := (i, j) \in A.
\] (2.5)

The left hand side in (2.5) will be denoted by \( \hat{c}_a := c_a + \pi_i - \pi_j \) and referred to as the reduced cost of arc \( a := (i, j) \). A shortest path solution can be found by using the complementary slackness conditions, i.e. an arc is in a shortest path if and only if it has reduced cost zero. This yields the following well-known theorem.

**Theorem 2.1**
An arc, \( a := (i, j) \), is in some shortest path from the root node, \( s \), if and only if there exist a feasible node potential, \( \pi \), where

\[
c_a + \pi_i - \pi_j = 0.
\] (2.6)

These optimality conditions are frequently used in the modelling of inverse shortest path problems.

### 2.2 The Inverse Shortest Path Routing Problem

The inverse shortest path (ISPR) problem is to decide if a set of tentative routing patterns are simultaneously realizable as shortest paths. In this section, a very brief introduction to this problem is given. In particular, we present a model for a simplified variant of the ISPR problem that is sufficient for the reformulation of the BSP problem in the next section. The ISPR problem is thoroughly analyzed in Part II of this thesis.

Let \( G = (N, A) \) be a strongly connected digraph and \( L \subseteq N \) a set of destination nodes. For each destination, \( l \in L \), a (tentative) routing pattern is represented by a shortest path ingraph (SP-graph), defined by an arc subset pair, \((A^l, \bar{A}^l) \subseteq A \times A\). The arcs in \( A^l \) are required to be shortest path arcs (SP-arcs) and the arcs \( \bar{A}^l \) are required to be non-shortest path arcs (non-SP-arcs), i.e. prohibited to be on a shortest path. A family of SP-graphs, \( \mathcal{G}^L := \{(A^l, \bar{A}^l) : l \in L\} \) is realizable if there is a strictly positive cost vector, \( w \in \mathbb{Q}^A_+ \), such that all SP-arcs in all SP-graphs are in some shortest path to their respective destinations and no non-SP-arc is in a shortest path to its destination.

The ISPR problem is to decide if a family of SP-graphs is realizable. The main result in Chapter 4, see also [74], is that ISPR is in general NP-complete.

An important relaxation of ISPR, referred to as compatibility, is to decide if there is a strictly positive cost vector, \( w \in \mathbb{Q}^A_+ \), such that for each \( l \in L \) there is a node potential, \( \pi^l \in \mathbb{Q}^N \), such that the implied reduced costs are compatible with \((A^l, \bar{A}^l)\), i.e.

\[
w_a + \pi^l_i - \pi^l_j \begin{cases} = 0, & \text{if } a := (i, j) \in A^l, \\ > 0, & \text{if } a := (i, j) \in \bar{A}^l, \\ \geq 0, & \text{otherwise.} \end{cases}
\] (2.7)
2.2 The Inverse Shortest Path Routing Problem

The rationale of the compatibility relaxation follows from Theorem 2.1, i.e. in a feasible solution to (2.1), using node potentials (2.3), an arc is an SP-arc if and only if the associated reduced cost is zero. The weakness of compatibility, i.e. the reason that it only gives a relaxation of the ISPR problem, is that it only takes dual feasibility into account, but partly neglects primal feasibility and complementary slackness, see Chapters 4 and 5.

A mathematical model (widely available in the OSPF literature, see below) is directly obtained from the above description. Let $w_a$ be the cost (the term administrative weight is common in the OSPF context) for arc $a := (i, j) \in A$, and $\pi_i^l$ the node potential for node $i \in N$ and destination, $l \in L$. Assume that $\epsilon = 1$ is a lower bound on the strictly positive reduced costs; this holds for instance when the weights are required to be integral as in the OSPF case. We also assume that all arc costs must be at least 1. This yields the formulation,

\[
\begin{align*}
[\text{ISPR-C}] \\
& w_a + \pi_i^l - \pi_j^l = 0, & a := (i, j) \in A^l, & l \in L, \\& w_a + \pi_i^l - \pi_j^l \geq 1, & a := (i, j) \in \bar{A}^l, & l \in L, \\& w_a + \pi_i^l - \pi_j^l \geq 0, & a := (i, j) \in A \setminus (A^l \cup \bar{A}^l), & l \in L, \\& w_a \geq 1, & a \in A.
\end{align*}
\]

Similar, essentially equivalent, models are found in the literature, e.g. in [25, 46, 50, 61, 62, 73, 191, 193].

A family of SP-graphs, $A^L$, is compatible if and only if (2.8) is feasible. Intuitively, $A^L$ is not compatible if a subset of SP-arcs and non-SP-arcs directly induce a reduced cost routing conflict. Two examples of such conflicts are given in Example 2.1. Several routing conflict examples are given throughout the thesis.

--- Example 2.1

\[\text{(a)}\] A simple routing conflict later referred to as subpath inconsistency.

\[\text{(b)}\] A slightly more complicated routing conflict later referred to as a valid cycle.

*Figure 2.1: Two (potential) routing conflicts involving two destinations. SP-arcs to destination $l_0$ and $l_1$ are represented by solid and dashed arcs, respectively. Non-SP-arcs have been omitted.*

Let $L := \{l_0, l_1\} \subset N$ be a set of destination nodes. If the sets of SP-arcs, $A^{l_0}$ and $A^{l_1}$, to destinations $l_0$ and $l_1$, respectively, contain the arcs indicated in Figure 2.1a, i.e.

\[
A^{l_0} \supseteq \{(1, 2), (2, 3)\}, \tag{2.9}
\]
\[
A^{l_1} \supseteq \{(1, 3)\}. \tag{2.10}
\]
then one can (and we will later) show, that the same arcs must also be SP-arcs for the other destination, i.e.

\[ A^{l_0} \supseteq \{(1,3)\}, \quad (2.11) \]

\[ A^{l_1} \supseteq \{(1,2), (2,3)\}. \quad (2.12) \]

Hence, the SP-arc sets, \( A^{l_0} \) and \( A^{l_1} \), depicted in Figure 2.1a forms a potential routing conflict. Indeed, as soon as one of the arcs in these sets are required to be a non-SP-arc, a conflict arises, i.e. if

\[ \{(1,2), (2,3), (1,3)\} \cap \bar{A}^{l_0} \neq \emptyset \quad \text{or} \quad \{(1,2), (2,3), (1,3)\} \cap \bar{A}^{l_1} \neq \emptyset. \quad (2.13) \]

Observe that this kind of routing conflict was presented already in Example 1.1 on page 2, along with an ad hoc argument for the SP-arc and non-SP-arc set claims above. A more complicated example where it is non-trivial to provide an ad hoc argument is given in Figure 2.1b. We will later show that it is not possible that the indicated arcs are simultaneously SP-arcs, i.e. that

\[ A^{l_0} \supseteq \{(1,4), (3,2)\}, \quad \text{and} \quad A^{l_1} \supseteq \{(1,2), (3,4)\}, \quad (2.14) \]

unless also

\[ A^{l_0} \supseteq \{(1,2), (3,4)\}, \quad \text{and} \quad A^{l_1} \supseteq \{(1,4), (3,2)\}. \quad (2.15) \]

Our primary motivation for studying ISPR problems is that they naturally arise as crucial subproblems when BSP problems are solved in methods based on generating cutting planes to prohibit routing conflicts.

### 2.3 Bilevel Shortest Path Problems

To model a BSP problem in \( G = (N,A) \), let \( K \subseteq N \times N \) be a set of followers. Each follower, \( k := (o^k, d^k) \in K \), is associated with an OD-pair with origin \( o^k \) and destination \( d^k \). For each follower, \( k \in K \), there is a demand, \( h^k \), that must be sent from \( o^k \) to \( d^k \).

We use three sets of variables: the leader’s control variables, \( u \), the followers’ flow variables, \( x \), and the arc cost variables, \( w \). The control variables, \( u \), are highly dependent upon the actual application. The follower’s flow variable, \( x_{a}^{k} \), denotes the fraction of the demand, \( h^{k} \), sent on an arc, \( a \in A \), by follower \( k \in K \). The cost, \( w_{a} \), for arc \( a := (i,j) \in A \) depends (possibly implicitly) on the leaders control variables, \( u \), and possibly also on the induced flow. The exact (application dependent) relation is modelled via the set \( W(u,x) \). It can be assumed that the cost vectors in \( W(u,x) \) do not induce negative cost cycles. Finally, the feasible combinations of leader decisions and follower flow assignments are modelled via the set \( \Omega \).
The objective is for the leader to maximize an objective function, \( F(u, x, w) \), while followers minimize their costs by using shortest paths w.r.t. \( w \). Hence, the leader solves

\[
\begin{align*}
\text{maximize} & \quad F(u, x, w) \\
\text{subject to} & \quad (u, x, w) \in \Pi, \\
& \quad w \in W(u, x), \\
& \quad x^k \in S^k(w), \quad k \in \mathcal{K},
\end{align*}
\]

where \( S^k(w) \) denotes the set of optimal solutions to the shortest path problem associated with follower \( k \in \mathcal{K} \) given the costs \( w \), i.e.

\[
\begin{align*}
\text{minimize} & \quad \sum_{a \in A} w_a \bar{x}_a \\
\text{subject to} & \quad \sum_{a \in \delta^+(i)} \bar{x}_a - \sum_{a \in \delta^-(i)} \bar{x}_a = b^k_i, \quad i \in \mathcal{N}, \\
& \quad \bar{x}_a \geq 0, \quad a \in A,
\end{align*}
\]

and

\[
b^k_i := \begin{cases} 
1, & \text{if } i = o^k, \\
-1, & \text{if } i = d^k, \\
0, & \text{otherwise},
\end{cases}
\]

Some concrete examples of the abstract entities \( F, W \) and \( \Pi \) are given in the next section.

We reformulate (2.16) into a single-level program with bilinear constraints by using the common approach based on the complementary slackness optimality conditions, see e.g. [157, 158, 172]. Another common option is to use strong duality of linear programming, i.e. that the optimal values of the primal and dual problems coincide, see e.g. [157, 172]. The absence of negative cost cycles in the followers’ shortest path problems implies that these approaches are feasible, i.e. Theorem 2.1 applies; indeed, a rational leader’s optimal decisions satisfy this assumption.

Modelling LP complementary slackness optimality conditions or strong duality requires primal and dual feasibility. The former is handled by the constraints in (2.17) and the latter is handled in the same manner as in (2.8) in the previous section. Thus, node potentials are introduced and the reduced costs are restricted appropriately for all arcs. Note that it is possible to combine all followers with the same destination and let them share the node potential, cf. Section 2.1. Denote the set of destinations by

\[
L := \{ i \in \mathcal{N} \mid i = d^k \text{ for some } k \in \mathcal{K} \}
\]

and let the followers with destination \( l \) be denoted by \( \mathcal{K}^l := \{ k \in \mathcal{K} \mid d^k = l \} \). To use model (2.8), we introduce a binary shortest path indicator variable \( y_a^l \) for each arc \( a \in A \).

Each \( y_a^l \) is set to 1 if \( x_a^l > 0 \) for some \( k \in \mathcal{K}^l \) and 0 otherwise. Observe the connection to
SP- and non-SP-arcs, i.e. \( y_a^l = 1 \) if \( a \) is an SP-arc w.r.t. destination \( l \). Again using \( \epsilon \) as a lower bound on strictly positive reduced costs, we obtain the integer bilinear problem,

\[
\begin{align*}
\text{maximize} & \quad F(u, x, w) \\
\text{subject to} & \quad \sum_{a \in \delta^+(i)} x^k_a - \sum_{a \in \delta^-(i)} x^k_a = b^k, \quad i \in N, k \in K, \\
& \quad w_a + \pi^l_i - \pi^l_j \geq \epsilon(1 - y_a^l), \quad a := (i, j) \in A, k \in K^l, \\
& \quad (w_a + \pi^l_i - \pi^l_j) y_a^l = 0, \quad a := (i, j) \in A, k \in K^l, \\
& \quad 0 \leq x^k_a \leq y_a^l, \quad a \in A, k \in K^l, l \in L, \\
& \quad y_a^l \in \mathbb{B}, \quad a \in A, l \in L, \\
& \quad (u, x, w) \in \Pi, \\
& \quad w \in W(u, x).
\end{align*}
\]

(2.20a)

(2.20b)

(2.20c)

(2.20d)

(2.20e)

(2.20f)

(2.20g)

Instead of using the complementarity constraint, (2.20c), it is sometimes preferable to use strong LP duality, i.e. to augment model (2.20) or replace (2.20c) by

\[
\sum_{a \in A} w_a x^k_a = \pi^l_{d_k} - \pi^l_{o_k}, \quad k \in K^l, l \in L.
\]

(2.20c')

A potential pitfall in model (2.20) is the connection between the \( x \) and \( y \) variables. Indeed, if \( x^k_a > 0 \), then \( y_a^l \) must be 1 for \( l = d_k \). However, if \( y_a^l = 1 \) it is still possible that \( x^k_a = 0 \) for each \( k \in K^l \).

In the remainder of the thesis we will be particularly interested in the projection onto the shortest path indicator variables, \( y \), i.e. the set of paths simultaneously realizable as shortest paths, or the feasible routing patterns,

\[
\mathcal{Y}(K) := \{ \bar{y} \in \mathbb{B}^{A \times L} \mid (2.20a) - (2.20e) \text{ has a feasible solution with } y = \bar{y} \}. \tag{2.21}
\]

We will later give alternative definitions of \( \mathcal{Y}(K) \) and describe it as an integer linear inequality system. The exact solution methods that we consider are based on generating a good approximation of the convex hull of \( \mathcal{Y}(K) \). When the connection between \( x \) and \( y \) mentioned above is weak, this approach may be ineffective. A desirable situation is when the values of the \( x \)-variables are uniquely determined from the \( y \)-variables, and vice versa. An example of this is when all shortest paths are required to be unique.

Since (2.20) is very general, we specify the objective, \( F \), and the sets \( \Pi \) and \( W \) for some prominent applications that fit into the BSP framework in the following section. When these applications are considered, we also address some important BSP issues, e.g. handling non-uniqueness of shortest paths and linearization. Thereafter, two solution approaches are outlined in Section 2.5. In Parts II and III, we will focus on the core of the model: constraints (2.20a)-(2.20e), i.e. the set of feasible routing patterns, \( \mathcal{Y}(K) \).
2.4 Examples of Bilevel Shortest Path Applications

Since the shortest path structure frequently occurs in problems in networks and often is in line with rational user behavior, it is clear that BSPs arise in several applications. Examples include: Stackelberg network pricing games [57, 219], network interdiction [87, 143, 162], revenue management [66, 67], yield management in the airline industry [83, 172], pricing in telecommunication networks to maximize revenues and manage traffic [56, 172], pricing in electricity markets [224], transportation of hazardous material [171] and traffic assignment [135, 175, 188]. More examples are found in the annotated bibliography [86]. Here we focus on IP routing and tariff optimization.

2.4.1 IP Network Routing Problems

Routing in IP networks is often conducted in accordance with an SPR protocol, e.g. OSPF [177] or IS-IS [76]. This means that all routing paths are shortest paths w.r.t. some artificial arc costs, in this setting referred to as administrative weights. Therefore, the majority of the Internet traffic is directed by SPR. Since the Internet is steadily growing, this class of optimization problems is of major importance for the quality of service (QoS) provided to customers.

Since our primary concern is routing in IP networks, we give a very brief presentation of the technical background and describe the structure of the Internet and how SPR usually works.

Technical Background

The basic building blocks of the Internet are smaller subnetworks called routing domains or autonomous systems (AS). The operator of an AS is called an Internet service provider and is responsible of the routing of the traffic within the domain, i.e. the determination of the path from a source to a destination for every single data package. This decision heavily affect the performance of the network and has to be made very quickly. Therefore, a network operator rely on a routing protocol to perform these decisions, i.e. a standardized specification of how the traffic is routed in a network. The single most important task for the operator is to select a routing protocol and its parameters to provide an acceptable level of the QoS experienced by customers. This is referred to as traffic engineering.

Within an AS, the routing is conducted by routers via static or dynamic routing tables. Static routing implies that paths are configured manually, which is feasible for small domains. However, in larger domains, dynamic routing is more common. The routers maintain the routing tables by communicating with each other via an interior gateway protocol (IGP). This implies that the routing paths are no longer selected manually, but by the parameters of the routing protocol. There are several IGPs, e.g. RIP, IS-IS, OSPF, IGRP and EIGRP. All protocols include the administrative weights as parameters. Actually, the weights are the only means an operator have to (indirectly) control the traffic.

The open shortest path first (OSPF) protocol and the intermediate system to intermediate system (IS-IS) are the most common IGPs. In OSPF and IS-IS, it is required that all administrative weights are integral and in the interval 1 to 65536 and $2^{24} - 1$, respectively. In practice, shortest paths are easily determined, e.g. by Dijkstra’s algorithm, and stored
implicitly in forwarding tables at the routers. This is very efficient since it only requires a
lookup of the next hop on the path to the destination.

A standard of how to deal with multiple shortest paths is not specified in the current
OSPF [177], nor IS-IS [76], specification. Therefore, most authors that consider opti-
mization problems in IP networks require all shortest paths to be unique; this is referred
to as unique shortest path routing (USPR). Some authors have also considered using mul-
tiple shortest paths. The common assumption used in the mathematical modelling of these
protocols is then the following equal cost multi-path (ECMP) splitting rule. If, at a node,
there are several outgoing arcs that are on shortest paths to a given destination, then the
ingoing traffic for that destination to this node is evenly divided among the outgoing arcs.
Note that this is in general not the same as an even distribution of the traffic on all shortest
paths, see Example 2.2 below.

--- Example 2.2 ---

The ECMP principle is demonstrated for the set of administrative weights in the left of
Figure 2.2. The induced flow from node $O$ to node $D$ is shown in the right of Figure 2.2.
There are 3 shortest paths, and two of them carry 0.25 units of flow and the last carries
0.5 units of flow.

![Figure 2.2: The weights and induced flow according to the ECMP principle.](image)

In practice, it is common that network operators use some default weight settings.
The simplest idea is to use the hop count, i.e. all administrative weights are set to 1. An
apparently more sophisticated choice of administrative weights, recommended in [79], is
to use a weight that is inversely proportional to the capacity of the link. This yields lower
weights, and therefore more traffic, on high capacity links. It turns out that both these
suggested settings often perform poor in minimizing the link load, cf. [100, 109].

If an operator does not prefer to use a default setting it is easy to determine the routing
induced by any administrative weights and then use some simulation procedure to mea-
sure the network performance in different senses. Unfortunately, it is not clear how to
adjust the weights if the shortest paths or performance measure(s) are not satisfactory. A
major problem is that the control of the flow distribution is only indirect, which makes
it hard to foresee or estimate some, or all, effects of the adjustments without potentially
expensive calculations. In practice, and from an engineer’s perspective, it may be enough
to evaluate the measure for a reasonably large collection of weights. To decide which
settings to evaluate, search methods in the weight space may be used, e.g. tabu search,
simulated annealing and other metahuristics. This pragmatic approach is considered in
Section 2.5.1.
2.4 Examples of Bilevel Shortest Path Applications

A Traffic Engineering Problem

We adopt model (2.20) to a minimalistic traffic engineering problem in an IP network where the QoS is measured by the most congested link. Here, \( G = (N, A) \) represents an IP network; \( N \) corresponds to routing devices and \( A \) corresponds to links between routing devices. Further, each link, \( a \in A \), has capacity, \( u_a \). Typically, the number of OD-pairs is large, i.e. \(|K| \in \mathcal{O}(n^2)\). In an IP network, traffic must be routed along shortest paths w.r.t. some integral administrative weights. If there are multiple shortest paths, the traffic should be divided in accordance with the ECMP splitting principle. The objective for the network operator is to minimize the utilization of the most congested link, measured by \( \zeta \), i.e.

\[
\begin{align*}
\text{minimize} & \quad \zeta \\
\text{subject to} & \quad \sum_{a \in \delta^+(i)} x_a^k - \sum_{a \in \delta^-(i)} x_a^k = b^k, \quad i \in N, k \in K, \\
& \quad w_a + \pi^l_i - \pi^l_j \geq 1 - y_{a}^l, \quad a := (i, j) \in A, k \in K^l, \\
& \quad (w_a + \pi^l_i - \pi^l_j) y_{a}^l = 0, \quad a := (i, j) \in A, k \in K^l, \\
& \quad \sum_{k \in K} h^k x_a^k \leq u_a \zeta, \quad a \in A, \\
& \quad 0 \leq x_a^k \leq y_{a}^l, \quad a \in A, k \in K^l, l \in L, \\
& \quad 0 \leq x_a - v_{a}^k \leq 1 - y_{a}^l, \quad a \in \delta^+(i), i \in N, k \in K^l, l \in L, \\
& \quad 1 \leq w_a \leq w_{MAX}, \quad a \in A. \\
\end{align*}
\]

Model (2.22) specify \( F \) and the sets \( \Pi \) and \( W(u, x) \) as required in (2.20). In particular, constraint (2.22f) models ECMP and constraint (2.22d) introduces link capacities. The ECMP principle is handled by the auxiliary variables, \( v_{a}^k \). They determine the common flow value on shortest path arcs emanating from \( i \) to node \( l = d^k \). If USPR is considered, it suffices to bound the outdegrees accordingly, i.e.

\[
\sum_{a \in \delta^+(i)} y_{a}^l \leq 1, \quad i \in N, l \in L, \tag{2.22f'}
\]

which also makes the ECMP constraint (2.22f) redundant.

We stress that routing in IP networks is much harder than multicommodity routing since SPR yields restrictions both on the paths that can be used and the amount of flow that may be sent along the paths via USPR or ECMP. It is indeed very unlikely that an optimal multicommodity flow solution is realizable in an SPR protocol. Therefore tailored mathematical models and solution methods that take SPR into account must be developed.
A major issue with these models is that the control of the flow distribution is indirect via the administrative weights. There are two approaches to handle this. Directly, by including the weights and the SPR constraints, (2.22b) and (2.22c), in the model to simulate the SPR protocol, or indirectly, by prohibiting infeasible routing patterns.

At first, the direct approach seems natural, i.e. using model (2.22). To solve (2.22) as a MILP, the complementarity constraints (2.22c) can be linearized with big-$M$'s. The major drawback with this approach is the big-$M$'s since they may have to be as large as a longest shortest path in the graph, cf. Theorem 9.1 in Chapter 9. This big-$M$ is typically huge. (recall that $w_{MAX}$ is $2^{16} - 1$ or even $2^{24} - 1$). Hence, the LP relaxation is typically extremely weak and does not really improve upon the multicommodity relaxation completely neglecting the SPR aspect.

The indirect, and less intuitive, approach of omitting, i.e. projecting out, the weights from the model and prohibiting infeasible routing patterns in a combinatorial Bender’s fashion is covered in detail in the remainder of this thesis.

To the best of our knowledge, the first MILP formulations of a routing problem in an IP network using SPR were given in [51] for USPR and in [139, 225] for ECMP splitting. We believe that the first model without the weight variables is from [52]. The reader is referred to the book [191] for an early overview of routing problems in IP networks and early models. Some MILP approaches to IP routing problems encountered in the literature include [46, 47, 49, 85, 93, 94, 95, 139, 187, 190, 192, 218].

In summary, some distinguishing features of IP network routing problems are:

- The actual weights do not really matter. They can in theory be very large, resulting in very poor LP relaxations when the complementarity constraints are linearized with big-$M$'s.
- The handling of multiple shortest paths is well defined via USPR or ECMP splitting and does not admit an arbitrary optimal solution to a followers shortest path problem. This implies that the connection between $x$ and $y$ in (2.22) is very strong.
- The number of OD-pairs is in practice large, often maximal, i.e. $O(n^2)$.
- Congestion is taken into account which implicitly induces the traffic to bifurcate. Further, there is no underlying problem structure that induces traffic to naturally select shortest paths. Hence, there is an overwhelming risk of routing conflicts.
- Only small, or even very small, instances can be solved to optimality.

### 2.4.2 Tariff Optimization Problems

Tariff optimization problems are naturally posed as Stackelberg network pricing problems, a prime example of bilevel programming problems. In fact, they frequently occur as illustrative introductory examples in this context. The tariff optimization problem considered here was introduced in [157]. It is a highway pricing problem where the operator (leader) can set tolls on a subset, $A_1 \subset A$, of arcs to maximize its revenue determined as the toll, $T_a$, times the number of travellers. If there are multiple shortest paths the leader freely selects an optimal solution for the follower that best suits the objective. Adapting model (2.20) to this problem yields,
maximize \[ \sum_{a \in A_1} \sum_{k \in K} h^k x^k_a \]
subject to
\[ \sum_{a \in \delta^+(i)} x^k_a - \sum_{a \in \delta^-(i)} x^k_a = \delta^k_i, \quad i \in N, k \in K, \quad (2.23a) \]
\[ w_a + \pi^l_i - \pi^l_j \geq \epsilon(1 - y^l_a), \quad a := (i,j) \in A, l \in L, \quad (2.23b) \]
\[ \sum_{a \in A} w_a x^k_a = \pi^l_{\delta^k_a} - \pi^l_{\delta^{k-1}_a}, \quad k \in K^l, l \in L, \quad (2.23c) \]
\[ w_a = d_a, \quad a \in A \setminus A_1, \quad (2.23d) \]
\[ w_a = d_a + \bar{T}_a, \quad a \in A_1, \quad (2.23e) \]
\[ 0 \leq x^k_a \leq y^l_a, \quad a \in A, k \in K^l, l \in L, \quad (2.23f) \]
\[ x^k_a \in \mathbb{R}, \quad a \in A, k \in K^l, \quad (2.23g) \]
\[ w \in \mathbb{Z}^A, \quad \pi \in \mathbb{R}^{N \times L}, \quad T \in \mathbb{R}^A, \quad (2.23h) \]
\[ x \in \mathbb{R}^{A \times K}, \quad y \in \mathbb{B}^{A \times L}. \quad (2.23i) \]

Note that tolls may be negative in (2.23), i.e. subsidies are allowed (and can be part of an optimal solution, see e.g. [157]). It is even possible that arc costs are negative. However, a rational leader will never select tolls inducing a negative cycle. Also observe that the complementarity constraints in the general BSP model, (2.20c), have been replaced by the strong duality constraints, (2.20c’), i.e. (2.23c) in (2.23). The version with complementarity constraints also occurs in the literature, e.g. in [66, 67, 157]. However, the most frequently occurring model in the literature, e.g. [56, 65, 66, 90, 157], is obtained from (2.23) by setting \( y^l_a = 1 \) for all \( a \in A \) and \( l \in L \).

This illustrates that there is no connection between the \( x \) and \( y \) variables when (2.20c’) is used. If (2.20c) is used, there is a connection, but it is still possible that some arc \( a \in A \) has \( y^l_a = 1 \) while \( x^k_a = 0 \) for all \( k \in K^l \). Hence, the connection is very weak for this variant of the tariff optimization problem. We discuss some problem modifications where this is not the case below.

Model (2.23) is bilinear due to the terms \( \bar{T}_a x^k_a \). By assumption, the leader freely chooses among the followers optimal solutions. In particular, there is an extreme optimal solution to each followers’ problem that suits the leader and therefore an optimal solution to (2.23) where \( x^k_a \) is binary. Hence, the bilinear terms \( \bar{T}_a x^k_a \) can be linearized by using an auxiliary variable \( \bar{T}_a := \bar{T}_a x^k_a \) as follows,

\[ -M x^k_a \leq \bar{T}_a \leq M x^k_a, \quad a \in A_1, k \in K, \quad (2.24a) \]
\[ -M (1 - x^k_a) \leq \bar{T}_a - \bar{T}_a \leq M (1 - x^k_a), \quad a \in A_1, k \in K, \quad (2.24b) \]
\[ \bar{T}_a \in \mathbb{B}, \quad a \in A, k \in K. \quad (2.24c) \]
Appropriate values of $M$ in this linearization are derived in [88, 89]. In general, these values are not very large since they are related to the maximal toll level that can be profitable, i.e. the "tollable gap" between a shortest path without tolled arcs and a shortest path with tolled arcs. Since the maximum arc costs are small, e.g. $w_{\text{MAX}} \leq 40$ in [65], this implies that $M$ is relatively small, cf. Section 2.4.1.

Finally, we address the issue of non-uniqueness in a followers shortest path problem. Recall that the polytope $S^k(w)$ denotes the set of optimal solutions to the shortest path problem (2.17) for follower $k \in K$ using arc costs $w \in R^A$.

In the literature, and in model (2.23), this issue is handled by letting the leader choose a solution in $S^k(w)$. Since the leader maximizes profit, this results in a solution

$$x^k \in \underset{x \in S^k(w)}{\text{argmax}} \sum_{a \in A_1} T_a x_a.$$  (2.25)

Note that this approach is compatible with the upper level problem in the sense that it is not necessary to handle the optimization problem (2.25) explicitly. Indeed, the leader will select an optimal solution to this problem anyway.

A requirement for the above approach to be applicable is that all followers are fully cooperative. It yields an extreme and optimistic toll scheme. The other extreme is to consider fully non-cooperative followers. In this pessimistic worst-case approach it is assumed that each follower selects a solution in $S^k(w)$ that is as bad as possible for the leader, i.e.

$$x^k \in \underset{x \in S^k(w)}{\text{argmin}} \sum_{a \in A_1} T_a x_a.$$  (2.26)

Note that this implies that it is necessary to handle the optimization problem (2.26) explicitly in model (2.23).

A third approach is to only consider toll schemes that induce unique solutions to the lower level problems. This implies that the connection between $x$ and $y$ considered above is very strong. Also, it allows us to add some strong inequalities (to be presented in Part III) to model (2.23). Hence, it may be an interesting choice from a computational perspective. However, it has two major drawbacks. First, it yields overly conservative toll schemes that can at best be as good as an optimal toll scheme produced by the worst-case approach above. Second, it may render the problem infeasible (however, this issue may be remedied by using the SPGM reformulation in [56]). We investigate the worst-case approach further below.

**Tariff Optimization with Non-Cooperative Followers**

To the best of our knowledge, the worst-case version of tariff optimization problems, i.e. where followers are not fully cooperating with the leader, have not earlier been considered in the literature.

Handling followers’ worst-case behavior, i.e. incorporating (2.26) into model (2.23) can be viewed as adding a third optimization level or as a robust approach where the uncertainty set is the polytope of optimal solutions. In both these paradigms, the worst-case aspect is commonly handled via duality.
For notational convenience, we use a toll variable, $\bar{T}_a$, for all arcs, $a \in A,$ and force $\bar{T}_a := 0$ if $a \not\in A.$ For future reference, we write the non-cooperative tariff optimization problem as the following tri-level programming problem.

\[
\begin{align*}
\text{maximize} & \quad \sum_{a \in A_1} \bar{T}_a \sum_{k \in K} h^k x^k_a \\
\text{subject to} & \quad x^k \in \text{argmin} \sum_{a \in A_1} \bar{T}_a x^k_a \\
& \quad x^k \in \text{argmin} \sum_{a \in A} (d_a + \bar{T}_a) x^k_a \\
& \quad \sum_{a \in \delta^+(i)} x^k_a - \sum_{a \in \delta^-(i)} x^k_a = b^k_i, \quad i \in N, \quad k \in K; \\
& \quad x^k \in \mathbb{B}^A, \quad k \in K.
\end{align*}
\]

We re-write this problem as a single-level MILP using the same techniques that are used to derive (2.23). The notation is similar, but with a few, important differences and is therefore restated here.

Let $y^l_a$ be a shortest path indicator variable, i.e. $y^l_a$ is 1 if (and only if) arc $a \in A$ is in some shortest path w.r.t. arc costs $w_a = d_a + \bar{T}_a$ for some OD-pair $k \in K^l.$ Let $x^k_a$ be the fraction of the demand, $h^k$, sent on arc $a \in A$ from $o^k$ to $d^k$ along a shortest path w.r.t. arc costs $\bar{T}_a$ in the subgraph induced by $y^l,$ i.e. where arcs can only be used if they are SP-arcs w.r.t. arc costs $w_a.$ This implies that the $y$-variables are primal variables in the optimization problem induced by (2.27c) and likewise for the $x$-variables and the optimization problem induced by (2.27b). The dual variables, or node potentials, w.r.t. arc costs $w_a = d_a + \bar{T}_a$ in (2.27c) are denoted by $\pi^l_i$ and the node potentials w.r.t. arc costs $T_a$ in (2.27b) are denoted by $\pi^l_{ij}$ for $i \in N$ and $l \in L.$

Given $y \in \mathbb{B}^{A \times K},$ the optimality conditions for the lower level shortest path problems in (2.27b) for all followers are as follows.

\[
\begin{align*}
\sum_{a \in \delta^+(i)} x^k_a - \sum_{a \in \delta^-(i)} x^k_a = b^k_i, & \quad i \in N, \quad k \in K; \\
T_a + \pi^l_i - \pi^l_{ij} & \geq -M (1 - y^l_a), \quad a := (i, j) \in A, \ l \in L; \quad (2.28b) \\
\sum_{a \in A_1} \bar{T}_a x^k_a = \pi^l_{ik} - \pi^l_{jk}, & \quad k \in K; \quad (2.28c) \\
0 \leq x^k_a \leq y^l_a, & \quad a \in A, \quad k \in K^l, \ l \in L. \quad (2.28d)
\end{align*}
\]

To derive the optimality conditions for (2.27c), we use two properties induced by the $x$-variables. First, constraints (2.28a) and (2.28d) implies that $y$ is primal feasible.
Second, the path induced by the \( x \)-variables is optimal and therefore its length can be used to determine the primal objective function value. This yields,

\[
\begin{align*}
    w_a + \pi^i_a - \pi^j_a & \geq e (1 - y^j_a), \quad a := (i, j) \in A, \; l \in L, \\
    \sum_{a \in A} w_a x^k_a &= \pi^k_{a^e} - \pi^i_{a^e}, \quad k \in \mathcal{K},
\end{align*}
\]

(2.29a), (2.29b)

To obtain a linear model, the same linearization technique as above is used, i.e. let \( T^k_a := T_a x^k_a \) and use constraints (2.24). This yields the following MILP formulation of the non-cooperative tariff optimization problem,

\[
\text{maximize} \quad \sum_{a \in A} \sum_{k \in \mathcal{K}} h^k T^k_a
\]

subject to

\[
\begin{align*}
    \sum_{a \in \delta^+(i)} x^k_a - \sum_{a \in \delta^-(i)} x^k_a &= b^k_i, \quad i \in N, \; k \in \mathcal{K}, \\
    w_a + \pi^i_a - \pi^j_a & \geq e (1 - y^j_a), \quad a := (i, j) \in A, \; l \in L, \\
    \sum_{a \in A} (T^k_a + d_a x^k_a) &= \pi^k_{a^e} - \pi^i_{a^e}, \quad k \in \mathcal{K}^l, \; l \in L, \\
    T_a + \pi^i_a - \pi^j_a & \geq -M (1 - y^j_a), \quad a := (i, j) \in A, \; l \in L, \\
    \sum_{a \in A} T_a &= \pi^L_{a^e} - \pi^i_{a^e}, \quad k \in \mathcal{K}^l, \; l \in L, \\
    -M x^k_a & \leq T^k_a \leq M x^k_a, \quad a \in A_1, \; k \in \mathcal{K}, \\
    -M (1 - x^k_a) & \leq T^k_a - T_a \leq M (1 - x^k_a), \quad a \in A_1, \; k \in \mathcal{K}, \\
    0 & \leq x^k_a \leq y^j_a, \quad a \in A, \; k \in \mathcal{K}^l, \; l \in L, \\
    w_a &= d_a + T_a, \quad a \in A \setminus A_1, \\
    T_a &= 0, \quad a \in A \setminus A_1, \\
    T^k_a &= 0, \quad k \in \mathcal{K} \setminus \mathcal{K}_a,
\end{align*}
\]

(2.30a), (2.30b), (2.30c), (2.30d), (2.30e), (2.30f), (2.30g), (2.30h), (2.30i), (2.30j), (2.30k), (2.30l), (2.30m)

The values of the big-M:s can be set as in the ordinary tariff optimization problem, (2.23), from [88, 89].

Note that the connection between \( x \) and \( y \) in (2.30) is strong, especially compared to model (2.23). Indeed, if a \( y \)-variable is 0, then as usual the corresponding arc is not an SP-arc and the associated \( x \)-variables must not be 1. When a \( y \)-variable is 1, this implies that the arc can be an SP-arc, see (2.30b). However, since this also opens up this arc in the worst-case problem (2.27b), i.e. constraint (2.30d), it potentially reduces the leaders income. Hence, there is incintive to set a \( y \)-variable to 1 only if the arc is actually an SP-arc. It is of course also possible to augment the linearized complementarity constraint,
2.5 Solving Bilevel Shortest Path Problems

We consider two solution approaches for BSP problems here. First, the class of heuristic methods based on searching in the arc cost space and then an exact method of solving the MILP models from the previous sections.

2.5.1 A Heuristic Approach: Search in the Arc Cost Space

Conceptually, many BSP problems can be described by an abstract model where the important decision variables are the arc costs, \( w \) and the application dependent decision variables, \( u \). Given the costs, the auxiliary shortest path indicator variables \( y \) are determined from the arc costs. Finally, the actual arc flow, \( x \), is determined from \( y \) and possibly also \( u \) and \( w \). Let \( Y \) and \( X \) be mappings that describe these relations. This yields the conceptual model,

\[
w_a + \pi^i_j - \pi^j_i \leq M \left( 1 - y_a \right), \quad a := (i, j) \in A, \quad k \in K^j,
\]

(2.31)

to model (2.30) to make this connection more explicit.

In summary of the tariff optimization problems considered in this section, we lift the following distinguishing features:

- The arc costs are central and relatively small in practice. Hence, the LP relaxations obtained by big-M linearizations are not affected as much as in the IP routing case.

- There is an underlying shortest path structure in the problem which implies that a path that is beneficial for one OD-pair is also good for other OD-pair. Hence, there is no incitement for traffic to bifurcate or diverge from shortest paths. This implies that the risk of routing conflicts is much smaller than in IP routing problems.

- Multiple shortest paths are handled by implicitly solving a maximization problem in the ordinary tariff optimization problem which makes the connection between \( x \) and \( y \) in (2.23) weak or even non-existing. In the non-cooperative version, a minimization problem is explicitly solved, which makes the connection between \( x \) and \( y \) in (2.30) strong.

- Congestion is usually not taken into account.

- Medium sized instances can be solved, see e.g. [65, 89] where networks with 50-200 nodes and few tollable arcs (5-20%) and OD-pairs (10-40) are solved. We are not aware of computational experiments with many tollable arcs or many OD-pairs.
maximize \( F(u, x, w) \)
subject to
\[
\begin{align*}
y &= Y(w), \quad (2.32a) \\
x &= X(y, u, w), \quad (2.32b) \\
w &\in W(u, x), \quad (2.32c) \\
(u, x, w) &\in \Pi. \quad (2.32d)
\end{align*}
\]

The purpose of model (2.32) is to illustrate that many BSP problems are conceptually easy to describe. Further, since the implicitly defined mappings, \( Y \) and \( X \), typically are easy to evaluate, the conceptual formulation indicates that these problems may be well suited for heuristic approaches. If it is hard to explicitly enforce the constraints \( w \in W(u, x) \) and \( (u, x, w) \in \Pi \), they may be handled implicitly by using some penalty scheme. A heuristic scheme for model (2.32) often has the following main elements:

1. Initialize the penalty scheme. Generate initial arc costs, \( w \), and decisions, \( u \).
   (If possible, take feasibility w.r.t. \( w \in W(u, x) \) into account.)
2. Find the shortest paths, i.e. evaluate \( y = Y(w) \).
3. Determine the flow, i.e. evaluate \( x = X(y, u, w) \).
   (If possible, take feasibility w.r.t. \( (u, x, w) \in \Pi \) into account.)
4. Determine the objective, i.e. evaluate \( F(u, x, w) \).
   (If necessary, adjust the objective w.r.t. the penalty scheme.)
5. Update arc costs, \( w \), and decisions, \( u \).
   (If necessary, update the penalty scheme.)
6. Repeat Step 2-5 until some stopping criteria is fulfilled.

We give a very brief description of these steps here. Since Step 1, 2, 4 and 6 are often straightforward, they are omitted. Step 3 is application dependent and is covered below for the two applications from Section 2.4. After this, Step 5 is discussed.

**Adapting the Heuristic Approach to IP Routing Problems**

A thorough treatment of the heuristic weight space search method for IP routing problems can be found in the survey [50]. Several metaheuristics have been successfully applied to such problems since they were introduced in [108, 109]. Approaches considered include e.g. genetic algorithms [69, 100], simulated annealing [139, 192], Lagrangean relaxation [44, 192], local search [51, 106, 192]. The reader is referred to the recent surveys [8, 50, 106] for information about the success of the metaheuristics approach to solve these problems. For IP routing problems, Step 2 and 3 in the algorithm above can be implemented as follows.

**Algorithm 2.5.1.** Given arc weights, \( w \), determine the induced arc flow \( x \).

**Notation:** the inflow of for OD-pair \( k \) to node \( i \) is denoted by \( x^k_+(i) \).
1. For each destination \( l \in L \) do
   
   (a) Determine the inigraph \( G^l \) as the union of all shortest path arborescences to \( l \),
   i.e. \( a \in G^l \Leftrightarrow y^{a} = 1 \Leftrightarrow w_a + \pi_l^a - \pi_l^j = 0 \).
   
   (b) Find a topological ordering of the nodes in \( G^l \).

2. For each destination \( l \in L \) do
   
   (a) For each OD-pair \( k \in K^l \), set the inflow to the origin of the OD-pair to the traffic demand,
   
   \[ x^k_{+}(o^k) := h^k. \]
   
   (b) Process the nodes in the topological order of \( G^l \). For each node \( i \) and each OD-pair \( k \in K^l_i \)
   
   i. distribute the inflow evenly on all outgoing arcs in \( G^l \),
   
   \[ x^k_a := x^k_{+}(i)/|\delta^-(i)|, \quad a = (i, j) \in \delta^-(i) \cap G^l, \]
   
   ii. increase the inflows accordingly,
   
   \[ x^k_{+}(j) := x^k_{+}(j) + x^k_a, \quad a = (i, j) \in \delta^-(i) \cap G^l. \]

Step 1a can be solved by storing all predecessor indices in Dijkstra’s algorithm or by collecting all arcs with reduced cost 0. This requires \( O(|L|(|m + n \log n|)) \) time which dominates the topological sorting’s \( O(|L|m) \) time. In the flow distribution step, each arc in each inigraph is considered once for each OD-pair, hence this step requires \( O(|K|m) \) time. If only the aggregated arc flow matters, the required time becomes \( O(|L|m) \). In practice, the above implementation may be inefficient. It is often possible to speed up shortest path calculations by using a dynamic version of the shortest path method when weights are updated, see further [70].

A completely different, and seemingly less known, approach of determining the flow induced by a metric is by solving an auxiliary optimization problem. A linear program for this purpose is given in [93].

**Adapting the Heuristic Approach to Tariff Optimization Problems**

Note that for tariff optimization problems, we can only affect the arc costs on some arcs (in \( A_1 \)). Hence, the search is rather in the tariff space. Then, the arc costs are calculated from (2.23d) and (2.23e) which also implies that \( w \in W(u, x) \). To determine the aggregated flow, and hence tariffs, given arcs costs, i.e. perform Step 3, the following algorithm can be used.

**Algorithm 2.5.2.** Given arc distances, \( d \), and tariffs, \( T \), determine the maximal tariffs.

**Notation:** the aggregated inflow to node \( i \) is denoted by \( x^i_{\text{tot}}(i) \) and the aggregated flow on arc \( a \) is \( x^i_{\text{tot}} \).

1. Set the total tariff, \( T^{\text{tot}} \), to 0.
2. For each destination \( l \in L \) do

(a) Determine the ingraph \( G^l \) as the union of all shortest path arborescences to \( l \), w.r.t. arc costs \( w = d + T \).

(b) Solve a longest path problem in \( G^l \) w.r.t. the tariffs \( T \) from all nodes in \( G^l \) to \( l \). Denote the successor of node \( i \) by \( j = s(i) \).

(c) For each OD-pair \( k \in K^l \), set the inflow to the origin of the OD-pair to the traffic demand,
\[
x^\text{tot}_+ (o^k) := h^k.
\]

(d) Process the nodes in the topological order of \( G^l \). For each node, \( i \), do

i. Propagate the flow on the longest path arc, \( a \), emanating from \( i \),
\[
x^\text{tot}_a := x^\text{tot}_+ (i), \quad a = (i, j), \ j = s(i),
\]

ii. increase inflow accordingly,
\[
x^\text{tot}_+ (j) := x^\text{tot}_+ (j) + x^\text{tot}_a, \quad a = (i, j), \ j = s(i),
\]

iii. increase the total tariff,
\[
T^\text{tot} := T^\text{tot} + \bar{T}_a x^\text{tot}_a.
\]

Remark 2.2. If Algorithm 2.5.2 is used in a heuristic for the non-cooperative version of the tariff optimization problem, then a shortest path problem (and not a longest path problem) should be solved in step 2b.

As in the IP routing context, Step 2a requires \( \mathcal{O}(|L|(m + n \log n)) \) time. The longest path problems are solved in \( \mathcal{O}(|L|m) \) time since each ingraph, \( G^l \), is acyclic. Finally, propagating the flow also requires \( \mathcal{O}(|L|m) \) time.

### Updating Arc Costs

The most important issue in a heuristic approach based on searching in the arc cost space is how to update arc costs. This often boils down to selecting suitable neighbourhoods and meta-heuristics.

A crucial issue in designing an algorithm is to obtain a good balance between intensification and diversification, i.e. when to focus the search in promising areas and when to explore new areas of the solution space. Methods aimed at intensification often achieve steep improvements of the objective value and find a good solution quickly, but have a tendency to get stuck in local optima more often. Exploratory methods are less likely to find good solutions initially but are compensated by their reduced risk of getting stuck at local optima.

We give natural examples of a intensification-based neighbourhood in the IP routing problem above where the maximal link utilization is to be minimized. A reasonable idea is to reduce the weight on some edge(s) where the link utilization equals (or is close to) the maximal link utilization. This neighbourhood results in reduced traffic on congested
2.5 Solving Bilevel Shortest Path Problems

links and seems quite plausible, but is likely prone to getting stuck quite fast. An impor-
tant situation where the neighbourhood is very useful is when model (2.22) is solved by
branch-and-bound (B&B) or branch-and-cut (B&C). Then, this heuristic can be applied
at nodes in the enumeration tree for a few iterations to generate feasible solutions from
a current weight assignment, \( w \), even if the remaining part of the solution is not integral.
This point illustrates that heuristics can often play a very important role also in exact
solution methods for BSP problems.

In summary of the heuristic method one can say that it is often an excellent approach
to find good solutions to some BSP problems since this class of problems is typically not
trivial to model and often hard to solve using exact methods. In particular, the approach is
flexible and based on the property that it is often easy to determine the flow and objective
function from the arc costs. The major drawback of the heuristic approach is that it is not
known how far from an optimal solution the best found solution is. Therefore, optimality
can neither be guaranteed, nor verified. Because of this we will from now on consider
exact solution methods.

2.5.2 An Exact Approach: Solve the MILP Formulation

In principle, it is easy to solve many BSP problems to optimality; it suffices to solve the
MILP formulation, i.e. model (2.20) (assuming that the sets \( I \) and \( W \) as well as the ob-
jective, \( F \), can be expressed by linear inequalities). In many cases this formulation is too
weak to be solved to optimality by a state-of-the-art MILP solver such as CPLEX [142],
GuRoBi [132] or XPress [222] with reasonable time and memory limitations. In particu-
lar, this is the case for IP routing problems. The currently most successful method for these
problems is to project out the weight variables in a combinatorial Bender’s fashion, see
e.g. [50]. This approach is outlined here and further developed in Part III of the thesis. For
an outline of a computational scheme, see Section 13.4 and Figure 13.1. (We recommend
looking at this section after reading this chapter.)

The context of the rest of this thesis is that the constraints in model (2.20) have been
linearized. A B&C approach relies on the following elements.

1. Find an optimal solution to the LP-relaxation of the linearized model. Denote the
values of the \( y \)-variables by \( y^* \in \mathbb{R}^{A \times L} \).
2. Determine if \( y^* \) violates some SPR feasibility cut.
3. If a violated SPR feasibility cut is found, augment it to the LP-relaxation. Goto 1.
4. If the solution is integral, it is optimal, stop. Otherwise, select a new node/branch
and goto 1.

The part of a solution to model (2.20) that \( y^* \) corresponds to is referred to as a (partial)
tentative routing pattern. In Step 2 of this two-stage approach, we should determine if it
is realizable as shortest paths or not, i.e. if the point \( y^* \) belongs to the feasible routing
pattern polytope, \( \text{conv} \mathcal{Y} \). This problem is handled \textit{approximately} by solving the ISPR
problem in Section 2.2 induced by the (near-) binary part of \( y^* \). More precisely, for each
\( l \in L \), the set of SP-arcs, \( A^l \) are the arcs where \( y^*_a \) are close to 1 and the set of non-SP-
arcs, \( \overline{A}^l \) are the arcs where \( y^*_a \) are close to 0. If the resulting ISPR problem, i.e. model
(2.8), is infeasible, the Farkas ray acting as an infeasibility certificate provides a violated SPR feasibility cut.

The objective of the above approach is to illustrate the fundamental role of the ISPR problem in some BSP problem solution procedures and how it is used to handle arc costs implicitly. Note that it is not necessary (and sometimes not desirable) to remove the arc cost variables from the model. Obviously, there are some BSPs where the outlined method is not applicable in practice. However, the ISPR problem still plays a fundamental role. Indeed, the feasibility cuts it generates are still valid inequalities that strengthen the formulation and the ISPR problem acts as a (heuristic) separation procedure for these inequalities. This illustrates that a characterization of feasibility cuts can be useful for all MILP approaches.

We elaborate on the ISPR problem in the next chapter and in Part II where we also give a characterization of its infeasibility certificates. In Part III, we consider the resulting valid inequalities and how to separate them.
Introduction to Inverse Shortest Path Routing Problems

Exact solution approaches to bilevel shortest path (BSP) problems sometimes require that an inverse shortest path routing (ISPR) subproblem is repeatedly solved. In particular, this is the case when the BSP problem is a routing problem in an IP network. The BSP branch-and-cut approach was briefly outlined in Section 2.5.2. We elaborate on this approach further in Part III. The role of the ISPR problem in this framework is to determine if a routing pattern is realizable as a set of simultaneously shortest path w.r.t. a common set of arc costs.

In this chapter, we first consider the classical inverse shortest path (ISP) problem. Then, we present an ISPR problem as a generalization of an ISP problem where prohibited shortest paths, or arcs, are taken into account. A thorough analysis of ISPR problems is the foundation for deriving strong cutting planes in the branch-and-cut approach to solve BSP problems. Such an analysis is the subject of Part II of this thesis. The induced cutting planes and enhanced separation algorithms are discussed in Part III.

Outline Two formulations of the ISP problem are given in Section 3.1; one formulation is based on paths and the other on the reduced cost optimality conditions for the shortest path problem. The two variants of the ISPR problem earlier considered in the literature are presented in Section 3.2. Finally, some remarks on the ISP and ISPR problems are made in Section 3.3.

3.1 Inverse Shortest Path Problems

The ordinary shortest path problem is to determine the shortest paths given a set of arc costs. Conceptually, the inverse shortest path problem is just the other way around. Given a collection of paths, \( \mathcal{P} \), determine the optimal arc costs such that all paths in \( \mathcal{P} \) become shortest paths w.r.t. these costs. To the best of our knowledge, the ISP problem is the first occurrence of an inverse optimization problem in the combinatorial optimization literature. It was first considered by Burton and Toint in [72] and later also in
Introduction to Inverse Shortest Path Routing Problems

e.g. [5, 71, 136, 223, 227]. The motivation of the ISP problem in [72] is by two practical applications, mathematical traffic modelling and seismic tomography. This lead to their choice of the objective; to minimize the deviation from some ideal arc costs.

Their problem formulation is as follows. Given a digraph, \( G = (N, A) \), a non-negative cost vector, \( \bar{w} \), and a path collection \( \mathcal{P} \), find a minimal modification, w.r.t. the Euclidean norm, of \( \bar{w} \) such that all paths in \( \mathcal{P} \) are shortest paths. To model this problem, some additional notation is required. For a path \( P \in \mathcal{P} \), let \( o^P \) and \( d^P \) be the origin and destination, respectively. Further, let \( \mathcal{P}_{st} \) be the collection of all paths in \( G \) with origin \( s \) and destination \( t \). This yields the convex quadratic programming problem,

\[
\begin{align}
\text{minimize} & \quad \frac{1}{2} \sum_{a \in A} (w_a - \bar{w}_a)^2 \\
\text{subject to} & \quad \sum_{a \in Q} w_a - \sum_{a \in P} w_a \geq 0, \quad Q \in \mathcal{P}_{o^Pd^P}, \quad P \in \mathcal{P}, \\
& \quad w_a \geq 0, \quad a \in A.
\end{align}
\]

Note that the number of constraints is potentially exponential in the size of the graph since paths are enumerated. In practice, it will be necessary to handle this issue by a constraint generation scheme to solve (3.1). Note that the associated separation problem is a polynomially solvable shortest path problem so the ellipsoid method can be used to solve (3.1) in polynomial time, see e.g. [128]. In [72], it is pointed out that it is possible reformulate (3.1) to overcome the problem with exponentially many constraints. The key is to avoid enumerating paths. We present a similar approach in terms of node potentials and the optimality conditions of the shortest path problem, see (2.1) and Theorem 2.1.

### 3.1.1 A Polynomial Model for the Inverse Shortest Path Problem

To obtain an arc based formulation, the collection of required shortest paths, \( \mathcal{P} \), can be transformed into a collection of subgraphs that represent the collection of shortest paths, i.e. a family of SP-graphs, cf. Section 2.2. We illustrate such a transformation.

Let \( L \subseteq N \) be the set of all destinations and denote the set of origins of the OD-pairs with the same destination, \( l \in L \), by \( D^l \), i.e.

\[
L := \bigcup_{P \in \mathcal{P}} \{d^P\}, \quad \text{and} \quad D^l := \{o^P \mid d^P = l, \text{ for some } P \in \mathcal{P}\}.
\]

The essence of our transformation is to map each path set, \( S \subset \mathcal{P} \), to the set of arcs covered by some path in \( S \), i.e. define

\[
\tilde{C}(S) = \{a \mid a \in P \text{ for some } P \in S\}.
\]

Using \( \tilde{C}(S) \), form the SP-arcs of the SP-graph to destination \( l \in L \) by combining all path sets with destination \( l \), i.e. set

\[
A^l = \bigcup_{k \in D^l} \tilde{C}(\mathcal{P}_{kl}), \quad l \in L.
\]
From the optimality conditions in Theorem 2.1 it follows that the reduced costs of all SP-arcs must be zero, and the reduced costs of other arcs must be non-negative. This is modelled by introducing a node potential, \( \pi_l \), for each destination, \( l \in L \). The resulting, polynomial, arc based formulation of ISP becomes

\[
\begin{align*}
& \text{minimize } \frac{1}{2} \sum_{a \in A} (w_a - \tilde{w}_a)^2 \\
& \text{subject to } \\
& \quad w_a + \pi_i^l - \pi_j^l = 0, \quad a := (i, j) \in A^l, \quad l \in L, \quad (3.5a) \\
& \quad w_a + \pi_i^l - \pi_j^l \geq 0, \quad a := (i, j) \in A \setminus A^l, \quad l \in L. \quad (3.5b)
\end{align*}
\]

Since this formulation is polynomial it implies that ISP is solvable in polynomial time by an interior point algorithm.

**Theorem 3.1 ([72])**

The ISP problem is solvable in polynomial time.

It is not clear if it in practice is better to use constraint generation or to directly solve the polynomial model. A third "hybrid" option is to use the cycle basis reformulation technique that we present in Chapter 8. Such a model is presented in [75].

Observe that model (3.5) requires a family of SP-graphs instead of a collection of paths. We elaborate on this difference in Section 3.3 after the ISPR problem has been considered, since our remarks apply to both problems.

### 3.2 Inverse Shortest Path Routing Problems

ISPR problems generalize the ordinary ISP problem discussed above in the sense that a set of prohibited paths, or non-SP-arcs, is taken into account. A new element of ISPR problems is that they involve deciding if a path set, or a family of SP-graphs, implicitly introduce some undesired non-SP arcs as SP-arcs. The primary motivation for studying ISPR problems is that they arise as subproblems in BSP problems. In this framework, it can be crucial that no path that is not specified to be shortest becomes a shortest path since this can yield undesired traffic along such a path.

In an ISPR problem it has to be guaranteed that no undesired path becomes a shortest path, i.e. the arc costs must not induce some non-SP-arc as an SP-arc. Starting from models (3.1) and (3.5), it is straightforward to formulate some variants of ISPR problems, in particular the following. Let \( G = (N, A) \) be a strongly connected digraph and \( P \) a path collection; we must decide if there exists some set of arc costs such that all paths in \( P \) are shortest paths and for all \( P' \) \( \in \mathcal{P} \), no unspecified shortest path with the same origin and destination as \( P \) is a shortest path, i.e. every path \( Q \in \mathcal{P}_{or,dr} \setminus \mathcal{P} \) is longer than \( P \). Also recall that we require that all arc costs are at least one. Models essentially equivalent to the following feasibility model are presented in the literature, e.g. in [25, 45, 46, 62].
find \( w \)
subject to
\[
\sum_{a \in Q} w_a - \sum_{a \in P} w_a = 0, \quad P, Q \in \mathcal{P}, \quad o(Q) = o^p, \quad d(Q) = d^p, \quad (3.6a)
\]
\[
\sum_{a \in Q} w_a - \sum_{a \in P} w_a \geq 1, \quad P \in \mathcal{P}, \quad Q \in \mathcal{P}_{o^p,d^p} \setminus \{P\}, \quad (3.6b)
\]
\[
w_a \geq 1, \quad a \in A. \quad (3.6c)
\]

Note that in the case of unique shortest paths considered in [25, 45, 46], constraint \((3.6a)\) is redundant since \( P = Q \), i.e. there can only be one path in \( \mathcal{P} \) that has \( o^p \) as origin and \( d^p \) as destination.

In the shortest path routing context, it is required that all costs are integral. This is not a problem when feasibility is concerned since a non-integral solution can be scaled and rounded. However, for some relevant objectives, the integrality requirement makes the problem hard. In particular, it is shown in [45, 46] that the following problem is APX-hard.

\[
\text{minimize } z
\]
subject to
\[
(3.6a), (3.6b), (3.6c),
\]
\[
w_a \leq z, \quad a \in A, \quad (3.7a)
\]
\[
w_a \in \mathbb{Z}, \quad a \in A. \quad (3.7b)
\]

In the sequel, we only consider feasibility of ISPR problems. This is motivated by the observation that infeasibility of ISPR problems due to upper bounds on arc costs is of no practical concern, see [46] and the LP-rounding procedure in [25].

Model \((3.6)\) suffers from the same drawbacks as its ISP problem counterpart, \((3.1)\). However, since the separation problem is polynomially solvable, so is the feasibility problem, see [25]. Alternatively, this result can be proved by a reformulation strategy similar to the one used for the ISP problem.

### 3.2.1 A Polynomial Model for the Inverse Shortest Path Routing Problem

Recall that the key to a polynomial formulation is to use arcs instead of paths. We use the same transformation as above to obtain arc sets from path collections. An important addition in the ISPR case is the presence of non-SP-arcs, i.e. arcs prohibited to be on shortest paths. Recall that for each destination, \( l \in L \), an SP-graph is defined by the arc subset pair, \((A^l, \bar{A}^l) \subset A \times A\), where \( A^l \) are SP-arcs and \( \bar{A}^l \) are non-shortest path arcs. An SP-graph, \((A^l, \bar{A}^l)\), is obtained from a path collection as follows.

Use \( C_l \), defined in \((3.3)\), to form the SP-arcs to destination \( l \in L \) by combining all path sets with destination \( l \), i.e. set
3.2 Inverse Shortest Path Routing Problems

\[ A^l = \bigcup_{k \in D^l} \tilde{C}(\mathcal{P}_{kl}), \quad l \in L. \]  

(3.8)

Note that \( A^l \) is an ingraph rooted at \( l \), but not necessarily spanning. If \( A^l \) contains nodes with outdegree larger than 1, then it may induce more paths as shortest paths than specified by \( \mathcal{P} \), i.e. \( A^l \) and \( \mathcal{P} \) are not consistent. When this occurs, the ISPR problem is infeasible. As an example, the paths 1 → 2 → 3 → 4 → 5 and 1 → 3 → 5 induce the SP-arcs \( (1, 2), (1, 3), (2, 3), (3, 4), (3, 5), (4, 5) \) which in turn induce four, and not two, shortest paths from node 1 to 5. In the case of unique shortest paths, this is of no concern since the ingraph is an arborescence.

To make sure that no unspecified path becomes a shortest path, the set of non-SP-arcs to destination \( l \in L \) are defined in terms of the SP-arcs. If \( a, a \notin A^l \) but \( a \notin A^l \), then arc \( a \) must be a non-SP-arc, since it would otherwise induce some prohibited path via node \( i \) to destination \( l \) as a shortest path, cf. Proposition 4.19 in [46]. We define

\[ \tilde{A}^l = \bigcup_{\delta^+(i) \cap A^l \neq \emptyset} \left( \delta^+(i) \setminus A^l \right), \quad l \in L. \]  

(3.9)

Extending the reasoning from the ISP problem, Theorem 2.1 implies that the reduced costs must be zero for SP-arcs, at least one for non-SP-arcs and non-negative for all other arcs. Using a node potential, \( \pi^l \), for each destination, \( l \in L \), the resulting, polynomial, ISPR formulation becomes,

\[ \text{find } w \]

subject to

\[ w_a + \pi^l_i - \pi^l_j = 0, \quad a := (i, j) \in A^l, \quad l \in L, \]  

(3.10a)

\[ w_a + \pi^l_i - \pi^l_j \geq 1, \quad a := (i, j) \in \tilde{A}^l, \quad l \in L, \]  

(3.10b)

\[ w_a + \pi^l_i - \pi^l_j \geq 0, \quad a := (i, j) \in A \setminus (A^l \cup \tilde{A}^l), \quad l \in L, \]  

(3.10c)

\[ w_a \geq 1, \quad a := (i, j) \in \tilde{A}^l. \]  

(3.10d)

When \( A^l \) and \( \mathcal{P} \) are consistent it follows that models (3.6) and (3.10) are equivalent, see e.g. [46, 73].

An important special case of the above ISPR problem is when all SP-graphs are spanning, i.e. for all \( l \in L \) the graph induced by the SP-arcs spans all nodes in \( G \) and also \( A = A^l \cup \tilde{A}^l \). This yields the simplified model below, cf. [61].

\[ \text{find } w \]

subject to

\[ w_a + \pi^l_i - \pi^l_j = 0, \quad a := (i, j) \in A^l, \quad l \in L, \]  

(3.11a)

\[ w_a + \pi^l_i - \pi^l_j \geq 1, \quad a := (i, j) \in \tilde{A}^l, \quad l \in L, \]  

(3.11b)

\[ w_a \geq 1, \quad a := (i, j) \in A. \]  

(3.11c)

We conclude this chapter by some general remarks on ISPR problems.
3.3 Remarks on Inverse Shortest Path Routing

We claim that the basic variant of the ISPR problem considered in this chapter is not general enough to be practically relevant in some BSP problem solution schemes. The important problem is to decide if a family of SP-graphs, \( A^L := \{ (A^l, A^l) : l \in L \} \) is realizable, i.e. if there exists \( w \in \mathbb{Z}^A \), \( w \geq 1 \) so that for all \( l \in L \) each SP-arc, \( a \in A^l \), is in some shortest path to \( l \) and no non-SP-arc, \( a \in A^l \), is in any shortest path to \( l \). Realizability can be expressed in terms of path sets as well, but we omit this since the SP-graph approach is more intuitive and because of the drawbacks with the path formulation discussed below. Indeed, realizability takes all branching decisions into account and is a necessary and sufficient condition for not pruning a node in an enumeration tree, whereas the basic variant of the ISPR problem in a sense neglects some non-SP-arc decisions and only gives a sufficient condition for pruning.

The complexity issue of ISPR problems is a bit intricate. The basic variant of the ISPR problem is solvable in polynomial time as an LP. The main result in Chapter 4 is that the general ISPR problem, i.e. realizability, is NP-complete. This is somewhat counter-intuitive. At first it may seem as (3.10) is a correct formulation of realizability; this is even claimed in the recent survey \[50\], i.e. it seems to be the general belief.

The reason that (3.10) is not a correct formulation of realizability for arbitrary families of SP-graphs is that it only takes dual feasibility into account and partly neglects primal feasibility and complementary slackness, see further Chapter 4. However, (3.10) is of course a quite reasonable relaxation of realizability.

As a matter of fact, the hardness issue seems to be of no practical concern. It is non-trivial to construct families of SP-graphs that are not realizable and yield feasible solutions to (3.10). In the branch-and-cut setting, the necessary condition that (3.10) is feasible seems to be good enough in practice, i.e. it produces good enough valid inequalities and prunes infeasible branches well enough.

Let us also consider which relaxation of realizability is better suited; an appropriate adaption of the path based model, (3.6), or the arc based model (3.10).

The path based formulation, (3.6), is intimately related to a column generation approach to solve a master problem. It is in theory possible to branch on paths in the master and forbid specific paths in (3.6). However, this approach is very likely doomed. Indeed, the enumeration tree becomes very poorly balanced and the pricing problem loses its structure. These are crucial issues in branch-and-price-(and-cut) schemes, see e.g. \[166\]. Note that branching on the so called original variables is equivalent to putting an arc in \( A^l \) or \( A^l \) which balances the tree and does not affect the pricing problem structure. We elaborate this in Chapter 11.

We believe that the above is a strong argument in favor of arc based formulations. In the remainder of this thesis we will not focus on path based versions of ISPR problems. We will however consider valid inequalities based on path variables, but these can be derived and analyzed via the corresponding arc based formulations.

Finally, we also make some comments on the use of SP-graphs. First, observe that SP-graphs are more general than path sets. From a practical point of view they are also more convenient. It is in our opinion much better to present input and output using SP-graphs; who would prefer a list of paths as output from Dijkstra’s algorithm rather than a shortest path tree?
3.3 Remarks on Inverse Shortest Path Routing

Figure 3.1: An instance where an SP-graph have multiple origins and destinations and the ISPR model incorrectly indicates infeasibility. Observe that the arc costs in the figure induce the desired shortest paths.

A more general notion of SP-graphs, i.e. the SP-graphs are not necessarily ingraphs, is used in [61]. There it is possible for SP-graphs to have multiple origins and destinations. This flexibility may be advantageous in some contexts. However, it comes at a price; the models derived above are not always applicable. Consider in particular the two SP-graphs in Figure 3.1. According to model (3.10) or (3.11), these SP-graphs are not simultaneously realizable. However, this is obviously wrong since the arc costs in the graph yields the paths induced by the SP-graphs as shortest paths. To avoid this erroneous outcomes it is required that an SP-graph with multiple origins and destinations have a "pivot" node. Then, each path from an origin to a destination must go via the pivot node.

In the setting of these more general SP-graphs it is actually non-trivial to derive a polynomial model. An exponential formulation is presented in [62] along with combination strategies to reduce the problem size. An approach based on an extended formulation yields a polynomial model in [73].

In this thesis, we will only use ingraphs as SP-graphs. Observe that this gives us the additional information that from every node there is a path to the root. This can be exploited both in the modelling of ISPR problems and when valid inequalities are derived.
3 Introduction to Inverse Shortest Path Routing Problems
Part II

Inverse Shortest Path Routing
Complexity of Realizability

Branch-and-cut approaches to bilevel shortest path (BSP) problems require repeated solution of inverse shortest path routing (ISPR) subproblems. In principle, or ideally, it has to be decided at every node in the enumeration tree if it is possible to augment a partial tentative routing pattern to a complete routing pattern. This variant of the ISPR problem is referred to as realizability. When a partial routing pattern is not realizable, valid inequalities that prohibit routing conflicts are augmented to the incomplete formulation. From the above it is clear that a very fundamental issue for the solution of some BSP problems is to solve the realizability problem. Our main result in this chapter is that this problem is NP-complete.

Our hardness result implies that relaxations of realizability should also be considered, e.g. the version of ISPR occurring in the literature, referred to as compatibility, is a very reasonable relaxation. Another fundamental issue is to derive valid inequalities from routing patterns that are not realizable. The latter is in focus in Part III of this thesis.

Outline
We give a short background and motivation for ISPR problems in Section 4.1. The inadequate compatibility formulation of the ISPR problem from the literature is presented in Section 4.2 along with a model for the realizability variant of the ISPR problem. In Section 4.3, the main result that realizability is indeed NP-complete is proved. Finally, we conclude in Section 4.4.

4.1 Background

Let $G = (N, A)$ be a strongly connected digraph and $L \subseteq N$ a set of destinations. For a destination, $l \in L$, a (tentative) routing pattern to $l$ is represented by a shortest path graph (SP-graph), defined by an arc subset pair, $(A^l, \bar{A}^l) \subseteq A \times A$. Arcs in $A^l$ are required to be shortest path arcs (SP-arcs) and arcs in $\bar{A}^l$ are non-shortest path arcs (non-SP-arcs), i.e. prohibited to be on shortest paths.
In our setting, tentative routing patterns arise from intermediate solutions to model (2.20) on page 18 in a branch-and-cut approach. The SP-arcs and non-SP-arcs correspond to the shortest path indicator variables. If $y^L_a = 1$, then $a$ is an SP-arc, i.e. it is in $A^L$; analogously if $y^L_a = 0$, then $a$ is a non-SP-arc and belongs to $\bar{A}^L$. For fractional $y$, it has yet to be decided for some arcs if they are SP-arcs or non-SP-arcs. This implies that the induced SP-graphs may be non-spanning and disconnected since the non-binary part of $y$ is, in a sense, ignored.

A family of SP-graphs, $\{(A^L, \bar{A}^L) : l \in L\}$ is realizable if there exists a strictly positive weight vector, $w \in \mathbb{R}^A_{++}$, such that all SP-arcs in all SP-graphs are in some shortest path to their respective destinations and no non-SP-arc is in a shortest path to its destination. An ISPR problem closely related to realizability is compatibility. A family of SP-graphs, $\{(A^L, \bar{A}^L) : l \in L\}$ is compatible if there is a weight vector, $w \in \mathbb{R}^A_{++}$, such that for each $l \in L$ there is a node potential, $\pi^l \in \mathbb{R}^N$, such that the implied reduced costs are compatible with $(A^L, \bar{A}^L)$, i.e.

$$w_a + \pi^L_a - \pi^L_b \begin{cases} = 0, & \text{if } a := (i, j) \in A^L, \\ > 0, & \text{if } a := (i, j) \in \bar{A}^L, \\ \geq 0, & \text{otherwise.} \end{cases} \quad (4.1)$$

The realizability and compatibility problems are formulated as follows.

**The Realizability Problem.**

Given a digraph, $G = (N, A)$, a set of destinations, $L \subseteq N$, and a family of SP-graphs, $\{(A^l, \bar{A}^l) : l \in L\}$, decide if $\{(A^l, \bar{A}^l) : l \in L\}$ is realizable.

**The Compatibility Problem.**

Given a digraph, $G = (N, A)$, a set of destinations, $L \subseteq N$, and a family of SP-graphs, $\{(A^l, \bar{A}^l) : l \in L\}$, decide if $\{(A^l, \bar{A}^l) : l \in L\}$ is compatible.

Realizability can be put in terms of a tentative routing pattern, $\hat{y}$, that stems from an intermediate solutions to model (2.20). The family of SP-graphs induced by $\hat{y}$ is realizable if there exists a $\tilde{y} \in \mathbb{B}^A \times L$ that induces a compatible family of SP-graphs and for all $a \in A$ and $l \in L$ where $\hat{y}^L_a \in \mathbb{B}$ it holds that $\hat{y}^L_a = \tilde{y}^L_a$. This implies that the inequality

$$\sum_{a, l: \hat{y}^L_a = 0} y^L_a + \sum_{a, l: \hat{y}^L_a = 1} (1 - y^L_a) \geq 1 \quad (4.2)$$

is valid for (2.20) and cuts off $\hat{y}$ when $\tilde{y}$ induces a family of SP-graphs that is not realizable. Inequality (4.2) is valid also for families of SP-graphs that are not compatible.

We will show that the realizability problem is NP-complete, contrary to what has been claimed earlier in the literature, see e.g. [50] (§8.3.2). The compatibility problem has received much attention in the OSPF literature, see e.g. [25, 46, 50, 61, 62, 58, 73, 191, 193]. It is polynomially solvable as an LP.

A common misconception is actually that compatibility is equivalent to realizability. We provide a counter example to this in the next section. There we also give mathematical models for both problems and compare them to gain some insight into the weakness of the compatibility variant of the ISPR problem.
4.2 Models for Realizability and Compatibility

Remark 4.1. Observe that ISPR problems, in particular realizability and compatibility, are significantly different from the ordinary inverse shortest path (ISP) problem in [72], see Section 3.1. In particular, in ISPR problems a partial desired optimal solution is specified via SP-arcs and non-SP-arcs and we must determine if it can be completed to an optimal solution without implicitly introducing some undesired non-SP-arcs. The introduction of prohibited arcs is actually what makes ISPR problems hard. Also, the actual weight vector in most ISPR problems is irrelevant, while many ISP problems involve the minimization of the deviation from some ideal weight vector. A counterpart for an OSPF variant of the ISPR problem would be to find small integral weights, i.e. a small deviation from zero. This problem is shown to be APX-hard for the infinity norm and some other relevant measures in [45, 46]. A related ISPR problem is the inverse subproblem in tariff revenue optimization, see e.g. [64].

4.2 Models for Realizability and Compatibility

We first present a model for compatibility, widely available in the literature. Based on this, a model for realizability is given.

Let \( A^L = \{ (A_l^l, \bar{A}^l) : l \in L \} \) be a family of SP-graphs for a given destination set, \( L \). The below model follows straightforward from the definition of compatibility. Let \( w \in \mathbb{R}^A_+ \) be arc weight variables and for each destination, \( l \in L \), let \( \pi^l \in \mathbb{R}^N \) be node potential variables associated with \( l \). Restricting the reduced costs according to (4.1) yields the problem to

\[
\begin{align*}
\text{find} \quad & w \\
\text{subject to} \quad & w_a + \pi^l_i - \pi^l_j = 0, \quad a := (i, j) \in A^l, \ l \in L, \tag{4.3a} \\
& w_a + \pi^l_i - \pi^l_j \geq 1, \quad a := (i, j) \in \bar{A}^l, \ l \in L, \tag{4.3b} \\
& w_a + \pi^l_i - \pi^l_j \geq 0, \quad a := (i, j) \in A \setminus (A^l \cup \bar{A}^l), \ l \in L, \tag{4.3c} \\
& w_a \geq 1, \quad a \in A. \tag{4.3d}
\end{align*}
\]

Observe that the reduced cost constraints in (4.1) defines an open polyhedral cone. By a simple scaling argument it follows that the cone is non-empty if and only if the polyhedron defined by (4.3) is non-empty.

Intuitively, \( A^L \) is not compatible if a subset of SP-arcs and non-SP-arcs directly induce a reduced cost routing conflict. Two particular routing conflicts used in our NP-completeness proof are given in Example 4.1. A less complicated conflict was given in Example 1.1. More examples are given throughout the thesis, in particular in Chapter 6.

Let us consider the realizability problem. Note that realizability is equivalent to finding a weight vector that induces a compatible family of spanning SP-graphs. Therefore, compatibility is a necessary condition for realizability.

Proposition 4.1
If a family of SP-graphs, \( A^L \), is realizable then it is compatible.
Compatibility does not in general imply realizability. A compatible family of SP-graphs is realizable if it can be completed to a spanning family of SP-graphs. To derive a model that can be used to decide if a family of SP-graphs is realizable, the following observation is crucial.

**Proposition 4.2**
A family of SP-graphs, $A^L$, is realizable if and only if there is a weight vector $w$ and tight node potentials, $\{\pi^l : l \in L\}$, induced by $w$ that are feasible in (4.3).

This proposition yields the straightforward modelling approach to force the existence of (reverse) spanning arborescences rooted at each destination, $l \in L$, where all reduced costs are zero, since this yields tight node potentials.

Since SP-arcs and non-SP-arcs correspond to the shortest path indicator variables, $y$, used above, these variables are reused for this purpose, i.e. $y^l_a = 1$ if $a$ is an SP-arc and $y^l_a = 0$ otherwise. To decide if a family of SP-graphs is realizable it suffices to solve the following bilinear integer program,

\[
\text{find } w \\
\text{subject to } \\
w_a + \pi^i_a - \pi^j_a + y^l_a \geq 1, \quad a := (i,j) \in A, \; l \in L, \quad (4.4a) \\
(w_a + \pi^i_a - \pi^j_a)y^l_a = 0, \quad a := (i,j) \in A, \; l \in L, \quad (4.4b) \\
\sum_{a \in \delta^+(i)} y^l_a \geq 1, \quad i \in N \setminus \{l\}, \; l \in L, \quad (4.4c) \\
y^l_a = 1, \quad a \in A^l, \; l \in L, \quad (4.4d) \\
y^l_a = 0, \quad a \in \bar{A}^l, \; l \in L, \quad (4.4e) \\
y^l_a \in \mathbb{Z}, \quad a \in A, \; l \in L, \quad (4.4f) \\
w_a \geq 1, \quad a \in A. \quad (4.4g)
\]

The correctness of the model is motivated by the following observations. First, since all weights are strictly positive, the $y^l$-variables can not induce a directed cycle. Because of this, the outdegree constraints imply that the $y^l$-variables induce an ingraph that contains an arborescence rooted at $l$ for each $l \in L$. In terms of the shortest path problem, this corresponds to primal feasibility. Dual feasibility and complementary slackness is ensured by constraints (4.4a) and (4.4b). Alternatively, correctness can be motivated in view of Proposition 4.2; the existence of arborescences implies that node potentials are tight, since the compatibility part of the model is satisfied, the result follows. Observe that realizability in the case of unique shortest paths is modelled by replacing the inequality in (4.4c) by equality.

**Theorem 4.1**
A family of partial SP-graphs, $A^L$, is realizable if and only if (4.4) has a feasible solution.

It is possible to derive an integer linear formulation of (4.4) by utilizing the valid inequalities in Part III of the thesis. We omit this. Note that it is not necessarily possible to bound the reduced costs from below and above, hence a standard big-M linearization may not be feasible.
The major difference between compatibility and realizability is that compatibility only takes dual feasibility of the shortest path problem into account, but partly neglects primal feasibility and complementary slackness. See models (4.3) and (4.4). This issue becomes apparent in our counter example to the equivalence of realizability and compatibility, i.e. Example 4.2.

4.2.1 Incompatible and Unrealizable SP-graphs

In this section we give three examples of (potential) routing conflicts. The first two are required in the complexity proof in Section 4.3 and the third is our counter example to the equivalence of compatibility and realizability. The first conflict is referred to as a valid cycle in [61, 62], see also Section 7.1.

--- Example 4.1 ---

Consider the SP-graph family, $A^L$, with destinations $a, b \in L$ drawn in Figure 4.1. Suppose that there are two (start) nodes, $s_1$ and $s_2$, and two (end) nodes, $e_3$ and $e_4$, that form the following SP-arc patterns,

$$A_a = \{(s_1, e_3), (s_2, e_4)\} \quad \text{and} \quad A_b = \{(s_1, e_4), (s_2, e_3)\}.$$  \hfill (4.5)

In all feasible solutions to Model (4.3) (and (4.4)) it must hold that

$$w_{13} + \pi_1^b - \pi_3^b = w_{23} + \pi_2^a - \pi_3^a = w_{14} + \pi_1^a - \pi_4^a = w_{24} + \pi_2^b - \pi_4^b = 0.$$  \hfill (4.6)

This is direct from the surrogate constraint composed of the corresponding reduced cost constraints,

$$0 = w_{13} + \pi_1^b - \pi_3^b + w_{23} + \pi_2^a - \pi_3^a + w_{14} + \pi_1^a - \pi_4^a + w_{24} + \pi_2^b - \pi_4^b =$$
$$= (w_{13} + \pi_1^b - \pi_3^b + w_{23} + \pi_2^a - \pi_3^a + w_{14} + \pi_1^a - \pi_4^a + w_{24} + \pi_2^b - \pi_4^b) =$$
$$\geq 0 + 0 + 0 - (0 + 0 + 0 + 0) = 0.$$  \hfill (4.7)

Hence, the arcs $(s_1, e_3)$ and $(s_2, e_4)$ are induced SP-arcs to destination $b$ and the arcs $(s_1, e_4)$ and $(s_2, e_3)$ are induced SP-arcs to destination $a$. 

--- Figure 4.1: The example setting is illustrated on the left and the induced SP-arcs are drawn on the right. The SP-arcs to destinations $a,b \in L$, are drawn with solid and dashed arrows, respectively. In the right part, the upper arcs of parallel arc pairs are the original SP-arcs and the lower are the induced arcs. ---
**Figure 4.2:** The example setting is illustrated on the left and the induced SP-arcs are drawn on the right. The SP-arcs to destination \( a, b \) and \( c \) are drawn with solid, dashed and dotted arrows, respectively. In the right part, the upper arcs of parallel arc pairs are the original SP-arcs and the lower are the induced arcs.

A similar, but slightly more complicated example is illustrated in Figure 4.2. Here, the SP-graph family, \( A_L \), with destinations \( a, b, c \) contains the following SP-arc patterns:

\[
A_a = \{(s_1, e_3), (s_2, e_5)\}, \quad A_b = \{(s_1, e_4), (s_2, e_3)\}, \quad A_c = \{(s_1, e_5), (s_2, e_4)\}. \quad (4.8)
\]

A surrogate constraint argument again shows that all feasible solutions to (4.3) satisfy

\[
\begin{align*}
  w_{13} + \pi_1^b - \pi_3^b &= w_{14} + \pi_1^c - \pi_4^c = w_{15} + \pi_1^a - \pi_5^a = \\
  w_{23} + \pi_2^a - \pi_3^a &= w_{24} + \pi_2^b - \pi_4^b = w_{25} + \pi_2^c - \pi_5^c = 0. \quad (4.9)
\end{align*}
\]

Hence, the arcs \((s_1, e_5)\) and \((s_2, e_3)\) are induced SP-arcs to destination \( a \), the arcs \((s_1, e_3)\) and \((s_2, e_4)\) are induced SP-arcs to destination \( b \) and the arcs \((s_1, e_4)\) and \((s_2, e_5)\) are induced SP-arcs to destination \( c \).

We conclude the section with an example of a family of SP-graphs that is compatible but not realizable. Hence, the inadequacy of model (4.3) and the need for model (4.4) is well motivated.

---

**Example 4.2**

**Figure 4.3:** A graph and four SP-graphs that cannot be realized. All dashed arcs are SP-arcs to destination 0. The dotted arcs emanating from node 1, 2 and 3, respectively, are SP-arcs to destinations \( a, b \) and \( c \), respectively.

Consider the graph in Figure 4.3 and the SP-graphs with arc sets
4.3 Complexity of ISPR Problems

This family of partial SP-graphs is not realizable. Note that the dashed path starting at node 1 must be augmented to a path that ends in node 0. Any augmentation of the path implies that it goes via node 1’, 2’ or 3’. Therefore, the shortest path subpath optimality property implies that there are two disjoint shortest paths from 1 to 1’, 2 to 2’ or 3 to 3’. This is however not feasible w.r.t. the non-SP-arcs.

Now consider compatibility. Setting the cost on arcs entering node a, b and c to a large number and the costs on all other arcs to 1 yields a feasible solution to (4.3). Hence, the instance is compatible but not realizable.

Note that the node potentials in the "feasible" solution are not tight. Also, the tight node potentials are not feasible w.r.t. the reduced cost constraints. It is straightforward to generalize this example to one with arbitrarily many nodes and the same property.

4.3 Complexity of ISPR Problems

In this section we focus on the complexity of ISPR. From the literature and model (4.3) above it is clear that compatibility can be decided in polynomial time by solving a linear (feasibility) program. The complexity of the problem to decide if a family of partial SP-graphs is realizable has until recently been open.

We prove that it is NP-complete to decide if a family of SP-graphs is realizable by a polynomial reduction to the exclusive 1 in 3 Boolean satisfiability (X3SAT) problem. Recall that the X3SAT problem is to decide if a formula (in conjunctive normal form) where each clause contains three literals has a truth assignment that makes exactly one literal in each clause is true. Canonical X3SAT instances only contain sorted clauses where all variables are positive and different and no two clauses share more than one variable.

Definition 4.1
An X3SAT instance, $I = (X, \mathcal{C})$, with the variable set $X = \{x_1, \ldots, x_n\}$ and clause collection $\mathcal{C} = \{C_1, \ldots, C_m\}$ is canonical if

- For each clause $C = (x_i \lor x_j \lor x_k) \in \mathcal{C}$ it holds that $i < j < k$.
- No pair of variables is included in two or more clauses.

It is actually no restriction to only consider canonical X3SAT instances. Indeed, if there are clauses $(a \lor b \lor c)$ and $(a \lor b \lor d)$, then $c = d$ in all feasible truth assignments and one variable and (at least) one clause can be dropped. Since X3SAT is NP-complete, applying this argument recursively yields the following result.
Theorem 4.2
It is NP-complete to decide if a canonical X3SAT instance is satisfiable.

The reduction in our proof requires the use of a next operator. Informally, it yields the modulo-wise next variable in a clause.

Definition 4.2
If \( C = (x_i \lor x_j \lor x_k) \in \mathcal{C} \) is a clause in a canonical X3SAT instance, \( I = (X, \mathcal{C}) \), then the next operator, \( n : X \times \mathcal{C} \rightarrow X \), is defined by

\[
n(x, C) = \begin{cases} 
  x_j, & \text{if } x = x_i, \\
  x_k, & \text{if } x = x_j, \\
  x_i, & \text{if } x = x_k. 
\end{cases}
\] (4.11)

Example 4.3
The X3SAT instance with variable set \( \{a, b, c, d, e, f, g\} \)
and clause collection

\[\{C_1, C_2, C_3, C_4\} = \{a \lor b \lor c, a \lor d \lor e, a \lor f \lor g, b \lor d \lor f\}\]
is in canonical form.

It has three feasible assignments (set \( \{b, e, g\} \) to true, \( \{c, d, g\} \) to true, or \( \{c, d, g\} \) to true). Note that the instance remains in canonical form if the clause \( (c \lor e \lor f) \) is added, but not if the clause \( (b \lor e \lor f) \) is added since \( C_4 \) already contains both \( b \) and \( f \). The next operator takes the following values for clause \( C_1 \): \( n(a, C_1) = b \), \( n(b, C_1) = c \) and \( n(c, C_1) = a \).

Before a formal reduction to X3SAT from realizability is given we briefly describe the idea behind the graph that is used in the realizability problem. The graph obtained from clause \( ABC \) is given in Figure 4.4. For each clause, \( ABC \) say, a node \( ABC \) is created with the purpose to force at least one variable in the clause to be true. Then, three auxiliary nodes \( AB, AC \) and \( BC \) are created such that it is guaranteed that at most one of the variables in each pair (hence, also in the clause) is true. To accomplish these tasks, potential routing conflicts of the kind in Example 4.1 are created with the SP-arcs and the non-SP-arcs. The actual truth assignment is determined by arcs from an auxiliary starting node to auxiliary variable nodes.

We may now formally describe how to construct a realizability instance from a canonical X3SAT instance that is feasible if and only if the X3SAT instance is.

Given a canonical X3SAT instance, \( I = (X, \mathcal{C}) \), the graph \( G(I) = (N, A) \) and the family of SP-graphs, \( A(I) \), is created as follows.

For each variable \( x \in X \), create three nodes in \( G \): \( x^+, x^- \) and \( x \). For each clause \( C \in \mathcal{C} \), create four nodes \( C_{ij}, C_{ik}, C_{jk} \) and \( C \). Also add an auxiliary starting node, \( S \).

To determine the set of arcs consider each variable \( x \in X \) and add the arcs
4.3 Complexity of ISPR Problems

Figure 4.4: The SP-graphs associated with variables $A, B$ and $C$ in a realizability instance obtained from a canonical X3SAT instance with clause $ABC$. The solid, dashed and dotted arcs are required to be SP-arcs for destination $A$, $B$ and $C$, respectively. Note that appropriate destination assignments to arc $(S, A^-)$, $(S, B^-)$ and $(S, C^-)$ can be used to create the situation in Figure 4.2 in Example 4.1. Similarly, arc $(S, A^+)$, $(S, B^+)$ and $(S, C^+)$ can be used to create the situation in Figure 4.1 in Example 4.1.

\[(S, x^+), (x^+, x), (S, x^-), (x^-, x), (S, x^-)(C_{ij}, x^+), (C_{ik}, x^+), (C_{jk}, x^+), \]
\[(x^+, y), (x^-, y), (x, y), \]

where $y \neq x$ is also a variable and $C = (x_i \lor x_j \lor x_k) \in C$ a clause that contains $x$.

It remains to construct the family of SP-graphs. For each variable, $x \in X$, form an SP-graph to the node $l = x$ with SP-arcs, $A^l$, non-SP-arcs $A^l$ and unrestricted arcs $U^l = A^l \setminus (A^l \cup A^l)$ determined according to the rules below.

1. Add the arcs $(x^+, x)$ and $(x^-, x)$ to $A^l$ as SP-arcs. Also add the arc $(y, x)$ to $A^l$ for each variable $y \neq x$.

2. Add the arcs $(S, x^+)$ and $(S, x^-)$ to $U^l$.

3. Add all arcs emanating from $S$ except $(S, x^+)$ and $(S, x^-)$ to $A^l$ as non-SP-arcs.

4. For each clause $C$, let $y = n(x, C)$, then add the arcs $(C, y^-)$ and $(y^-, x)$ to $A^l$.

For each clause $C = (x_i \lor x_j \lor x_k)$ add arcs as SP-arcs to the associated SP-graphs according to the following rules.

5. Add $(C_{ij}, x^+_j)$ and $(x^+_j, x_i)$ to $A^l$ if $l = x_i$ and $(C_{ij}, x^+_i)$ and $(x^+_i, x_j)$ to $A^l$ if $l = x_j$.

6. Add $(C_{ik}, x^+_k)$ and $(x^+_k, x_i)$ to $A^l$ if $l = x_i$ and $(C_{ik}, x^+_i)$ and $(x^+_i, x_k)$ to $A^l$ if $l = x_k$. 

\[(S, x^+), (x^+, x), (S, x^-), (x^-, x), (S, x^-)(C_{ij}, x^+), (C_{ik}, x^+), (C_{jk}, x^+), \]
7. Add \((C_{jk}, x_k^+)\) and \((x_k^+, x_j)\) to \(A^l\) if \(l = x_j\) and \((C_{jk}, x_j^+)\) and \((x_j^+, x_k)\) to \(A^l\) if \(l = x_k\).

8. Arcs not put in \(A^l\) or \(\bar{A}^l\) due to one of the rules above is put in \(U^l\).

Note that the SP-arcs for destination \(l = x\) induce a tree that spans all nodes in \(G\) associated with a variable or clause that is connected to \(x\) via some clause. Also, \(S\) is not contained in any such tree and all its emanating arcs are in \(U^l\) or \(\bar{A}^l\).

These rules applied to the single clause \((A \lor B \lor C)\) gives the result in Figure 4.4.

**Theorem 4.3**

**Given a canonical X3SAT instance,** \(I = (X, C)\), let \(G(I) = (N, A)\) and \(A^l \cup U^l \cup \bar{A}^l\) be constructed from rules 1-8 above for each variable \(x^l \in X\). Denote the induced family of SP-graphs by \(A^L\). Then, the X3SAT instance \(I = (X, C)\) is feasible if \(A^L\) is realizable.

To prove this theorem some lemmas are used.

**Lemma 4.1**

**Given an SP-graph family,** \(A^L\), and a cost vector, \(w\), that verifies that \(A^L\) is realizable, let \(a, b \in L\) be two destinations. If there are nodes \(s_1, s_2, e_3, e_4 \in N\) such that the SP-arc sets satisfy \(I_a \supseteq \{(s_1, e_1)\} \cup \{(s_2, e_4)\}\) and \(I_b \supseteq \{(s_1, e_4)\} \cup \{(s_2, e_3)\}\). Then, the induced SP-arc sets must also satisfy \(I_a \supseteq \{(s_1, e_1)\} \cup \{(s_2, e_3)\}\) and \(I_b \supseteq \{(s_1, e_3)\} \cup \{(s_2, e_4)\}\).

**Lemma 4.2**

**Given an SP-graph family,** \(A^L\), and a cost vector, \(w\), that verifies that \(A^L\) is realizable, let \(a, b, c \in L\) be three destinations. If there are nodes \(s_1, s_2, e_3, e_4, e_5 \in N\) such that the SP-arc sets satisfy \(I_a \supseteq \{(s_1, e_3)\} \cup \{(s_2, e_5)\}\), \(I_b \supseteq \{(s_1, e_4)\} \cup \{(s_2, e_3)\}\), and \(I_c \supseteq \{(s_1, e_5)\} \cup \{(s_2, e_4)\}\). Then, the induced SP-arc sets must also satisfy \(I_a \supseteq \{(s_1, e_3)\} \cup \{(s_2, e_5)\}\), \(I_b \supseteq \{(s_1, e_5)\} \cup \{(s_2, e_3)\}\), \(I_c \supseteq \{(s_1, e_4)\} \cup \{(s_2, e_5)\}\).

These lemmas were exemplified in Example 4.1 above. Their proof follows immediately by the surrogate argument in these examples. The lemmas are used to derive properties of realizable instances obtained from canonical X3SAT instances as described above. The properties are summarized in Lemma 4.3 which is the foundation of the proof of Theorem 4.3.

**Lemma 4.3**

**Let** \(A^L\) **be an SP-graph family induced by a canonical X3SAT instance,** \(I = (X, C)\). **Let** \(w\) **be a cost vector that verifies that** \(A^L\) **is realizable, i.e.,**

\[ A^l \subseteq T^l(w) \quad \text{and} \quad \bar{A}^l \cap T^l(w) = \emptyset, \quad \text{for all} \ l \in L. \]

**Then,** the following properties of \(T^l(w)\) **are satisfied for all** \(l \in L\). **Here, shortest paths, SP-arcs and non-SP-arcs are considered w.r.t.** \(w\), i.e., SP-arcs are in \(T^l(w)\) and non-SP-arcs are not.

1. **For any clause,** \((A \lor B \lor C)\) **say, at least one of the arcs** \((S, A^+)\), \((S, B^+)\) **and** \((S, C^+)\) **is an SP-arc to destination** \(A, B\) **and** \(C\) **respectively.
2. For any clause, \((A \lor B \lor C)\), it holds that: (1) at most one of the arcs \((S, A^+)\) and \((S, B^+)\) is an SP-arc to destination \(A\) and \(B\), respectively; (2) at most one of the arcs \((S, A^+)\) and \((S, C^+)\) is an SP-arc to destination \(A\) and \(C\), respectively; and (3) at most one of the arcs \((S, B^+)\) and \((S, C^+)\) is an SP-arc to destination \(B\) and \(C\), respectively.

3. For any clause, \((A \lor B \lor C)\), exactly one of the arcs \((S, A^+)\), \((S, B^+)\) and \((S, C^+)\) is an SP-arc to destination \(A\), \(B\) and \(C\) respectively.

4. For any variable, \(x\) say, exactly one of the arcs \((S, x^-)\) and \((S, x^+)\) is an SP-arc to destination \(x\).

**Proof:** All properties essentially follows from Lemma 4.1 and Lemma 4.2. By construction, given a variable \(x\), it holds that an arc, \((S, i)\), emanating from the starting node is a non-SP-arc unless \(i\) equals \(x^+\) or \(x^-\). Therefore, at least one of the arcs \((S, x^-)\) and \((S, x^+)\) is an SP-arc to destination \(x\). Using this, we prove Property 1-4.

1. Consider a clause, \((A \lor B \lor C)\) say. Assume that none of the arcs \((S, A^+)\), \((S, B^+)\) and \((S, C^+)\) is an SP-arc to destination \(A\), \(B\) and \(C\), respectively. Then, all the arcs \((S, A^+)\), \((S, B^-)\) and \((S, C^-)\) must be SP-arcs to destination \(A\), \(B\) and \(C\), respectively. Recall that also \((ABC, B^-)\), \((ABC, C^-)\) and \((ABC, A^-)\) are SP-arcs to destination \(A\), \(B\) and \(C\), respectively, by construction. This yields that the requirements in Lemma 4.2 are satisfied with start nodes \(s_1 = S\) and \(s_2 = ABC\) and end nodes \(e_3 = A^-\), \(e_4 = B^-\) and \(e_5 = C^-\). Therefore, (for instance) the arc \((S, A^-)\) is also an SP-arc to destination \(C\) which yields a contradiction.

2. Consider a clause, \((A \lor B \lor C)\) say. Assume that both of the arcs \((S, A^+)\) and \((S, B^+)\) are SP-arcs to destination \(A\) and \(B\), respectively. Recall that also \((AB, B^+)\) and \((AB, A^+)\) are SP-arcs to destination \(A\) and \(B\), respectively, by construction. This yields that the requirements in Lemma 4.1 are satisfied with start nodes \(s_1 = S\) and \(s_2 = AB\) and end nodes \(e_3 = A^+\) and \(e_4 = B^+\). Therefore, (for instance) the arc \((S, A^+)\) is also an SP-arc to destination \(B\) which yields a contradiction. The cases \(AC\) and \(BC\) are proved analogously.

3. Consider a clause, \((A \lor B \lor C)\) say. Combining the three constraints in 2 yields that at most one of the arcs \((S, A^+)\), \((S, B^+)\) and \((S, C^+)\) is an SP-arc to destination \(A\), \(B\) and \(C\) respectively. Since Property 1 states that at least one of the arc is an SP-arc to the respective destination, exactly one SP-arc must be so.

4. Consider a variable \(x\) and a clause \(C = (x \lor y \lor z)\). At least one of the arcs \((S, x^-)\) and \((S, x^+)\) is an SP-arc to destination \(x\). Assume that both are. This yields that \((S, y^-)\) and \((S, z^-)\) are SP-arcs to destinations \(y\) and \(z\), respectively, from Property 2 with \(xy\) and \(xz\). Since \((S, x^-)\) was also assumed to be an SP-arc to destination \(x\) the same situation as in the proof of Property 1 occurs and Lemma 4.2 yields a contradiction.

**Proof of Theorem 4.3:** Given a realizability certificate construct the assignment from the SP-arcs emanating from the starting node as follows. If \((S, x^+)\) is an SP-arc, then set \(x\) to true, otherwise, set \(x\) to false. It now follows from Lemma 4.3 that exactly one variable in each clause is true and the assignment is feasible.
To complete our NP-completeness proof it is required to construct a realizability certificate for a given Boolean assignment that satisfies the X3SAT instance.

**Theorem 4.4**
Given a canonical X3SAT instance, \( I = (X, C) \), let \( G(I) = (N, A) \) and \( A^I \cup U^I \cup \bar{A}^I \) be constructed from rules 1-8 above for each variable \( x^I \in X \). Denote the induced family of SP-graphs by \( A^L \). Then, \( A^L \) is realizable if the X3SAT instance \( I = (X, C) \) is feasible.

**Proof:** It suffices to find a cost vector, \( w(\tilde{X}) \), from a given Boolean assignment, \( \tilde{X} \), that verifies the realizability of \( A^L \) in \( G(I) = (N, A) \). The rules in Table 4.2 and Table 4.1 are used to determine \( w \) from \( \tilde{X} \). Here \( x \) and \( y \) are different variables and \( C = (x \lor y \lor z) \) is a clause. Costs on arcs not covered by a rule below are set to 5.

### Table 4.1: Rules to determine the cost vector for variable and clause arcs from a truth assignment.

<table>
<thead>
<tr>
<th>((i, j))</th>
<th>(x) is true</th>
<th>(y) is true</th>
<th>(z) is true</th>
</tr>
</thead>
<tbody>
<tr>
<td>((C_{xy}, x^+))</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>((C_{xy}, y^+))</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>((C_{xz}, x^+))</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>((C_{xz}, z^+))</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>((C_{yz}, y^+))</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>((C_{yz}, z^+))</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

### Table 4.2: Rules to determine the cost vector for arcs from the auxiliary node \( S \) from a truth assignment.

<table>
<thead>
<tr>
<th>((i, j))</th>
<th>if (x) is true</th>
<th>if (x) is false</th>
</tr>
</thead>
<tbody>
<tr>
<td>((S, x^+))</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>((S, x^-))</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>((x^+, x))</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>((x^-, x))</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>((x, y))</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

Since the X3SAT instance is canonical, there is no variable pair that is in two clauses. This implies that the rules above are unambiguous. The possible source of ambiguity would be for a clause, \( C = (x \lor y \lor z) \), from \( x^- \), \( y^- \), or \( z^- \) to \( x \), \( y \), or \( z \). However, since no other clause can contain two of the variables \( x \), \( y \), or \( z \) this is of no concern. This “independence” property implies that it essentially suffices to consider one clause in isolation.

The cost vector obtained from the rules in Table 4.2 is illustrated in Figure 4.5 for the part of the graph involving the clause \( ABC \) when \( A \) is assigned to true.
From these rules it is straightforward to derive the tight node potential for any destination. Due to the independence there are only three cases to consider for each clause, $C = (x \lor y \lor z)$, depending on which position $x$ has relative to the true variable in $C$. The result is illustrated for the clause $ABC$ and destinations $A$ and $B$ when $A$ is true in Figure 4.6 on page 56. From this it is easy to verify that all required SP-arcs are SP-arcs and that no non-SP-arc is an SP-arc. That is, the family of SP-graphs is realizable.

It follows from Theorems 4.3 and 4.4 that realizability is NP-complete. Note that, by construction of the reduction, realizability is NP-complete also when it is required that all shortest paths are unique.

**Theorem 4.5**

*It is NP-complete to decide if a family of SP-graphs is realizable.*

From the proof above it is clear that a stronger statement holds. Namely, it is NP-complete to decide if a family of SP-graphs is realizable even if for each SP-graph the SP-arcs form a rooted, non-spanning tree with maximum depth 2.

### 4.4 Conclusion

In this chapter we considered inverse shortest path routing problems. In particular, we showed that there is a difference between the previously considered compatibility variant of the problem and the more complete variant, here referred to as realizability. We gave a mathematical programming formulation for the realizability problem. Most importantly, it was proved that the realizability problem is NP-complete. This result has significant theoretical consequences for bilevel programs where the lower level is a shortest path problem.
Figure 4.6: The parts of the shortest path tree to destinations $A$ (top) and $B$ (bottom) that involve nodes associated with the clause $ABC$ which is assumed to be satisfied by $A$. Solid arcs represent SP-arcs and dotted arcs represent non-SP-arcs. The dotted arcs have not been SP-arcs or non-SP-arcs.
SOLVING inverse shortest path routing (ISPR) problems is a crucial element of branch-and-cut (B&C) approaches to bilevel shortest path (BSP) problems. Preferably, the realizability version of the ISPR problem should be solved. Due to the hardness of realizability, the compatibility relaxation is in practice solved instead. In this chapter, we propose a new relaxation that we refer to as partial realizability. It is stronger than compatibility and has structure that can be exploited to develop more efficient solution methods.

Outline We give a short background and restate the ISPR problems from the previous chapter and their models in Section 5.1. The new relaxation, partial realizability, is presented in Section 5.2 where we also introduce generalized SP-graphs and compare realizability, partial realizability and compatibility. Finally, some valid inequalities induced by partial realizability are described in Section 5.3.

5.1 Background

As usual, $G = (N, A)$ is a strongly connected digraph and $L \subseteq N$ a set of destinations. For destination, $l \in L$, the SP-graph $(A^l, \bar{A}^l) \subseteq A \times A$ represents a tentative routing pattern to $l$, i.e. arcs in $A^l$ are SP-arcs and arcs in $\bar{A}^l$ are non-SP arcs. The context is that the tentative routing patterns arise from intermediate solutions to model (2.20) on page 18 in a B&C approach. Hence, the SP-arcs and non-SP-arcs correspond to the shortest path indicator variables, $y \in \mathbb{B}^{A \times L}$, i.e. $y^l_a = 1$ if $a \in A^l$ and $y^l_a = 0$ if $a \in \bar{A}^l$. In general, some values in $y$ will be fractional and the induced SP-graphs are non-spanning and disconnected.

Two variants of the ISPR problem have been considered, compatibility and realizability, see further Chapter 4. In particular, we observed that a shortest path routing (SPR) conflict, $(C, \bar{C})$ where $C, \bar{C} \in A \times L$, arising from a family of SP-graphs that is not realizable implies that the inequality
Relaxations of Realizability

\[
\sum_{(a,l) \in C} (1 - y_a^l) + \sum_{(a,l) \in C} y_a^l \geq 1, \quad (5.1)
\]

is valid for the BSP model (2.20).

There are exponentially many SPR conflicts, hence these inequalities are separated by solving ISPR problems. The intractability of realizability force us to resort to relaxations, such as compatibility. Clearly, stronger relaxations implies that more inequalities of the form (5.1) can be separated. Strong inequalities based on shortest path indicator variables are of utmost importance for efficient solution of some BSP problems.

We propose a new relaxation of realizability referred to as partial realizability. Our relaxation is stronger than compatibility and still solvable in polynomial time. Moreover, from a descriptive point of view, the additional structure in our partial realizability model allows us to generalize the concept of SP-graphs. Using generalized SP-graphs, partial realizability can be interpreted as compatibility of generalized SP-graphs. Generalized SP-graphs yield a unified combinatorial description of a huge class of SPR conflicts, and therefore SPR inequalities. In particular, all valid inequalities in \([218]\) are generalized valid cycle inequalities.

Using generalized SP-graphs we exploit the additional structure to obtain a compact formulation of partial realizability based on cycle bases in Chapter 8. In \([202]\) computational experiments shows that the cycle basis formulation of partial realizability can often be solved more efficiently than compatibility. Hence, our relaxation gives more and stronger SPR inequalities and often allows for more efficient solution.

Here, we compare our new relaxation, partial realizability, to compatibility and illustrate some of its advantages. In particular, the aspect of valid inequalities is considered for subpath based routing conflicts. A more thorough treatment of valid inequalities is carried out in Part III.

### 5.1.1 Inverse Shortest Path Routing Problems

Given a family of SP-graphs, \(A^L = \{ (A^l, \bar{A}^l) \}_{l \in L} \), consider the ISPR problems to decide if \(A^L\) is compatible and realizable, respectively. We restate the models for these problems for convenience.

Recall from Section 4.2 in the previous chapter that a family of SP-graphs, \(A^L\), is compatible if and only if model (4.3) is feasible, i.e. \(A^L\) is compatible if we can

\[
\begin{align*}
\text{find} & \quad w \\
\text{subject to} & \quad [\text{ISPR-C}] \\
& \quad w_a + \pi^l_i - \pi^l_j = 0, \quad a := (i,j) \in A^l, \; l \in L, \\
& \quad w_a + \pi^l_i - \pi^l_j \geq 1, \quad a := (i,j) \in \bar{A}^l, \; l \in L, \\
& \quad w_a + \pi^l_i - \pi^l_j \geq 0, \quad a := (i,j) \in A \setminus (A^l \cup \bar{A}^l), \; l \in L, \\
& \quad w_a \geq 1, \quad a \in A. 
\end{align*}
\]

As pointed out in the previous chapter, a weakness of model (5.2) is that it does not guarantee the existence of reduced cost zero arborescences rooted at the destinations. This
is however a necessary and sufficient condition for the reduced cost modeling approach to be valid. A bilinear integer model, (4.4), was proposed to correctly model realizability. The problem is,

\[
\begin{align*}
\text{find } & w \\
\text{subject to } & w_i + \pi^l_i - \pi^l_j + y^l_{ij} \geq 1, & a := (i, j) \in A, l \in L, \\
& (w_i + \pi^l_i - \pi^l_j) y^l_{ij} = 0, & a := (i, j) \in A, l \in L, \\
& \sum_{a \in \delta^+ (i)} y^l_a \geq 1, & i \in N \setminus \{i\}, l \in L, \\
& y^l_a = 1, & a \in A^l, l \in L. \\
& y^l_a = 0, & a \in \bar{A}^l, l \in L. \\
& y^l_a \in \mathbb{B}, & a \in A, l \in L, \\
& w_i \geq 1, & a \in A. \\
\end{align*}
\]  

Here, \( y^l_a \) is 1 if \( a \in A \) is an SP-arc for \( l \in L \) and 0 if the arc is a non-SP-arc.

We give the following example to illustrate an important difference between realizability and compatibility.

**Example 5.1**

Consider an ISPR instance with destinations \( L = \{a, b\} \) where the SP-graphs are given by the SP- and non-SP-arc sets

\[
A^a = \{(1, 2), (2, 3)\}, \quad \bar{A}^a = \emptyset, \quad A^b = \{(1, 3)\} \quad \text{and} \quad \bar{A}^b = \{(1, 2)\}.
\]  

These SP-arc patterns are illustrated on the left in Figure 5.1.

![Figure 5.1: Two ISPR instances. The SP-arcs for destination a and b are drawn with solid and dashed arcs, respectively. (Left) Two SP-graphs that directly induce a reduced cost routing conflict, i.e. they are not compatible. (Right) Removing arc (2, 3) from \( A^a \) on the left and setting \( a = 3 \) yields the SP-graphs on the right. They are compatible but not realizable since they indirectly induce a conflict via some path to node 3 represented by the curly dotted arrow.](image)

It is straightforward to use an ad hoc argument to show that arc \( (1, 2) \) must be both an SP- and a non-SP-arc to destination \( b \). Therefore, the instance is not compatible, nor realizable. A simple rigorous proof is obtained by adding the reduced cost constraints
\[ w_{12} + \pi_1^b - \pi_2^b \geq 1, -(w_{12} + \pi_1^a - \pi_2^a) = 0, \]
\[ w_{13} + \pi_1^a - \pi_3^a \geq 0, -(w_{13} + \pi_1^b - \pi_3^b) = 0, \tag{5.5} \]
\[ w_{23} + \pi_2^b - \pi_3^b \geq 0, -(w_{23} + \pi_2^a - \pi_3^a) = 0, \]
which yields the surrogate constraint \( 0 \geq 1 \).

Suppose that the SP-arc set for destination \( a \) is altered to \( A^a = \{1, 2\} \) and also that destination \( a \) is node 3. This modification is illustrated on the right in Figure 5.1. It makes model (5.2) feasible, so the new instance is compatible. It is however not realizable, since any augmentation to SP-graphs that contain arborescences yields some reduced cost zero path from node 2 to node 3 w.r.t. the node potentials for destination \( a = 3 \). Any such path induces a conflict in a similar manner as above.

Example 5.1 shows a severe weakness of compatibility; it does not take information about the destinations into account. One purpose of our new relaxation is to find a remedy of this flaw. The central idea is to incorporate some destination information via auxiliary destination dependent arcs.

### 5.2 Partial Realizability

Observe that model (5.2) is obtained from (5.3) by dropping the outdegree constraints and the complementarity constraints that involve non-fixed binary variables. Our new relaxation is derived by lifting and strengthen model (5.3) before relaxing these constraints.

The effect of lifting in our case is to explicitly introduce some destination information via a set of distance variables. These new variables will be associated with a set of auxiliary destination arcs, referred to as D-arcs. By treating a node-destination pair without an emanating SP-arc as an arc we can define the set, \( D \), of D-arcs as

\[ D = \bigcup_{l \in L} D^l, \quad \text{and} \quad D^l = \{ a := (i, l) \in N \times N \mid i \neq l, \delta^+(i) \cap A^l = \emptyset \}. \tag{5.6} \]

The purpose of a D-arc is to represent the following fundamental fact. When an SP-graph is eventually augmented to contain a shortest path arborescence, there will be some shortest path from each node to the respective destination.

We lift (5.3) by introducing an auxiliary distance variable, \( d_a = \pi_i^l - \pi_j^l \), for each D-arc, \( a := (i, l) \in D \). It follows directly that the below distance lower bound inequalities are valid.

**Proposition 5.1**

The inequalities

\[ d_{st} \geq \pi_i^l - \pi_j^l, \quad s, t \in N, l \in L, \]
\[ d_{st} \geq 1, \quad s \neq t, s, t \in N, \tag{5.7} \]
are valid for (5.3) where

\[ d_{st} = \pi^t_s - \pi^s_t, \quad s, t \in N. \] (5.8)

Augmenting the valid inequalities in Proposition 5.1 associated with D-arcs before dropping the outdegree and complementarity constraints yields the following stronger relaxation of realizability, referred to as partial realizability, i.e.

\[
\begin{align*}
\text{find} \quad w \\
\text{subject to} \\
& w_a + \pi^l_i - \pi^l_j = 0, \quad a := (i, j) \in \mathcal{A}^l \cup \mathcal{D}^l, \ l \in L, \quad (5.9a) \\
& w_a + \pi^l_i - \pi^l_j \geq 1, \quad a := (i, j) \in \mathcal{A}^l, \ l \in L, \quad (5.9b) \\
& w_a + \pi^l_i - \pi^l_j \geq 1, \quad a := (i, j) \in \mathcal{A} \setminus (\mathcal{A}^l \cup \mathcal{A}^i), \ l \in L, \quad (5.9c) \\
& d_a + \pi^l_i - \pi^l_j = 0, \quad a := (i, j) \in \mathcal{D}^l, \ l \in L, \quad (5.9d) \\
& d_a + \pi^l_i - \pi^l_j \geq 0, \quad a := (i, j) \in \mathcal{D}^l, \ l \in L, \quad (5.9e) \\
& w_a \geq 1, \quad a \in \mathcal{A}, \quad (5.9f) \\
& d_a \geq 1, \quad a \in \mathcal{D}. \quad (5.9g)
\end{align*}
\]

We say that a family of SP-graphs, \( \mathcal{A}^L = \{(\mathcal{A}^l, \mathcal{A}^i)\}_{l \in L} \), is partially realizable if model (5.9) is feasible. A definition obtained by incorporating D-arcs is given in terms of compatibility of generalized SP-graphs in Definition 5.2 on page 62.

Since model (5.9) is still a relaxation of (5.3), it can only be used to identify some SPR conflicts. In addition to the direct reduced cost routing conflicts found by (5.2) it also finds indirect conflicts that can be deduced by the additional fact that there will eventually be a path from each node to its destination. One example of such a conflict was given in Example 5.1 above.

--- Example 5.2 ---

Re-consider the SP-graphs on the right in Figure 5.1 from Example 5.1, i.e.

\[
\mathcal{A}^u = \{(1, 2)\}, \quad \mathcal{A}^a = \emptyset, \quad \mathcal{A}^b = \{(1, 3)\} \quad \text{and} \quad \mathcal{A}^h = \{(1, 2)\},
\]

(5.10)

where the destination for SP-graph \( a \) is 3.

Recall that this instance is compatible, but not realizable. We show that it is not partially realizable. The outdegree of node 2 is 0 w.r.t. SP-arc set \( \mathcal{A}^u \), therefore, the arc \((2, a) = (2, 3)\) is in \( D^3 \subseteq D \). Therefore, model (5.9) contains the constraints,

\[
\begin{align*}
&w_{12} + \pi^b_1 - \pi^b_3 \geq 1, \quad -(w_{12} + \pi^b_1 - \pi^b_3) = 0, \\
&w_{13} + \pi^b_1 - \pi^b_3 \geq 0, \quad -(w_{13} + \pi^b_1 - \pi^b_3) = 0, \\
&d_{23} + \pi^b_2 - \pi^b_3 \geq 0, \quad -(d_{23} + \pi^b_2 - \pi^b_3) = 0,
\end{align*}
\]

(5.11)

where the constraints at the bottom in (5.11) stem from a D-arc. As in Example 5.1, the surrogate constraint \( 0 \geq 1 \) is obtained by adding the constraints in (5.11). This implies
that model (5.9) is infeasible and the SP-graphs are not partially realizable, nor realizable.

For more complicated examples that are compatible but not partially realizable we refer to [73]. It can be shown that Example 4.2 is not partially realizable. Hence, partially realizable is a necessary but not sufficient condition for realizability. Next, we augment ordinary SP-graphs by D-arcs to form generalized SP-graphs.

5.2.1 Generalized Shortest Path Graphs

The purpose of generalized SP-graphs is to unify the formulation of partial realizability and related SPR conflicts and their corresponding valid inequalities.

Since a D-arc is in a sense a shortest path arc, we refer to the union of SP-arcs and D-arcs as generalized SP-arcs. Given an SP-graph to destination \( l \), the corresponding generalized SP-graph is obtained by also including all arcs in \( D_l \), i.e. a generalized SP-graph consists of the generalized SP-arcs and (ordinary) non-SP-arcs associated with its destination. Thus, a generalized SP-graph also takes some destination information into account, via its D-arcs.

Definition 5.1

Let \( \mathcal{A}^L = \{ (A^l, \bar{A}^l) \}_{l \in L} \) be a family of SP-graphs. The associated family of generalized SP-graphs is defined as \( \bar{\mathcal{A}}^L = \{ (A^l, \bar{A}^l, D_l) \}_{l \in L} \) where

\[
D_l = \{ a := (i, l) \mid i \neq l, \delta^+(i) \cap A^l = \emptyset \}. \tag{5.12}
\]

To describe partial realizability in terms of generalized SP-graphs we observe from model (5.9) that there is no structural difference between weight variables and distance variables. Therefore, we rename all distance variables and combine SP-arcs and D-arcs to obtain the following definition.

Definition 5.2

Let \( \mathcal{A}^L = \{ (A^l, \bar{A}^l) \}_{l \in L} \) be a family of SP-graphs and denote the associated family of generalized SP-graphs by \( \bar{\mathcal{A}}^L = \{ (A^l, \bar{A}^l, D_l) \}_{l \in L} \). Then, \( \mathcal{A}^L \) (and \( \bar{\mathcal{A}}^L \)) is partially realizable if there exists a strictly positive weight vector, \( w \in \mathbb{R}^+ \) and a set of node potentials, \( \{ \pi_l \}_{l \in L} \), such that for all \( l \in L \) it holds that the reduced cost is

1. zero for generalized SP-arcs, i.e. \( \bar{w}_a + \pi_i^l - \pi_j^l = 0 \) if \( a := (i, j) \in A^l \cup D_l \),
2. at least one for non-SP-arcs, i.e. \( \bar{w}_a + \pi_i^l - \pi_j^l \geq 1 \) if \( a := (i, j) \in \bar{A}^l \),
3. non-negative for all arcs, i.e. \( \bar{w}_a + \pi_i^l - \pi_j^l \geq 0 \) if \( a := (i, j) \in A \cup D \).

To describe partial realizability in terms of generalized SP-graphs we observe from model (5.9) that there is no structural difference between weight variables and distance variables. Therefore, we rename all distance variables and combine SP-arcs and D-arcs to generalize SP-arcs to obtain the following definition.
This yields that a family of generalized SP-graphs is partially realizable if and only if the following problem is feasible,

\[
\begin{align*}
\text{find} & \quad \tilde{w} \\
\text{subject to} & \quad \tilde{w}_a + \pi^I_l - \pi^J_l = 0, \quad a := (i,j) \in A^l \cup D^l, \quad l \in L, \quad (5.13a) \\
& \quad \tilde{w}_a + \pi^I_l - \pi^J_l \geq 1, \quad a := (i,j) \in \bar{A}^l, \quad l \in L, \quad (5.13b) \\
& \quad \tilde{w}_a + \pi^I_l - \pi^J_l \geq 0, \quad a := (i,j) \in \bar{U}^l, \quad l \in L, \quad (5.13c) \\
& \quad \tilde{w}_a \geq 1, \quad a \in \bar{A}. \quad (5.13d)
\end{align*}
\]

Models (5.2) and (5.13) are indeed similar. In particular, partial realizability can be seen as compatibility of generalized SP-graphs. It is very important to note that despite the similar appearance of models (5.13) and (5.2), it should be clear from above that there is a substantial difference between them, see also the next section. Also observe that model (5.2) is obtained from (5.13) by setting \(D = \emptyset\).

We have two more remarks on generalized SP-graphs and partial realizability. First observe that the set of generalized SP-arcs associated with destination \(l\) always contains a reversely spanning arborescence rooted at \(l\). Further, all D-arcs are included in such an arborescence. This additional structure of generalized SP-graphs and model (5.13) can be crucial, see further Chapter 8.

Finally, we note that it is possible to consider partial realizability of generalized SP-graphs on a simple graph, i.e. without parallel arcs. This requires preprocessing and postprocessing to decide if an SP-arc or a D-arc should be used. Indeed, if \(a := (i,l') \in A^l\) for some \(l,l'\), and \(a \in D^{l'}\), then \(a\) must be in \(A^{l'}\) and can be removed from \(D^{l'}\). If for all \(l \in L\), arc \(a \notin A^l\), then arc \(a\) can be removed from \(A\). Hence, we can make \(A^{l'}\) and \(D^{l'}\) disjoint. In the latter case, postprocessing must be used to appropriately re-insert \(a\) into \(A\).

### 5.2.2 Relation Between ISPR Formulations

The major difference between model (5.3) and models (5.2) and (5.13), is that the relaxations only take dual feasibility of the shortest path problem into account, but partly neglect primal feasibility and complementary slackness. This implies that model (5.2) can only be used to identify direct reduced cost routing conflicts among a subset of SP-arcs and non-SP-arcs. Model (5.13) further finds the reduced cost routing conflicts involving destinations that will arise when the SP-graphs are augmented by arbitrary shortest paths to the destinations. Finally, model (5.3) detects when all possible augmentations of the SP-graphs eventually result in an arbitrary reduced cost routing conflict.

The relation among the feasible instances for the ISPR problem we have considered is summarized in the following theorem.

**Theorem 5.1**

Let \(C, P\) and \(R\) be the collections of families of SP-graphs that are compatible, partially realizable and realizable, respectively. Then,
\[ \mathcal{C} \subset \mathcal{P} \subset \mathcal{R} \]  \hspace{1cm} (5.14)

**Proof:** The inclusions are obvious. Strictness follows from Examples 5.1, 5.2, and 4.2 (see the remark after Example 5.2).

An interesting question is when the realizability problem is polynomially solvable, in particular, when compatibility implies realizability. On the negative side, the NP-completeness proof in Chapter 4 yields as a byproduct the following result; it is NP-complete to decide if a family of SP-graphs is realizable even if for each SP-graph the SP-arcs form a rooted arborescence with maximum depth 2. We give some sufficient conditions for the equivalence of realizability and compatibility.

We observed above that compatibility is a necessary, but in general not sufficient, condition for realizability. Therefore, incompatibility certificates are also un-realizability certificates. A compatibility certificate is a realizability certificate if and only if the associated tight node potentials also yield a compatibility certificate. This implies that realizability is equivalent to compatibility if all SP-graphs are spanning, cf. [61].

To obtain a more general sufficient condition we note that a compatible family of SP-graphs is realizable if it can be completed to a compatible family of spanning SP-graphs. This is the case if all non-SP-arcs involve only nodes spanned by an arborescence of SP-arcs rooted at the associated destination. Indeed, this implies that all affected node potentials are correct. Since nodes and arcs that involve nodes outside this arborescence are unrestricted, the compatible weights can be used to determine correct node potentials and to decide which arcs that are SP-arcs and non-SP-arcs.

**Proposition 5.2**

Let \( \mathcal{A}^L = \{ (A^l, \bar{A}^l) \}_{l \in L} \) be a compatible family of SP-graphs where for each \( l \in L \), the SP-arcs, \( A^l \), induce an arborescence rooted at \( l \) and \( \bar{A}^l \subseteq S^l \), where

\[ S^l = \{ a := (i, j) \in A \mid i, j \in N(A^l) \} \]  \hspace{1cm} (5.15)

Then, \( \mathcal{A}^L \) is realizable.

**Corollary 5.1**

Let \( \mathcal{A}^L = \{ (A^l, \bar{A}^l) \}_{l \in L} \) be a compatible family of SP-graphs where for each \( l \in L \), the SP-arcs, \( A^l \), induce a spanning arborescence rooted at \( l \). Then, \( \mathcal{A}^L \) is realizable.

Note that the conditions in Proposition 5.2 and Corollary 5.1 can easily be verified.

**Remark 5.1.** Note that there is a minor problem with the derivation of Proposition 5.2 above if paths are required to be unique since the weights can induce multiple shortest paths from unrestricted nodes. However, uniqueness can be obtained by using a perturbation, e.g. the binary perturbation technique in the proof of Proposition 5.4 in [46, p. 74]. Therefore, Corollary 5.1 holds in the case of unique paths as well.

### 5.3 Valid Inequalities from Partial Realizability

The primary reason to use partial realizability instead of compatibility is that a stronger relaxation yields more and stronger valid inequalities. Below, we consider SPR constraints.
that only involve binary shortest path indicator variables, \( y \). Recall that \( y^l_a = 1 \) if \( a \in A \) is on a shortest path to destination \( l \in L \).

By analyzing system (5.2) it is possible to derive a subclass of SPR conflict constraints that is sufficient, in the integer sense, to remove all weight variables. This class is induced by all irreducible infeasible subsystems of (5.2). If instead infeasible subsystems of (5.13) are considered we obtain a super class that induces more and stronger inequalities. This advantage is gained since (5.3) is lifted and strengthened before the weight variables are projected out.

A combinatorial characterization of a huge subclass of SPR conflict inequalities is derived by considering the Farkas system of (5.13), or (5.2), see further Chapters 6 and 7. A particularly comprehensible subclass of SPR conflicts is presented in [61, 62]. There, the Farkas system of (5.2) is analyzed to derive the class of SPR conflicts that stem from at most two (ordinary) SP-graphs. These conflicts are referred to as valid cycles. The simplest case of a valid cycle consists of two internally disjoint \((s, t)\)-paths. We here refer to such conflicts as subpath conflicts, see the top of Figure 5.2. A subpath conflict induced by the two \((s, t)\)-paths \( P_1 \) and \( P_2 \), associated with destinations \( l' \) and \( l'' \), respectively, yields the valid inequalities

\[
\sum_{a \in P_1} (1 - y^l_a) + \sum_{a \in P_2} (1 - y^{l''}_a) \geq 1 - y^{l'}_a, \quad a \in P_1,
\]

\[
\sum_{a \in P_1} (1 - y^l_a) + \sum_{a \in P_2} (1 - y^{l''}_a) \geq 1 - y^{l'}_a, \quad a \in P_2.
\]  

(5.16)

An interpretation of (5.16) is that if \( P_1 \) and \( P_2 \) are shortest paths to their respective destinations, then both paths must actually be shortest to the other destination.

It is straightforward to translate valid cycle results obtained by analyzing (5.2) to generalized valid cycles by mimicking the analysis for (5.13). For simplicity, we here only consider the simplest case of a generalized valid cycle. Analogously, it consists of two internally disjoint \((s, t)\)-paths, but the last arc in one of the paths must now be an auxiliary D-arc to \( t \), see the bottom of Figure 5.2. This conflict is here referred to as a generalized subpath conflict. A generalized subpath conflict induced by paths \( P_1 \) and \( P_2 \cup \{a\} \), where \( a \in D \) is an auxiliary arc, similarly yields the valid inequalities

**Figure 5.2:** A (generalized) subpath conflict consists of two node disjoint \((s, t)\)-paths, \( P_1 \) and \( P_2 \), represented by solid and dashed arcs respectively. (Top) An ordinary subpath conflict. (Bottom) In a generalized subpath conflict, the path \( P_2 \) is always associated with destination \( t \) and contains an auxiliary arc to \( t \). The auxiliary arc is represented by a curly dotted arrow.
Relaxations of Realizability

\[
\sum_{a \in \bar{P}_1} (1 - y_a') + \sum_{a \in \bar{P}_2} (1 - y_a'') \geq 1 - y_a'', \quad \bar{a} \in \bar{P}_1,
\]

\[
\sum_{a \in P_1} (1 - y_a') + \sum_{a \in P_2} (1 - y_a'') \geq 1 - y_a'', \quad \bar{a} \in \bar{P}_2.
\]  

(5.17)

Note that only one inequality in (5.17) is non-dominated. When \( \bar{P}_2 \neq \emptyset \), the only non-dominated inequality in (5.17) is the one associated with the last arc in \( \bar{P}_2 \), i.e. the arc \((u_{p-1}, u_p)\) in Figure 5.2. If \( \bar{P}_2 = \emptyset \), the non-dominated inequality in (5.17) is the one associated with the first arc in \( \bar{P}_1 \), i.e. arc \((s, v_1)\) in Figure 5.2. Also observe that \( \bar{P}_2 \) can be a subpath of \( P_2 \) if \( t = l'' \), i.e. when \( t \) is involved in the conflict. In this case, all but one of the inequalities in (5.16) are dominated by an inequality in (5.17), cf. Example 5.3 below.

To exemplify the descriptive power and unification capabilities of using generalized SP-graphs we note that all classes of combinatorial SPR constraints introduced in [218] are special cases of (generalized) subpath conflicts. Indeed, the transit cut, see Figure 1 and equation (10) in [218], is obtained when \( \bar{P}_2 = \emptyset \) and \((u, v)\) is the first arc in \( \bar{P}_1 \). The split cut, see Figure 1 and equation (10) in [218], is obtained when \( \bar{P}_2 = \{(s, i)\} \) and \((u, v) = (s, i)\) for some \( i \in L \).

As a byproduct of the above, we get a straightforward mathematical motivation in terms of model (5.13) for validity of the combinatorial inequalities in [218] instead of the ad hoc arguments used there.

---

**Example 5.3**

Consider the subpath conflict in Figure 5.3 on page 66 for destinations \( L = \{l', l''\} \) where the associated SP-arc sets are

\[
A_{l'} = \{(1, 4), (4, 5)\}, \quad A_{l''} = \{(1, 2), (2, 3), (3, 5)\},
\]

(5.18)

and destination \( l' \) is not involved in the conflict, but \( l'' = 5 \) is.

**Figure 5.3:** A subpath conflict consisting of the disjoint paths, \( P_1 = 1 \rightarrow 4 \rightarrow 5 \) and \( P_2 = 1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \), represented by solid and dashed arcs respectively.

Applying (5.16) with \( P_1 = \{(1, 4), (4, 5)\} \) and \( P_2 = \{(1, 2), (2, 3), (3, 5)\} \) gives the five valid inequalities below, one for each arc in the conflict,
5.3 Valid Inequalities from Partial Realizability

\begin{align*}
y_{14}^{r} + y_{45}^{r} + y_{12}^{r} + y_{23}^{r} + y_{35}^{r} & \leq 4 + y_{12}^{r}, \\
y_{14}^{r} + y_{45}^{r} + y_{12}^{r} + y_{23}^{r} + y_{35}^{r} & \leq 4 + y_{23}^{r}, \\
y_{14}^{r} + y_{45}^{r} + y_{12}^{r} + y_{23}^{r} + y_{35}^{r} & \leq 4 + y_{23}^{r}, \\
y_{14}^{r} + y_{45}^{r} + y_{12}^{r} + y_{23}^{r} + y_{35}^{r} & \leq 4 + y_{23}^{r}, \\
y_{14}^{r} + y_{45}^{r} + y_{12}^{r} + y_{23}^{r} + y_{35}^{r} & \leq 4 + y_{14}^{r}, \\
y_{14}^{r} + y_{45}^{r} + y_{12}^{r} + y_{23}^{r} + y_{35}^{r} & \leq 4 + y_{14}^{r}.
\end{align*}

(5.19)

Since $l'' = 5$, there are also generalized subpath conflicts via auxiliary D-arcs. The four possible generalized subpath conflicts are given in Figure 5.4.

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{figure5.4.png}
\caption{Four generalized subpath conflicts induced by the subpath conflict in Figure 5.4. The paths $P_1$ and $P_2$ are represented by solid and dashed arcs respectively. The auxiliary arc is represented by a dotted arrow. Note that the top right conflict violates the split property in [218]. Similarly, the bottom conflicts violate the transit property.}
\end{figure}

Applying (5.17) gives the four non-dominated valid inequalities

\begin{align*}
y_{14}^{r} + y_{45}^{r} + y_{12}^{r} + y_{23}^{r} & \leq 3 + y_{23}^{r}, \\
y_{14}^{r} + y_{45}^{r} + y_{12}^{r} & \leq 2 + y_{12}^{r}, \\
y_{14}^{r} + y_{45}^{r} & \leq 1 + y_{14}^{r}, \\
y_{45}^{r} & \leq 0 + y_{45}^{r}.
\end{align*}

(5.20)

Observe that all inequalities in (5.19) except the third are dominated by some inequality in (5.20).

To illustrate the connection to [218], we note that the second inequality in (5.20) is a split cut and that the last two inequalities are transit cuts. The first inequality is a generalization of a split cut that has not earlier been considered (to the best of our knowledge).
The technique of using D-arcs above is not restricted to the case of "simple" subpath conflicts. It is possible to apply the same reasoning to any class of conflicts.

Let us finally consider the computational aspect. Typically, stronger ISPR relaxations are desirable since they yield better valid inequalities. In practice, this often comes at the price of a higher computational effort of solving them, i.e. to separate the inequalities.

Models (5.2) and (5.13) are structurally equivalent. Since (5.13) is moderately larger, one can only expect that (5.13) requires some more effort to solve than (5.2). However, the preliminary computational experiments in [202] suggest the fastest methods to solve the above ISPR relaxations use fundamental cycle basis reformulations of the problem, see further Chapter 8. The efficiency of this approach depends completely on suitable reduced cost zero intrees to the destinations. In (5.13), the D-arcs yield reduced cost zero arborescences together with the SP-arcs. Hence, the additional structure in (5.13) makes it very well suited for the fundamental cycle basis method.

Observe that (5.2) and (5.13) can only be used to separate the binary part of tentative routing solutions. For some classes of SPR conflicts, separation algorithms for fractional solutions have been derived, see further [50]. It is straightforward to modify these to also include D-arcs, and hence separate their generalized counterpart, we do this in Part III, see also [73, Section 8.3].

For some classes of SPR conflicts, more efficient separation algorithms for binary solutions have been derived, e.g. the valid cycle algorithm in [62]. This algorithm is easily adapted to incorporate D-arcs, see further Chapter 6 and [73, Section 5.4.3].

Above, several arguments in favor of model (5.13) over (5.2) are given. The size argument may be the only argument against, but it is effectively countered by the fact that (5.13) can be solved more efficiently than (5.2) by using the fundamental cycle basis reformulation. In the remaining part of this thesis, we will focus on partial realizability instead of compatibility. Also, we will not distinguish between SP-graphs and generalized SP-graphs, but simply say SP-graphs in both cases. The same applies to SP-arcs and generalized SP-arcs.
Classes of Infeasible Structures

Analyzing the infeasibility of inverse shortest path routing (ISPR) problems is a rich source of routing based valid inequalities for bilevel shortest path (BSP) problems. We initiate such an analysis here by considering the Farkas system of the ISPR problems from the previous chapters. In particular, we characterize five classes of infeasible structures via the solutions to the Farkas system. These classes are shown to be exhaustive and strictly nested.

Outline The Farkas systems of the ISPR problems from Chapter 5 are presented in Section 6.1. The set of solutions to the Farkas system of an ISPR model yields a characterization of infeasible routing patterns in Section 6.2. The extremal structure of the polyhedral cone induced by the Farkas system is considered in Section 6.3. Finally, the relation between all considered classes of structures is presented in Section 6.4.

6.1 Problem Formulation

Formulations of ISPR problems can be found in previous chapters and in the literature, e.g. in [25, 46, 58, 191]. The Farkas systems of some of these models have been analyzed to find classes of infeasible routing patterns, e.g. in [59, 61]. Other methods, of more pragmatic character, have also be used to derive similar structures and subclasses of infeasible patterns, e.g. in [46, 85, 218].

In this chapter a very large class of (potentially) infeasible structures is derived based on the Farkas systems of the LP models (5.2) and (5.13) in Chapter 5. Each potentially infeasible structure in this class either implies that a family of SP-graphs is not realizable or forces some part of a routing pattern to be in a certain way. From now on we refer to this only as infeasibility.

Since partial realizability is only a necessary condition for realizability, the conditions derived from the Farkas system are in general not sufficient. Therefore, the class of struc-
tures derived from the Farkas systems cannot be exhaustive. When we, in terms of the discussion in Section 5.2.2, define a routing conflict to be a conflict that can be deduced solely from the the SP-arcs and non-SP-arcs, then the class is exhaustive. Further, in the important special case where all SP-graphs are spanning, partial realizability is equivalent to realizability, see Proposition 5.2 and Corollary 5.1. Also, in the context where these structures are used to produce valid inequalities there are typically constraints that induce the preconditions in Proposition 5.2.

Throughout the chapter $G = (N, A)$ is a digraph and $A^L$ a family of SP-graphs for the set of destinations $L \subseteq N$. Each SP-graph, $(A^l, \overline{A}^l) \in A^L$, describes which arcs that must be and which arcs that are not allowed to be in the shortest paths to the destination $l \in L$. Arcs not in the SP-graph, $(A^l, \overline{A}^l)$, are unrestricted and we do not care if they are in a shortest path to $l$ or not. This set of arcs is denoted by $U^l = A_n(A^l\cup \overline{A}^l)$.

### 6.1.1 The ISPR Compatibility Model

The compatibility model, i.e. (5.2), is repeated here for convenience. Recall that there is a link weight, $w_a$, for each arc $a := (i, j) \in A$ and a node potential, $\pi^l_i$, for each node $i \in N$ and each destination $l \in L$. From Section 4.2 and 5.1.1 we know that $A^L$ is compatible if and only if the following system has a feasible solution.

\[
\begin{align*}
\text{find} & \quad w \\
\text{subject to} & \quad w_a + \pi^l_i - \pi^l_j = 0, \quad a := (i, j) \in A^l, \ l \in L, \quad (6.1a) \\
& \quad w_a + \pi^l_i - \pi^l_j \geq 1, \quad a := (i, j) \in \overline{A}^l, \ l \in L, \quad (6.1b) \\
& \quad w_a + \pi^l_i - \pi^l_j \geq 0, \quad a := (i, j) \in A \setminus (A^l \cup \overline{A}^l), \ l \in L, \quad (6.1c) \\
& \quad w_a \geq 1, \quad a \in A. \quad (6.1d)
\end{align*}
\]

If the variable substitution $w := w - 1$ is applied then Farkas’ lemma can be used to conclude that $A^L$ is not compatible if and only if the following problem is feasible.

\[
\begin{align*}
\text{find} & \quad \theta \\
\text{subject to} & \quad \sum_{l \in L} \sum_{a \in A^l \cup \overline{A}^l} \theta_a^l < 0, \quad (6.2a) \\
& \quad \sum_{a \in A^l \cup \overline{A}^l} \theta_a^l - \sum_{a \in A \setminus (A^l \cup \overline{A}^l)} \theta_a^l = 0, \quad i \in N, \ l \in L, \quad (6.2b) \\
& \quad \sum_{i \in L} \theta_a^l \leq 0, \quad a \in A, \quad (6.2c) \\
& \quad \theta_a^l \geq 0, \quad a \in A \setminus (A^l \cup U^l), \ l \in L. \quad (6.2d)
\end{align*}
\]
Farkas’ lemma and the results in the Chapter 5 yield the following conclusion. The family of SP-graphs $A^L$ is not realizable if (6.2) has a feasible solution. If (6.2) does not have a feasible solution, then $A^L$ is realizable if the arc sets in each SP-graph induce a reversely spanning arborescence rooted at the destination and may be realizable otherwise.

**Remark 6.1.** The Farkas approach used above have earlier been used for models that are very similar to (6.1), e.g. in [61, 62, 193]. For instance, in [61] all SP-graphs are spanning, so $U^l = \emptyset$ for all $l \in L$ and $A^L$ is realizable if and only if (6.1) is feasible.

### 6.1.2 The ISPR Partial Realizability Model

We repeat the partial realizability model, (5.13), and the required definitions here. Let $\overline{G} = (N, \overline{A})$ be the directed multigraph induced by $\overline{A}^L$ where the multiset $\overline{A}$ consists of the ordinary arcs $A$ and all D-arcs as in Section 5.2.1, i.e.

$$D = \bigcup_{l \in L} \overline{D}^l. \quad (6.3)$$

where

$$\overline{D}^l = \{ a := (i, l) \mid i \neq l, \delta^+(i) \cap A^l = \emptyset \} , \quad (6.4)$$

As before, we write $\overline{A} = A \cup D$. Also, arcs in $\overline{G}$ outside the SP-graph, $\left( A^l, \overline{A}^l, \overline{D}^l \right)$, are unrestricted and denoted by

$$\overline{U}^l = \overline{A} \setminus (A^l \cup \overline{D}^l \cup \overline{A}^l). \quad (6.5)$$

This yields the partial realizability model from the previous chapter.

**[ISPR-PR]**

```
find $\overline{w}$
subject to
\begin{align*}
\overline{w}_a + \pi^l_i - \pi^l_j &= 0, & a := (i, j) \in A^l \cup \overline{D}^l, & l \in L, \quad (6.6a) \\
\overline{w}_a + \pi^l_i - \pi^l_j &\geq 1, & a := (i, j) \in \overline{A}^l, & l \in L, \quad (6.6b) \\
\overline{w}_a + \pi^l_i - \pi^l_j &\geq 0, & a := (i, j) \in \overline{U}^l, & l \in L, \quad (6.6c) \\
\overline{w}_a &\geq 1, & a \in \overline{A}. \quad (6.6d)
\end{align*}
```

Again, using the variable substitution $\overline{w} = \overline{w} - 1$ and Farkas’ lemma yields that $\overline{A}^L$ is not partially realizable if and only if the following problem is feasible.
find $\theta$

subject to

\[ \sum_{l \in L} \sum_{a \in \tilde{A} \setminus \tilde{A}^I} \theta_a^l < 0, \]  
\[ \sum_{a \in \delta^+(i)} \theta_a^l - \sum_{a \in \delta^-(i)} \theta_a^l = 0, \quad i \in N, \; l \in L, \]  
\[ \sum_{l \in L} \theta_a^l \leq 0, \quad a \in \tilde{A}, \]  
\[ \theta_a^l \geq 0, \quad a \in \tilde{U}, \; l \in L. \]  

(6.7a) \hspace{1cm} (6.7b) \hspace{1cm} (6.7c) \hspace{1cm} (6.7d)

Analogously as above, the conclusion is that the family of SP-graphs, $\tilde{A}^I$, is not realizable if (6.7) has a feasible solution and that $\tilde{A}^I$ may be realizable otherwise.

It is clear from above that (6.6) and (6.7) constitute a unified framework that covers all (polynomially solvable) cases that we have considered in this thesis. Therefore, it is sufficient to only analyze these models in the following. Two remarks are in place.

**Remark 6.2.** Models (6.1) and (6.2) are obtained from (6.6) and (6.7), by setting $\tilde{D} = \emptyset$.

**Remark 6.3.** A word of caution is in place. Despite the fact that all the feasibility models, and their Farkas systems, look almost identical, they are not. An obvious difference lies in the additional constraints (or the objective of the Farkas systems). A more subtle and important difference is (unfortunately) hidden by the notation. The properties of the SP-graphs strongly affects which conclusions one is able to draw from the (in)feasibility of the models. For instance, for spanning SP-graphs partial compatibility is equivalent to realizability, but in other cases a partially realizable family of SP-graphs may not be realizable.

The main objective in this chapter that we have been striving for is to characterize (classes) routing patterns that are not realizable. Due to the complexity of realizability we (have to) settle with partial realizability. This is a fair compromise since (6.7) have a lot of structure and seem to yield fairly strong necessary conditions in practice. To find infeasible structures the set of feasible solutions to (6.7) is analyzed.

### 6.2 Classes of Infeasible Routing Pattern Structures

There is a very close connection between (6.7) and an ordinary multicommodity flow problem. To utilize and also emphasize this, the constraints in (6.7) are referred to as: the objective, node balance, capacity and commodity specific flow bound constraints, respectively. The left hand side in the objective constraint is called the objective value.

When (6.7) is examined, it is clear that a solution is a multicommodity circulation since all node balances are zero. There are two significant differences between our model and a standard multicommodity flow model, namely, the aggregated arc capacities are zero and the flow on certain arcs is allowed to be negative. Despite these differences,
it is motivated to consider alternative modelling approaches based on multicommodity circulations. Such an approach based on fundamental cycle bases yields the novel model in Chapter 8.

Throughout this chapter, the set of feasible solutions to (6.7) is denoted by \( \Theta \). The closure of \( \Theta \), denoted by \( \text{cl} \, \Theta \), is the set of solutions that are feasible in (6.7) when the strict inequality in the objective constraint is replaced by a weak inequality. It is also of interest to consider points not in both these sets, i.e., circulations that satisfy the capacity and commodity specific flow bounds but where the objective value is 0. This set,

\[
\Theta^0 = (\text{cl} \, \Theta) \setminus \Theta,
\]

is referred to as the set of non-improving solutions.

The observations about the multicommodity structure of (6.7) should be used when the source of infeasibility is analyzed. Our aim is to explain infeasibility by the presence of some combinatorial structure formed by the SP-graphs in an infeasible instance. First we consider the general combinatorial structure, which is a special collection of cycles. Then this structure is specialized to cases that are easier to analyze, which yields stronger results and in some cases efficient algorithms for finding the structure.

Several examples of the structures to be presented below are given at the end of this chapter and throughout the thesis. First, we focus solely on the definitions of the structures.

### 6.2.1 The General Infeasible Routing Pattern Structure

Consider a point, \( \theta \), in the closure of the set of feasible solutions to (6.7), i.e., \( \theta \in \text{cl} \, \Theta \), so the objective value is allowed to be 0. From the discussion above we have that \( \theta \) is a kind of multicommodity circulation where the individual commodity specific flow bounds are satisfied and the aggregated flow on each arc is at most 0. A circulation can always be decomposed into flows in undirected cycles. In such a decomposition, each cycle consists of a set of “forward” arcs with positive flow and a set of “backward” arcs with negative flow. This implies that the orientation induced by the forward and backward labellings gives a directed cycle. Hence, a circulation may be represented by a collection of forward and backward arc sets along with the positive amount of flow in the cycles.

For destination, \( l \in L \), we decompose the circulation into the \( K^l \) cycles,

\[
\{ C^l_k \mid C^l_k = F^l_k \cup B^l_k, \, k = 1, \ldots, K^l \},
\]

with flow, \( x^l_k \).

Since the closure, \( \text{cl} \, \Theta \), is considered the origin is feasible. However, every other point must have some negative flow on some arc to fulfill capacity constraints. The commodity specific non-negative flow constraints imply that all arcs with negative flow must belong to their corresponding SP-graph, i.e., \( B^l_k \subseteq A^l \cup \bar{D}^l \).

When the cycle decomposition is applied to all destinations we obtain that the multicommodity circulation, \( \theta \), can be represented by a family of cycles and flows as follows.

\[
C = \{ C^l \}_{l \in L} \quad \text{and} \quad x = \{ x^l \}_{l \in L},
\]

where
74

6 Classes of Infeasible Structures

\[ C^l = \left\{ C^l_k = F^l_k \cup B^l_k \mid B^l_k \subseteq A^l \cup \bar{D}^l, \ k = 1, \ldots, K^l \right\} \quad \text{and} \quad x^l \in \mathbb{R}_{+}^{K^l}. \quad (6.11) \]

This yields that \( \theta \) is obtained from a family of cycles and the associated flow, \( C \) and \( x \), via

\[ \theta^l_a = \sum_{k:a \in F^l_k} x^l_k - \sum_{k:a \in B^l_k} x^l_k. \quad (6.12) \]

**Remark 6.4.** There is not a one-to-one correspondence between the multicommodity circulation \( \theta \) and the cycle family-flow pair, \( C \) and \( x \). A canonical form admitting a one-to-one correspondence is given in [73].

By construction, a \( \theta \) obtained from \( C \) and an \( x \) via (6.12) is feasible w.r.t. the node balance and the commodity specific flow bounds. If \( \theta \) is also feasible w.r.t. the capacity constraints, then \( C \) is feasible. If \( \theta \) is also strictly feasible w.r.t. the objective constraint, \( C \) is improving, otherwise it is non-improving. This yields the following definitions.

**Definition 6.1**
A family of cycles, \( C \), is feasible if there exist an \( x = \{ x^l \}_{l \in L} \geq 0 \) such that \( x \neq 0 \) and the \( \theta \) induced by (6.12) is feasible, i.e. \( \theta \in \Theta \).

**Definition 6.2**
A family of cycles, \( C \), is improving if there exist an \( x = \{ x^l \}_{l \in L} \geq 0 \) such that the \( \theta \) induced by (6.12) is feasible and has a strictly negative objective value, i.e. \( \theta \in \Theta^0 \).

**Definition 6.3**
A family of cycles, \( C \), is non-improving if for all \( x = \{ x^l \}_{l \in L} \geq 0 \) such that the \( \theta \) induced by (6.12) is feasible, the objective value is 0, i.e. \( \theta \in \Theta^0 \).

Note that the objective is always non-positive since

\[ \sum_{l \in L} \sum_{a \in A \setminus A^l} \theta^l_a \leq \sum_{l \in L} \left( \sum_{a \in A \setminus A^l} \theta^l_a + \sum_{a \in A^l} \theta^l_a \right) = \sum_{a \in A} \sum_{l \in L} \theta^l_a \leq 0. \quad (6.13) \]

These definitions yield the following theorem.

**Theorem 6.1**
System (6.7) is feasible if and only there is a family of oriented cycles that is feasible and improving.

For future reference, we also mention the following property of the cycle decomposition above. Each cycle, \( C^l_k \), consists of a set of forward arcs, \( F^l_k \), and backward arcs, \( B^l_k \). Further, both these arc sets consist of the consecutive arcs in \( P_{kl} \) path segments,

\[ F^l_k = \bigcup_{p=1}^{P_{kl}} P_{kp}^l \quad \text{and} \quad B^l_k = \bigcup_{p=1}^{P_{kl}} P_{kp}^l. \quad (6.14) \]
6.2 Classes of Infeasible Routing Pattern Structures

In these partitions, the path segments are alternating, i.e.

\[ C_{kl}^l = \left( \overrightarrow{P_{k1}} \overleftarrow{P_{k1}} \ldots \overrightarrow{P_{kP_k}} \overleftarrow{P_{kP_k}} \right) \]  \hspace{1cm} (6.15)

We also make a distinction between solutions based on whether that satisfy the capacity constraint with equality or not. Let

\[ \Theta^e = \left\{ \theta \in \text{cl} \Theta \mid \sum_{l \in L} \theta^l_a = 0 \text{ for all } l \in L \right\} , \]  \hspace{1cm} (6.16)

and

\[ \Theta^c = \left\{ \theta \in \text{cl} \Theta \mid \sum_{l \in L} \theta^l_a < 0 \text{ for some } l \in L \right\} . \]  \hspace{1cm} (6.17)

A solution to (6.7) is called saturating if it satisfies all capacity constraints with equality and non-saturating if it satisfies some capacity constraint with strict inequality.

**Definition 6.4**

Let \( \theta \) be a solution to (6.7), then \( \theta \) is saturating if \( \theta \in \Theta^e \).

**Definition 6.5**

Let \( \theta \) be a solution to (6.7), then \( \theta \) is non-saturating if \( \theta \in \Theta^c \).

**Remark 6.5.** To the best of our knowledge, the class of non-saturating solutions have not been discussed earlier in the literature. It is mentioned in [60, Theorem 2, p. 512], that it is necessary that not all constraints in a model equivalent to (6.2) for spanning SP-graphs are satisfied with equality if a solution should be improving, cf. (6.13), but no concrete example was given. We provide an example of these solutions with two SP-graphs in Section 7.1. (It is straightforward to turn any example of a saturating solution into an example that contains a non-saturating solution by turning some cycle into a slack cycle, cf. Chapter 8, in particular the proof of Theorem 8.2).

The following two propositions are obtained directly from the definitions and (6.13). They are more general than Theorem 2 in [60] since they are adapted to a more general model, (6.7). In our opinion, they are also more clear since they much better reveal how a strict inequality makes a solution improving (this depends on if the inequality corresponds to a capacity constraint or a commodity flow bound constraint).

**Proposition 6.1**

A non-saturating solution is improving.

**Proof:** If \( \theta \in \Theta^c \), then the last inequality in (6.13) is strict, hence it is improving. \( \square \)

**Corollary 6.1**

All non-improving solutions are saturating, i.e. \( \Theta^0 \subseteq \Theta^e \).

**Proposition 6.2**

A saturating solution, \( \theta \in \Theta^e \), is improving if and only if there is a commodity \( l \) and an arc \( a \) such that \( a \in A_l \) and \( \theta^l_a > 0 \).
Proof: When \( \theta \in \Theta^- \), it follows from (6.13) that

\[
\sum_{l \in L} \sum_{a \in \bar{A}\setminus\bar{A}^l} \theta^l_a \leq \sum_{a \in \bar{A}} \sum_{l \in L} \theta^l_a = \sum_{l \in L} \left( \sum_{a \in \bar{A}\setminus\bar{A}^l} \theta^l_a \right) = 0. \tag{6.18}
\]

Observe that \( \theta^l_a \geq 0 \) for all \( a \in \bar{A}^l \), hence

\[
\sum_{l \in L} \sum_{a \in \bar{A}^l} \theta^l_a = 0 \iff \theta^l_a = 0, \quad \text{for all } a \in \bar{A}^l \text{ and all } l \in L, \tag{6.19}
\]

and

\[
\sum_{l \in L} \sum_{a \in \bar{A}^l} \theta^l_a > 0 \iff \theta^l_a > 0, \quad \text{for some } a \in \bar{A}^l \text{ and some } l \in L. \tag{6.20}
\]

Since the left hand side can be seen as the slack in the objective constraint the claim follows.

In Chapter 8, the relation between saturating and non-saturating solutions is explored more. In particular, it is shown that there exists a saturating solution (not necessarily improving), whenever there exists a non-saturating solution. Further, a saturating solution obtained from a non-saturating solution is under fairly general assumptions improving.

We discuss non-improving solutions next. A family of cycles, \( \mathcal{C} \), that is feasible but not improving yields important information about the associated SP-graphs, \( \bar{A}^l \). In any expansion of \( \bar{A}^l \) to a family of spanning SP-graphs, all arcs in all cycles in \( \mathcal{C} \) must be on shortest paths to their respective destinations. If not, \( \mathcal{C} \) becomes improving. Therefore, a non-improving family of cycles forces some arcs to be SP-arcs in such an expansion. This conclusion can also be derived via a complementary slackness argument from the theory of LP-duality. The flow, \( \theta^l_a \), on an arc \( a := (i,j) \) in a cycle in \( \mathcal{C} \) must be nonzero, hence the corresponding constraint in the dual, i.e. (6.6), must be active. But this constraint is \( w_a + \pi^l_i - \pi^l_j \geq 0 \), therefore the arc must have reduced cost 0 and be on a shortest path to \( l \), so \( a \) must eventually belong to \( A^l \). This gives the following theorem.

**Theorem 6.2**

Let \( \bar{A}^L \) be a family of SP-graphs that is realizable and let \( \mathcal{C} = \{C^l\}_{l \in L} \) be a non-improving family of cycles, where

\[
\mathcal{C}^l = \left\{ C^l_k = F^l_k \cup B^l_k \mid B^l_k \subseteq A^l \cup \bar{D}^l, \ k = 1, \ldots, K^l \right\}. \tag{6.21}
\]

If \( \bar{A}^L \) is completed to a family of spanning SP-graphs, \( \mathcal{I}(w) = \{I^l(w)\}_{l \in L} \), then

\[
B^l_k \subseteq \mathcal{I}(w), \quad k = 1, \ldots, K^l, \ l \in L. \tag{6.22}
\]

Remark 6.6. In the unique shortest path routing (USPR) case, splitting is not allowed and all SP-graphs must be rooted forests. So if \( a := (i,j) \in \bar{A}^l \), then all other arcs emanating
from \(i\) in \(\bar{A}\). This implies that almost all solutions are improving. The following kind of solution with two cycles with destination \(l\) and \(l'\), respectively, is the only exception

\[
\left\{ \{C_{\bar{l}}^l = F_{\bar{l}}^l \cup B_{\bar{l}}^l\}, \{C_{\bar{l}}^{l'} = F_{\bar{l}}^{l'} \cup B_{\bar{l}}^{l'}\} \right\},
\]

where \(B_{\bar{l}}^l = F_{\bar{l}}^l\) is a path from \(i\) to \(l\) and \(F_{\bar{l}}^{l'} = B_{\bar{l}}^{l'}\) is the destination arc \(a := (i, l) \in \bar{D}\).

This yields as a special case that all solutions are improving in the USPR case when all SP-graphs are spanning, cf. [62] or [58, Paper IV, Lemma 11].

If some restrictions on the families of cycles are considered, it is possible to obtain more structure on the classes, which often makes them considerably more tractable. Below the classes of binary and unitary families of cycles are considered, where the nonzero flow in cycles are restricted to be 1.

### 6.2.2 The Binary, Unitary and Simplicial Structures

Let \(\theta\) be a solution to (6.7) and let \(C\) and \(x\) be an associated family of cycles and cycle flows, respectively. Since the feasible region of (6.7) is a polyhedral cone, any rational solution may be scaled such that \(\theta\) and \(x\) become integral. In the following, assume that this integral scaling has been performed such that the greatest common divisor of the elements in the \(x\) vector is 1. Hence, the solution is integral and in a natural sense also minimal. It is now natural to define binary and unitary solutions.

**Definition 6.6**

Let \(\theta\) be a solution to (6.7) and \(C\) and \(x\) be an associated family of cycles and cycle flow, respectively. Then \(\theta\) and \(C\) are called a binary solution and binary cycle family, respectively, if \(x\) is 1 for all \(a\) and \(l\).

**Definition 6.7**

If \(\theta\) is a solution to (6.7), then \(\theta\) is called unitary if \(\theta^a_l \in \{-1, 0, 1\}\) for \(a \in \bar{A}\) and \(l \in L\).

A cycle family corresponding to a unitary solution is also called a unitary cycle family.

**Proposition 6.3**

If \(\theta\) is a unitary solution to (6.7), then it is a binary solution to (6.7).

**Proof:** Consider a fixed \(l \in L\). Since all \(\theta^a_l \in \{-1, 0, 1\}\) for \(a \in \bar{A}\), the induced graph is Eulerian and it is simple. Any decomposition of this graph into simple cycles yields a feasible collection \(C^l\) in (6.10). Applying this for all \(l \in L\) and setting the associated \(x\) to 1 completes the proof.

When a binary (or unitary) solution is represented as a cycle family, \(C\), and a flow solution, \(x\), in the representation in (6.10), it suffices to provide only the cycles where \(x^k_l\) is 1, therefore \(x\) is superfluous and will in the following be omitted for all binary solutions. The binary solution \(\theta\) will only be represented by

\[
C^l = \left\{ C_k^l = F_k^l \cup B_k^l \mid B_k^l \subseteq A^l \cup \bar{D}^l, \quad k = 1, \ldots, K^l \right\}.
\]

Given this cycle family, (6.12) may be specialized and \(\theta\) can be obtained from \(C\) via
\[ \theta^l_a = |F^l_a| - |B^l_a|, \quad (6.25) \]

where

\[ F^l_a = \{ k : a \in F^l_k \} \quad \text{and} \quad B^l_a = \{ k : a \in B^l_k \}. \quad (6.26) \]

In the unitary case, all non-zero flows have the same absolute value. Thus, it suffices to use a single (not necessarily simple) cycle to represent \( \theta^l \) for each \( l \in L \) and we can drop the cycle subscripts in (6.26). This yields the following specialization of (6.24).

\[ C = \{ C^l = F^l \cup B^l, B^l \subseteq A^l \}. \quad (6.27) \]

With this representation, \( \theta \) is obtained from \( C \) via the relation

\[ \theta^l_a = \begin{cases} 
1, & \text{if } (i, j) \in F^l, \\
-1, & \text{if } (i, j) \in B^l, \\
0, & \text{otherwise.} 
\end{cases} \quad (6.28) \]

**Remark 6.7.** Observe that (6.28) is obtained directly from (6.25) since the sets \( F^l_a \) and \( B^l_a \) are empty or singletons in the unitary case.

The final specialization of binary and unitary solutions considered is when the number of cycles that use an edge, i.e. an undirected arc, is at most two. These solutions are called simplicial because of their close connection to simplicial 2-complexes, see [14] for an introduction to topology and simplicial complexes.

For a unitary solution to (6.7) represented as in (6.27), we drop the destination superscripts of the destination index sets, \( F_a \) and \( B_a \), in (6.26) to obtain

\[ F_a = \{ l : a \in F^l \} \quad \text{and} \quad B_a = \{ l : a \in B^l \}. \quad (6.29) \]

We obtain the following definition of simplicial solutions.

**Definition 6.8**

Let \( \theta \) be a unitary solution to (6.7) represented by \( C \) and let \( F_a \) and \( B_a \) be defined as in (6.29). Then, \( \theta \) (and \( C \)) is simplicial if

\[ |F_a \cup B_a| + |B_a \cup B_a| \leq 2, \quad \text{for all } a := (i, j), \, \bar{a} := (j, i) \in \bar{A}. \quad (6.30) \]

**Remark 6.8.** Observe that the capacity constraint in (6.7) yields that \( |F_a| \leq |B_a| \leq 1 \). Further, in the saturating case, Equation (6.30) is always satisfied at equality for all arcs in \( C \) and \( |F_a| = |B_a| = 1 \) and \( |F_a| = |B_a| = 0 \), or vice versa.

We analyze the class of simplicial solutions more thoroughly in Chapter 7, including the important subclass of solutions that involve at most two destinations in Section 7.1. For examples of infeasible structures we refer to e.g. Examples 1.1 (page 2), 2.1 (page 15), 4.1 (page 47), 6.2 (pages 82-83), 6.3 (page 84), 6.4 (page 85), 6.5 (page 86) and 7.1 (page 99).

Next, we briefly discuss the extremal structure of the cone induced by (6.7).
6.3 Extreme Rays and Generators

The set of feasible solutions to (6.7), denoted by $\Theta$, forms an open polyhedral cone. We consider a description of its closure, $\text{cl } \Theta$, via extremal solutions and a closely related concept that we refer to as generators.

6.3.1 Representation of Polyhedral Cones

The treatment of this subject here is very brief. We refer to the classical reference books \cite{199} and \cite{204} for more on polyhedral theory.

Let $C \subseteq \mathbb{R}^n$ be a polyhedral cone described as the intersection of a finite set of half spaces, i.e. $C := \{ x \in \mathbb{R}^n \mid Ax \geq 0 \}$, and let $L := -C \cap C = \{ x \in \mathbb{R}^n \mid Ax = 0 \}$ be its lineality space. When $C$ is pointed, i.e. $L = \{0\}$, it has a unique minimal representation (up to multiplication by positive scalars) as $C = \{0\} + \text{cone}\{y^{(1)}, \ldots, y^{(t)}\}$, where $y^{(1)}, \ldots, y^{(t)}$ are the extreme rays of $C$. If $C$ is not pointed, the pointed cone $C_0 = C \setminus L$ may be used to decompose $C$ into the orthogonal sum $C = L + C_0$. Given this decomposition one usually says that a ray is an extreme ray of $C$ if it is an extreme ray of $C_0$. The following theorem gives a characterization of the extreme rays of $C \setminus L$.

**Theorem 6.3 (\cite{68}, \cite{185})**

Let $C = \{ x \mid Ax \geq 0 \}$, $L = \{ x \mid Ax = 0 \}$ and $\dim L = d$. Then, $x \in C^0$ is an extreme ray of $C \setminus L$ if and only if there exist exactly $n - d - 1$ linearly independent rows $a^i$ of $A$ such that $a^i x = 0$.

For an arbitrary ray, $\lambda$ say, in $C$, the corresponding ray, $\lambda^1$ say, in $C_0$ is obtained via a projection. If $B$ is a basis of the lineality space $L$, then

$$\lambda^1 = (I - B' (BB')^{-1} B) \lambda.$$  \hspace{1cm} (6.31)

This implies that if $\lambda$ induces $n - d - 1$ linearly independent rows $a^i$ of $A$ such that $a^i x = 0$, then $\lambda^1$ is an extreme ray of $C$, see \cite{185}.

6.3.2 Extreme Rays and Generators of the Closure of $\Theta$

We consider the extremal structure of $\text{cl } \Theta$ and denote its lineality space by $\Theta^L$. It follows from Section 6.2.1 that $\Theta^L \subseteq \Theta^0$. Equality holds in some special cases, for instance when all SP-graphs are spanning.

To obtain an orthogonal decomposition of $\text{cl } \Theta$ into a non-improving affine subspace and a polyhedral cone with improving solutions, we define

$$\Theta^\perp = (\text{cl } \Theta) \cap (\Theta^L)^\perp.$$  \hspace{1cm} (6.32)

This yields

$$\text{cl } \Theta = \Theta^L + \Theta^\perp.$$  \hspace{1cm} (6.33)

The following example illustrates that an extreme ray of $\text{cl } \Theta$ may have some undesirable properties if used, as is, to form a valid inequality in a BSP problem.
Figure 6.1: An instance where it is more natural to represent \( \text{cl} \ \Theta \) by generators that are not extreme rays of \( \text{cl} \ \Theta \). The solid, dashed and dotted arcs describe the SP-arcs for the SP-graph with destination 4, 3 and 1 respectively.

Example 6.1
Consider the SP-graphs in Figure 6.1. It is easily verified that the lineality space, \( \Theta^L \), is generated by the vector \( \bar{\theta} \) with nonzero components

\[
\bar{\theta}_{12}^3 = \bar{\theta}_{23}^3 = \bar{\theta}_{13}^4 = 1, \quad \text{and} \quad \bar{\theta}_{12}^4 = \bar{\theta}_{23}^4 = \bar{\theta}_{13}^3 = -1.
\] (6.34)

To span the cone \( \text{cl} \ \Theta \), another solution, not in the lineality space, is required. One such solution is \( \bar{\theta} \) given by the nonzero components

\[
\bar{\theta}_{12}^1 = \bar{\theta}_{23}^4 = \bar{\theta}_{34}^1 = 1, \quad \text{and} \quad \bar{\theta}_{12}^4 = \bar{\theta}_{23}^1 = \bar{\theta}_{34}^4 = -1.
\] (6.35)

All solutions in \( \text{cl} \ \Theta \) are generated by non-negative combinations of \( \pm \bar{\theta} \) and \( \bar{\theta} \). An alternative to using \( \bar{\theta} \) as a generator is to use

\[
\bar{\theta} + \lambda \delta \theta
\] (6.36)

for some \( \lambda \in \mathbb{R} \). Two "natural" choices for \( \lambda \) are 0 and -1, which yields

\[
||\bar{\theta} + \lambda \delta \theta||_1 = 1.
\]

These generator candidates are depicted in Figure 6.2.

Figure 6.2: The two (natural) candidates of the additional generator of \( \text{cl} \ \Theta \). The solution on the left corresponds to \( \lambda = 0 \) in (6.36) and the one on the right to \( \lambda = -1 \).

The generator on the left in Figure 6.2 has the additional advantage that it minimizes the support which can be of importance in the BSP problem context, see Part III.

Instead consider the orthogonal decomposition of \( \text{cl} \ \Theta \) into \( \Theta^L \) and \( \Theta^\perp \). The extreme ray of \( \Theta^\perp \) is obtained via a projection onto \( \Theta^\perp \). This yields \( \lambda = 1/6 \). Hence, the orthogonal decomposition is

\[
\text{cl} \ \Theta = \left\{ \theta \mid \theta = a\bar{\theta} + b \left( \bar{\theta} + \frac{1}{6} \delta \theta \right) , \text{ for some } a \in \mathbb{R} \text{ and } b \in \mathbb{R}_+ \right\}.
\] (6.37)

Since the support of the solutions in Figure 6.2 are strictly contained in the support of this extreme ray it is easy to show that the induced valid inequalities are dominated. Hence, in a BSP problem context, this example shows that other decompositions of \( \text{cl} \ \Theta \) than the orthogonal decomposition should be considered when the lineality space is non-empty.
Motivated by Example 6.1, we will not use the ordinary extreme rays of $\text{cl } \Theta$. Instead, the closely related concept of a generator of $\text{cl } \Theta$ is defined as follows.

**Definition 6.9**
Let $\theta$ be a solution in $\text{cl } \Theta = \Theta^L + \Theta^\perp$ and $\theta^{(1)}, \ldots, \theta^{(t)}$ the extreme rays of $\Theta^\perp$. Then $\theta$ is a generator of $\text{cl } \Theta$ if

$$\theta = \theta^L + s \theta^{(i)},$$

(6.38)

for some point $\theta^L \in \Theta^L$, some scalar $s > 0$ and some $i \in \{1, \ldots, t\}$.

Each generator induces a whole (equivalence) class of generators. In the following, no distinction between different representative for the class is made, but we do emphasize that a solution of minimal support is to be preferred when it is not hard to find.

Let us compare the definition of a generator to an ordinary extreme ray to see when it is beneficial to use generators instead of extreme rays. The following observation is straightforward.

**Proposition 6.4**
Let $B$ be a basis of $\Theta^L$ then $\theta$ is generator of $\text{cl } \Theta$ if and only if

$$\theta^0 = \left( I - B'(BB')^{-1}B \right) \theta$$

(6.39)
is an extreme ray of $\Theta^\perp$.

A few comments are in place: first, the "only if" direction implies that any extreme ray is a generator, second, a solution that is not a generator cannot be an extreme ray and finally, when the lineality space contains only the origin, then all generators are extreme rays. The difference between a generator and an extreme ray is depicted in Figure 6.3.

![Figure 6.3: Illustration of our definition of a generator versus the definition of an extreme ray. The cone $C$ under consideration is generated by positive linear combinations of the vectors $x$, $y$ and $z$. The lineality space is spanned by the vectors $+z$ and $-z$ and $C^\perp$ is spanned by $x$ and $y$. This yields that $x$ and $y$ are the "ordinary" extreme rays. With our definition, any solution in one of the two (hyper)planes sketched is a generator, e.g. $x + z$ or $y$](image-url)

The advantage of using generators is that it allows us to essentially ignore the lineality space and the projection in (6.31). The emphasis can then be put on the actual conflict. In the following we will often assume that the lineality space is empty which implies that
the definitions coincide. Since generators are used, we got our back covered, so to speak and do not have to worry about cases such as Example 6.1.

6.3.3 Irreducible Solutions in the Closure of $\Theta$

Every generator (or extreme ray) corresponds to a routing conflict. However, a generator is not necessarily minimal, w.r.t. inclusion, even if it is of minimal support. Intuitively, a conflict is irreducible if the involved SP-arcs cannot be used to form a smaller conflict, i.e. a conflict that involves a subset of these SP-arcs. Consider the following example.

--- Example 6.2

Figure 6.4: A (reducible) conflict that corresponds to a generator but contains a smaller conflict. The SP-arcs of three SP-graphs are induced by solid, dashed and dotted arcs, respectively. Together, they induce a conflict. The SP-graphs corresponding to the solid and dotted arcs alone induce a smaller conflict.

Consider the solution induced by the family of SP-graphs in Figure 6.4. The SP-graphs with destinations $l'$, $l''$, and $l'''$, respectively, are given by solid, dashed and dotted arcs. Assuming that these are the only arcs that are involved in some conflict, it is clear that the solution $\theta$ with the following nonzero components is a generator

$$
\begin{align*}
\theta'_{14} &= \theta''_{21} = \theta''_{32} = \theta''_{54} = \theta''_{36} = \theta''_{65} = 1 \\
\theta'_{36} &= \theta''_{54} = \theta''_{65} = \theta''_{14} = \theta''_{21} = \theta''_{25} = \theta''_{32} = -1.
\end{align*}
$$

A smaller generator is induced by $\tilde{\theta}$, with the following nonzero components,

$$
\begin{align*}
\tilde{\theta}'_{25} &= \tilde{\theta}'_{32} = \tilde{\theta}''_{36} = \tilde{\theta}'''_{65} = 1 \\
\tilde{\theta}'_{36} &= \tilde{\theta}'_{56} = \tilde{\theta}''_{25} = \tilde{\theta}''_{32} = -1.
\end{align*}
$$

Note that the support of $\theta$ is not contained in the support of $\tilde{\theta}$, or vice versa, while the support of the negative components in $\tilde{\theta}$ is contained in the support of the negative components in $\theta$. The solution $\tilde{\theta}$ is in a sense reducible.

---

Remark 6.9. At first, Example 6.2 may seem odd in view of [117]. Indeed, the main result in [117] is that each irreducible subsystem of a linear system, $Ax \leq b$, corresponds to an extreme ray of the so called alternative polyhedron. In our example, both generators are indeed irreducible subsystems of the associated linear system induced by (6.7) which is our unbounded version of the alternative polyhedron. When we say irreducible conflict, we only consider the SP-arcs involved rather than the whole irreducible subsystem of (in)equalities. Thus, a reducible conflict corresponding to a generator has a subset of SP-arcs that induces another conflict and therefore another irreducible subsystem. In terms of extreme rays, we only compare the negative supports of solutions, see Definition 6.10.
Example 6.2 shows that the subset of generators that are minimal, or irreducible, is of particular importance. To describe the set of feasible routing patterns in Chapter 10 it is sufficient to forbid all sub-patterns arising from irreducible generators.

To define an irreducible conflict only the negative valued variables matter. Let

\[ \text{supp}(\theta^-) = \{(a, l) \in A \times L \mid \theta^a_l < 0\} \]  

(6.42)
denote the set of indices for the negative valued variables in \( \theta \).

**Definition 6.10**

Take \( \theta \in cl \Theta \). Then, \( \theta \) is irreducible if there does not exist a \( \tilde{\theta} \in cl \Theta \) such that

\[ \text{supp}(\tilde{\theta}^-) \subset \text{supp}(\theta^-) \quad \text{and} \quad \tilde{\theta} \neq s\theta, \text{ for some } s > 0. \]  

(6.43)

It follows directly that a solution that is not a generator is reducible.

**Proposition 6.5**

The set of irreducible solutions is a subset of the set of generators.

---

**Example 6.2: continued**

![Diagram of an irreducible conflict](image)

**Figure 6.5:** An irreducible conflict is obtained from the conflict in Figure 6.4 in Example 6.2 by subdividing arc (2, 5). This subdivision procedure is in general applicable to construct an irreducible conflict from a reducible generator, see Remark 7.12 and Section 7.4.

It can be verified that the conflict in Figure 6.5 is irreducible.

A characterization of irreducible solutions is given in Chapter 7.

### 6.4 The Hierarchy of Infeasible Structures

We have introduced five classes of solutions. They are, in decreasing order of generality, referred to as: general, binary, unitary, simplicial and valid cycle solutions. In this section, we prove that these sub-classes are strictly nested. The collections of all cycle families that correspond to an irreducible generator are denoted by, respectively,

- \( S^2 \) for simplicial solutions that involve at most two destinations,
- \( S \) for simplicial solutions,
- \( U \) for unitary solutions,
- \( B \) for binary solutions,
- \( G \) for general solutions.
Similarly, the subscript $I$ is used on the collections above to denote the collection of instances where the least complicated solution belongs to the corresponding collection.

We have the following two theorems.

**Theorem 6.4**
The cycle family collections $S^2, S, U, B$, and $G$ are strictly nested, i.e.

$$S^2 \subset S \subset U \subset B \subset G.$$  \hspace{1cm} (6.44)

**Theorem 6.5**
The cycle family instance collections $S^2_I, S_I, U_I, B_I$, and $G_I$ are strictly nested, i.e.

$$S^2_I \subset S_I \subset U_I \subset B_I \subset G_I.$$  \hspace{1cm} (6.45)

**Proof of Theorem 6.4 and 6.5**: All inclusions hold trivially from the definitions in Section 6.2. Strictness follows from Example 6.3-6.6 presented below. By construction, all these examples are created to have unique solutions, hence strictness applies both to structures (Theorem 6.4) and instances (Theorem 6.5).

To prove that the sub-collections in Theorem 6.4 and 6.5 are proper, it suffices to provide a single example of an instance where the least complicated conflict belongs to the corresponding class of solution. For readability, we presented the examples in increasing order of complexity of the infeasible structure.

**Remark 6.10.** In the examples below (and elsewhere) there is not necessarily a connection between the index of the SP-graph and the destination node unless this is explicitly mentioned. This is to be able to refer to nodes, arcs and SP-graphs conveniently. To keep the interpretation of the index as the destination it may be necessary make an example larger and to force them to contain arcs that do not matter for the actual structure under consideration. Note that this impose no loss of generality: an additional node not in the graph may be added with the sole purpose of being the destination of a given SP-graph so that no new conflict is induced.

Earlier in the thesis we have given several examples of instances with valid cycles, i.e. examples of elements in $S^2$ and $S^2_I$. In [59] an infeasible instance with a simplicial solution and no valid cycle is given. We give a smaller example here (actually, it is a minimal example with a simplicial solution and no valid cycle).

--- Example 6.3 ---

Consider the three intrees in Figure 6.6 and the induced graph, i.e. the union of the arcs in the intrees. It is straightforward to verify that there is no solution for this instance that uses only two commodities, either by inspection or by the algorithm in Section 7.1.4.

A feasible solution to (6.7) is induced by the family of cycles in Figure 6.7, i.e.

$$C = \{C^1 = F^1 \cup B^1, C^3 = F^3 \cup B^3, C^5 = F^5 \cup B^5\},$$  \hspace{1cm} (6.46)

where
6.4 The Hierarchy of Infeasible Structures

Figure 6.6: The SP-graphs \((A_1, \bar{A}_1), (A_3, \bar{A}_3)\) and \((A_5, \bar{A}_5)\), respectively. The figure should be interpreted as follows. A drawn arc represents that the arc is an SP-arc in the associated SP-graph. Arrows not drawn are non-SP-arcs.

\[ F^1 = \{(2, 1), (3, 2)\}, \quad B^1 = \{(3, 4), (4, 1)\}, \]
\[ F^3 = \{(2, 5), (4, 1)\}, \quad B^3 = \{(2, 1), (4, 5)\}, \]
\[ F^5 = \{(3, 4), (4, 5)\}, \quad B^5 = \{(2, 5), (3, 2)\}. \]

(6.47)

A solution \(\theta \in \Theta\) is obtained from \(\mathcal{C}\) via (6.28). Further, \(\Theta\) is generated by the extreme ray induced by \(\theta\).

Figure 6.7: The cycles \(C^1 = F^1 \cup B^1\), \(C^3 = F^3 \cup B^3\) and \(C^5 = F^5 \cup B^5\), respectively. A dashed arc \((i, j)\) is a forward arc, i.e. \((i, j) \in F^l\) and \(\theta_{ij}^l > 0\). A solid arc \((i, j)\) is a backward arc, i.e. \((i, j) \in B^l\) and \(\theta_{ij}^l < 0\).

The minimality of this example w.r.t. number of SP-graphs, and then, number of nodes follows since the absence of a valid cycle implies that there must be at least 3 SP-graphs. It is easy to rule out all cases with at most 4 nodes. Since our ingraphs are trees, minimality is obtained. The next example has more SP-graphs and fewer nodes.

To show that all unitary solutions are not simplicial we use an example due to Bley. This example is in our opinion very beautiful; it is very small, symmetric, planar, seemingly simple, but yet, rather complex.

--- Example 6.4 ---

A translation of the example in Figure 5.4 on page 77 in [46] to SP-graphs gives the 4 SP-graphs in Figure 6.8. The SP-arcs for these graphs are

\[ A^1 = \{(4, 3), (3, 1)\}, \quad A^2 = \{(3, 4), (4, 2)\}, \]
\[ A^3 = \{(2, 1), (1, 3)\}, \quad A^4 = \{(1, 2), (2, 4)\}. \]

(6.48)
Each of these SP-arc sets uniquely induces an oriented circuit in the underlying undirected cycle (1 2 4 3) where the associated SP-arcs are used backwards. By inspection, it follows that there is no simplicial solution.

Next we show that all binary solutions are not unitary. To accomplish this, a non-planar graph, very similar to a Möbius ladder is used. A cycle family can be obtained from an embedding of this graph into the simplest non-orientable surface, the Möbius strip.

**Example 6.5**

The graph in Figure 6.9 and the following four SP-graphs with SP-arc sets given by

\[
A^1 = \{(3, 8), (7, 2), (9, 4)\}, \\
A^2 = \{(1, 2), (5, 4), (7, 6), (9, 10)\}, \\
A^3 = \{(1, 10), (3, 2), (9, 8)\}, \\
A^4 = \{(3, 4), (5, 6), (7, 8)\}.
\]

will be used to obtain binary solution that is not unitary.

By construction, all nodes in this graph have either in or outdegree zero, this makes it easy to deduce which cycles are possible for the different SP-graphs. In Figure 6.10, the graph have been embedded into the Möbius strip. From this drawing it is possible to
verify that the only circuits that obey the commodity specific flow bounds are the ones sketched out. The five circuits are

\[
\begin{align*}
C_1^1 &= B_1^1 \cup F_1^1 = \{(3, 8), (7, 2)\} \cup \{(3, 2), (7, 8)\}, \\
C_2^1 &= B_1^2 \cup F_1^2 = \{(3, 8), (9, 4)\} \cup \{(3, 4), (9, 8)\}, \\
C_2 &= B_2 \cup F_2 = A^2 \cup \{(7, 2), (9, 4), (1, 10), (5, 6)\}, \\
C_3 &= B_3 \cup F_3 = A^3 \cup \{(3, 8), (9, 10), (1, 2)\}, \\
C_4 &= B_4 \cup F_4 = A^4 \cup \{(3, 8), (7, 6), (5, 4)\}.
\end{align*}
\]

The family of cycles consisting of all these circuits is binary. Further, in the associated solution obtained via (6.25), the flow on arc \((3, 8)\) is \(-2\) for commodity 1, therefore this solution is not unitary. It remains to verify that there is no other solution in this instance. This is clear since there is a dependence between all five circuits such that there can be no flow in one circuit unless there is some flow in all other circuits.

\[\text{Figure 6.10: An embedding of the graph in Figure 6.10 in the Möbius strip. To obtain the surface, the sides labeled with A are glued together so that the arrows overlap. All oriented circuits, } C_1^1, C_2^1, C_2, C_3 \text{ and } C_4 \text{ have been sketched out with a dotted circle with an orientation that is consistent with the associated SP-graph.}\]

We mentioned the similarity of the graph in Figure 6.10 with a Möbius ladder. Note that the Möbius ladder with ten nodes, \(M_{10}\), is obtained if the arcs \((1, 6)\) and \((5, 10)\) are added. We did indeed initially create this example from the Möbius ladder with six nodes \(M_6\) (also known as the complete bipartite \(K_{3,3}\)). Note that \(M_6 = K_{3,3}\) is obtained if the outermost nodes 1, 5, 6 and 10 are contracted. However, the solution induced by that graph is reducible since the cycle \((2, 7, 4, 9)\) would induce a valid cycle. To obtain an irreducible solution two arc subdivisions were made to obtain the above instance, see further Section 7.4.

To complete the analysis it remains to provide an infeasible instance where a general solution is not binary. This, in a sense, shows that a least complicated solution to (6.7)
may in principle be arbitrarily complicated. Our construction is essentially based on two Möbius strips that are glued together.

Example 6.6

![Diagram](image)

**Figure 6.11**: An instance with six SP-graphs that induce a unique cycle family that is not binary.

In this "huge" example we consider the instance induced by the graph in Figure 6.11 with six SP-graphs. Each SP-graph is associated with a commodity. The commodities are divided into three classes.

The four commodities associated with dotted, square marked dotted, dashed and square marked dashed arcs are similar and will be treated essentially equivalently. They are referred to as type-1 commodities. The commodity associated with square marked...
solid arcs is referred to as a type-2 commodity. Finally, the commodity associated with solid arcs is a type-2 commodity. It is this commodity that will make the solution in the example general instead of "just" binary. It will be associated with two circuits with different flow amounts.

The proof has two parts. First, we show that there is a solution based on a general family of cycles. Then, we show uniqueness, i.e. that there is no other family of cycles.

From the graph it is easily verified that each type-1 commodity has a unique oriented circuit that uses only the commodity specific arcs backwards. The only possible cycle for the commodity associated with dashed arcs is given in Figure 6.12. The three other type-1 commodities are obtained by symmetry.

Similarly, there is a unique oriented circuit that uses the upper square marked solid arcs backwards, it is sketched in Figure 6.12. The lower circuit is obtained by symmetry.

The only remaining commodity is the type-3 commodity associated with solid arcs. In this case there are several possible cycles and representation thereof. A natural representation of all cycles is in terms of three "basis" cycles: an outer cycle, an intermediate cycle and an inner cycle. These cycles are given in Figure 6.13.

It follows straightforward that a solution is obtained by sending:

- one unit of flow in the unique oriented circuit for each type-1 commodity,
- one unit of flow in the upper and lower oriented circuits for the type-2 commodity,
- one unit of flow in the outer and intermediate oriented circuits and two units of flow in the inner oriented circuit for the type-3 commodity.

By construction, all arcs not in the inner circuit carry one flow unit forwards and one backwards. In the inner circuit, the type-1 commodities carry one flow unit backwards on its own arc and forwards on the type-3 commodity arc. The type-3 commodity compensate by carrying two units of flow in the inner circuit.

The final task is to verify uniqueness. No type-1 commodity may carry flow unless there is a type-3 flow on all basis cycles associated with the type-3 commodity. Equivalently, both type-2 circuits requires type-3 flow on the outer and intermediate basis cycles. Finally, there can be no type-3 flow unless the associated type-1 and type-2 commodities are used. Hence, it is all or nothing, so to speak and the proposed solution is unique.

Figure 6.12: An illustration of the unique oriented circuit induced by using dashed arcs backwards and the the unique oriented circuit in the upper part of the graph induced by using square marked solid arcs backwards. Uniqueness is verified by selecting a tentative backward arc and considering the possible augmentations to a circuit that obeys the commodity specific flow bounds.
This example concludes the proof of Theorem 6.4 and 6.5.

In [59], a class of solutions called 3-valid cycles is defined, and it is asked whether the absence of 3-valid cycles is a sufficient condition for feasibility of (5.13). This class is the subset of the general solutions where all $x$ associated with the same destination have the same value.

**Corollary 6.2**

*Absence of 3-valid cycles is not a sufficient condition for the feasibility of (6.6).*

**Proof:** The instance in Example 6.6 yields a general solution that is not a 3-valid cycle since the flow in the inner circuit is 2 and the flow in the outer cycle is 1 for the SP-graph with solid arcs.

This completes our analysis and characterization of infeasible routing patterns. In the next chapter we consider the class of simplicial solutions further.
Understanding infeasibility is the key to deriving valid inequalities that prohibit the associated infeasible shortest path routing (SPR) structures. A comprehensible class of infeasible structures is the class of simplicial structures and in particular the subclass involving at most two SP-graphs referred to as valid cycles.

The aim of this chapter is to get a deeper understanding of the simplicial structure. This objective is approached by showing how graph embeddings yield infeasible simplicial structures. The connection gives a powerful tool for constructing interesting example instances which can be important to get a profound understanding of infeasibility. Based on an embedding that gives an infeasible structure, we show that the dual graph encodes a dependency relation between the cycles in the structure when its edges are given appropriate directions. The resulting dependency graph is then used to derive our main result: a characterization of simplicial structures that correspond to generators and irreducible solutions in the polyhedral cone arising from the Farkas’ system in the previous chapter.

Outline

The class of valid cycles is considered in Section 7.1 and the general case in Section 7.2. Then, the dependency graph is introduced in Section 7.3 where we also show how to form a simplicial cycle family with a specific dependency relation. In Section 7.4, we use the dependency graph to derive a purely graph theoretical characterization of irreducible simplicial generators.

7.1 The Valid Cycle Structure

Valid cycles form a sub-class of simplicial solutions that involve at most two destinations, they generalize the infeasible structure in [61, 62] with the same name. This class is very important in practice and due to the size restriction, it forms a very comprehensible class of solutions. Some of its sub-classes have been considered in the literature, e.g. in [25, 46, 85, 218]. We discuss the relation of our class to the literature on page 96. We
have already given a few examples of valid cycles, see e.g. Examples 1.1 (page 2), 2.1 (page 15) and 4.1 (page 47).

The practical importance of valid cycles is indisputable. Our computations on the core problem in Part III, i.e. model (9.4) on page 130, showed that conflicts with two SP-graphs are by far the most important conflicts. This is in line with the experiments in [61] where 99% of the conflicts in their compatibility version of the inverse shortest path routing (ISPR) problem are explained by valid cycles.

Restricting the number of SP-graphs to two implies that infeasibility is much easier to analyze. It is possible to determine if two SP-graphs induce a valid cycle in linear time, see [62] and the algorithm induced by Theorem 7.1. The separation problem for valid cycle inequalities can also be solved in polynomial time, see Section 13.3. Motivated by the clarity and practical importance, we examine valid cycles more thoroughly.

From Chapter 6 it follows that a simplicial solution, \( \theta \in \Theta \), involving at most two SP-graphs can be represented as

\[
C = \left\{ C^{iv} = F^{iv} \cup B^{iv}, \; C^{iv'} = F^{iv'} \cup B^{iv'} \right\},
\]

(7.1)

where \( C^{iv} \) and \( C^{iv'} \) are undirected cycles and \( B^{iv} \subseteq A^{iv} \cup \tilde{D}^{iv} \) and \( B^{iv'} \subseteq A^{iv'} \cup \tilde{D}^{iv'} \). Due to the capacity constraint, the forward and backward arc sets must satisfy

\[
F^{iv} \subseteq B^{iv'} \quad \text{and} \quad F^{iv'} \subseteq B^{iv}.
\]

If \( \theta \) is saturating, then \( F^{iv} = B^{iv'} \) and \( F^{iv'} = B^{iv} \) and we represent \( C \) in (7.1) by an undirected cycle \( C = F \cup B \) where \( F = F^{iv} = B^{iv'} \) and \( B = F^{iv'} = B^{iv} \). If \( \theta \) is non-saturating, the union \( E = (B^{iv'} \setminus F^{iv}) \cup (B^{iv} \setminus F^{iv'}) \) induces an Eulerian graph.

**Definition 7.1**

The solutions in \( \text{cl} \Theta \) that involve at most two SP-graphs are denoted by \( \Theta^2 \).

First, valid cycles with one SP-graph are considered and then, saturating and non-saturating cycles with two SP-graphs.

### 7.1.1 Valid Cycles with a Single SP-graph

An infeasible routing pattern with a single SP-graph \((A^{i} \cup \tilde{D}^{i}, \tilde{A}^{i})\), must be a cycle where the aggregated flow on all arcs are non-positive. This gives two cases: either, (1) the cycle is a directed cycle in \( A^{i} \cup \tilde{D}^{i} \), or (2) it is formed by a path consisting of arcs only in \((A^{i} \cup \tilde{D}^{i}) \cap \tilde{A}^{i}\), i.e. a path that is required to be a shortest path, and prohibited to be a shortest path. This can be seen as a cycle formed by parallel arcs. In a sense, both the cases above are absurd.

For completeness, we state when the solutions induce generators. In the former case, the flow on all arcs in the cycle is negative and the solution is non-saturating. The cycle induces a generator if and only if it is simple. In the latter case, the flow on an arc \( a \) is negative when it is associated with \( A^{i} \cup \tilde{D}^{i} \) and positive when associated with \( \tilde{A}^{i} \). This solution corresponds to a generator if and only if it involves a single arc. Note that a generator can be represented by a simple undirected cycle \( C = F \cup B \), i.e. as the union of its forward arcs and backward arcs.

In the following, we assume that \((A^{i} \cup \tilde{D}^{i}) \cap \tilde{A}^{i} = \emptyset\).
7.1 The Valid Cycle Structure

7.1.2 Saturating Solutions with Two SP-Graphs

Let \( \theta \in \Theta^2 \) be a solution that corresponds to a generator of \( \Theta \) that involves two SP-graphs with destination \( l' \) and \( l'' \). Without loss of generality (W.l.o.g.), assume that \( \theta \) has been scaled, so that all components are -1, 0 or 1. Since \( \theta \) is saturating, its cycle family representation in (7.1) reduces to \( C = F \cup B \), where

\[
F = F'' = B'' \subseteq A'' \cup \bar{D}'
\quad \text{and} \quad
B = F'' = B'' \subseteq A'' \cup \bar{D}'',
\]

i.e. the forward arcs for one destination are the backward arcs for the other. Observe that \( C = F \cup B \) becomes a directed cycle if the arcs in \( F \) are used forwards and the arcs in \( B \) are used backwards. The relation to \( C \) induced by \( C = F \cup B \) is given by

\[
\theta_a = -\theta_a'' = \begin{cases} 
1, & \text{if } a \in F, \\
-1, & \text{if } a \in B, \\
0, & \text{otherwise}. 
\end{cases}
\]  

(7.4)

**Remark 7.1.** Note that this representation also includes the case with a single SP-graph, e.g. by \( F = \emptyset \) for directed cycles.

By construction, the valid cycle induced by \( C = F \cup B \) is feasible. We apply Proposition 6.2 to obtain conditions for when it is improving.

**Proposition 7.1**

Let \( \theta \) be a saturating generator of \( \Theta^2 \) associated with \( C = F \cup B \). If \( F \subseteq A'' \cup \bar{D}' \) and \( B \subseteq A'' \cup \bar{D}'' \), then \( \theta \) is improving if and only if \( F \cap \bar{A}'' \neq \emptyset \) or \( B \cap \bar{A}' \neq \emptyset \).

**Proof:** The relation in (7.4) yields \( \theta_a' > 0 \) for \( a \in F \) and \( \theta_a'' > 0 \) for \( a \in B \). Hence, the result follows immediately from Proposition 6.2; \( \theta \in \Theta^\circ \) is improving if and only if there is a commodity \( l \) and an arc \( a \) such that \( a \in \bar{A} \) and \( \theta_a > 0 \).

**Proposition 7.2**

Let \( A^L \) be a family of SP-graphs. If \( L \) contains two destinations, \( l' \) and \( l'' \) say, such that \( A'' \cup \bar{D}' \) and \( A'' \cup \bar{D}'' \) induce a saturating valid cycle, then \( A^L \) is not partially realizable.

Our definitions of feasible and improving cycles above generalize valid cycles as introduced in [61, 62] since we use a generalization of SP-graphs where D-arcs are allowed.

The structure of feasible cycles may be described more in detail. Recall that the arc sets \( F \) and \( B \) can be decomposed into path segments. Let \( K \) be the number of segments, then the decomposition becomes

\[
F = \bigcup_{p=1}^K \overrightarrow{P}_p \quad \text{and} \quad B = \bigcup_{p=1}^K \overleftarrow{P}_p.
\]  

(7.5)

Since the path segments are alternating, \( C \) can also be described as

\[
C = \overrightarrow{P}_1 \overleftarrow{P}_1 \cdots \overrightarrow{P}_K \overleftarrow{P}_K.
\]  

(7.6)

Assume that no arc emanates from \( l' \) in \( A'' \) or from \( l'' \) in \( A'' \). Then, every destination arc must be the last arc in a path segment and such a segment must end at the destination.
Figure 7.1: A template for saturating valid cycles with $K = 3$ path segments for each SP-graph. A solid (dashed) arrow is associated with the SP-graph with destination $l' (l'')$. A curly (straight) arrow represent a path segment (destination arc). For the saturating valid cycle $C = F \cup B$, all solid (dashed) arcs are in $F$ ($B$).

The decomposition implies that a saturating generator obtained from two SP-graphs can be described by a template as in Figure 7.1. The interpretation is as follows. The solid and dashed arrows are associated with the SP-graph with destination $l'$ and $l''$, respectively. A curly arrow represents a path segment with arbitrarily many arcs (including 0 and 1) and the ordinary arrows are destination arcs. Thus, when $C = F \cup B$, all arcs in the solid and dashed path segments are in $F$ and $B$, respectively.

7.1.3 Non-Saturating Solutions with Two SP-Graphs

Let $\theta \in \Theta^2$ be a non-saturating generator associated with the SP-graphs with destination $l'$ and $l''$. Assume that $\theta$ has been scaled so that all components are $-1, 0$ or $1$. Then,

$$C = \left\{ C'' = F'' \cup B'' \right\}, \quad (7.7)$$

Since $\theta$ is non-saturating, $F' \neq B''$ and $F'' \neq B'$, thus

$$E = E' \cup E'' = \left( B' \setminus F'' \right) \cup \left( B'' \setminus F' \right) \quad (7.8)$$

forms an Eulerian graph that becomes directed when the orientation induced by the labelings is taken into account. We represent $\theta$ by the non-saturating valid cycle $C = F \cup B$, i.e. the union of all backward arcs, along with the cycle, $E' \cup E''$. This yields,

$$C'' = (F \cup B) \setminus E'' = C \setminus E'', \quad \text{and} \quad C''' = (F \cup B) \setminus E' = C \setminus E', \quad (7.9)$$

and the relation to $\theta$ is given by

$$\theta''_a = \begin{cases} 1, & \text{if } a \in F \setminus E'', \\ -1, & \text{if } a \in B, \end{cases} \quad \text{and,} \quad \theta'''_a = \begin{cases} -1, & \text{if } a \in F, \\ 1, & \text{if } a \in B \setminus E', \\ 0, & \text{otherwise}. \end{cases} \quad (7.10)$$

Since $C = F \cup B$ induces a non-saturating solution, $\theta$, it is improving, see Proposition 6.1, hence $\theta \in \Theta$. As above, the absence of non-saturating valid cycles is a necessary condition for realizability.

Proposition 7.3

Let $\mathcal{A}^l$ be a family of SP-graphs. If $L$ contains two destinations, $l'$ and $l''$, say, such that $\mathcal{A}^{l'} \cup \mathcal{D}'$ and $\mathcal{A}^{l''} \cup \mathcal{D}''$ induce a non-saturating valid cycle, then $\mathcal{A}^C$ is not partially realizable.
We apply the decomposition of $F$ and $B$ into path segments as above. In the same manner, this yields the template for non-saturating valid cycles on the left in Figure 7.2. Here it is optional to include one or both of the path arcs between a pair of nodes. All path segments ending in the associated destination may end with a destination arc. Aside from this, the interpretation of the figure is as in the saturating case, see the explanation of Figure 7.1 on page 94. Observe that a template for a non-saturating valid cycle can induce an exponential number of valid cycles. A realization of the template is given on the right in Figure 7.1.

Remark 7.2. Three path segments are used in both directions in the realization in Figure 7.1. This implies that there are $2^3 = 8$ saturating solutions that use a subset of the arcs in this realization, and hence the solution is reducible. See also Proposition 7.4 and Corollary 7.1 and Section 8.3 in Chapter 8, in particular Theorems 8.2 and 8.3.

The following observation is a corollary to Theorem 8.2 in Chapter 8.

Proposition 7.4
If there exists a non-saturating valid cycle, then there exists a feasible (but, not necessarily improving) cycle family that involves two SP-graphs.

When a family of SP-graphs is completed, the saturating solution obtained from a non-saturating valid cycle in Proposition 7.4 becomes improving, and thus, a valid cycle.

Corollary 7.1
Let $l'$ and $l''$ be two destinations such that the associated SP-graphs have been completed and none of their SP-arc sets contain a directed cycle, i.e. $A = A^l \cup \bar{D}^l \cup A^l_{\leq l}$ and $A^l \cup D^l$ is acyclic for $l \in \{l', l''\}$. Then, if these SP-graphs form a non-saturating valid cycle they also form a saturating valid cycle.

Proof: Let $C = F \cup B$ and $E = E' \cup E''$ represent the non-saturating valid cycle. Consider the directed cycle $E$, by assumption $E$ is not contained in $A^l \cup \bar{D}^l$ or $A^l_{\leq l}$ and since the SP-graphs have been completed at least one arc in every directed cycle must
be a non-SP-arc. By symmetry, let it be in \( F \) and consider the undirected cycle \( \tilde{C} = \tilde{F} \cup \tilde{B} \) where \( \tilde{F} = F \) and \( \tilde{B} = B \setminus E' \). Then \( \tilde{C} \) is feasible. Since \( \tilde{F} \cap \tilde{A}' \neq \emptyset \), Proposition 7.1 implies that \( \tilde{C} \) is improving.

The above results are similar to Theorem 5 in [60] (page 518) which states that there is no (improving) solution associated with two SP-graphs unless there is a valid cycle. Our statements are more accurate and general. Indeed, a more general case is considered and a distinction between saturating and non-saturating solutions is made. (Not considering non-saturating solutions is actually a flaw in the derivation of the theorem in [60].)

The coherency of the case with two SP-graphs has allowed us to describe and analyze all generators. This class of solutions will be very important when valid inequalities are derived in Part III (recall that valid cycles seems to explain most conflicts in practice). A crucial task is therefore to separate valid cycle inequalities. The foundation for such an algorithm is to be able to decide if a pair of SP-graphs induces a valid cycle. This problem is solved in Section 7.1.4.

### On the Relation of Our Valid Cycles to Similar Structures

An account of the relation between saturating valid cycles and other classes of solutions discussed in the literature is given here.

First, consider the case when there are no destination arcs. In this case, several authors describe the following, in principle equivalent, classes of necessary conditions: e.g. subpath optimality in [25], the Bellman property in [46] and subpath consistency in [62]. A routing pattern that fails to satisfy the above conditions yields a solution to (6.7) associated with two cycles that consist of two disjoint path segments, i.e. the simplest case of a saturating valid cycle. In [61, 62], the valid cycle can contain any number of path segments and corresponds to a our saturating valid cycles without destination arcs.

Infeasible structures involving destination arcs have in a sense been taken into account implicitly in an ad hoc manner in [218]. They present valid inequalities based on this class of solutions when there is only one path segment. Similar inequalities can be found in e.g. [46] and [85]. The case with two or more path segments is not considered in any of the above papers. In the unique shortest path routing case, considered in [46], it suffices to prohibit conflicts with a single path segment since all solutions with more path segments are reducible.

To the best of our knowledge, the class of non-saturating solutions has not been considered earlier in the literature.

### 7.1.4 Algorithms to Find Generalized Valid Cycles

We describe an efficient algorithm for deciding if there exists a valid cycle among two SP-graphs. Our algorithm is a straightforward generalization of the algorithm in [62]. The (important) difference is that we include destination arcs, which implies that our algorithm finds more conflicts earlier.

The following observation, adapted from [62], translates straightforward to an algorithm that can be used to find a saturating feasible cycle formed by two SP-graphs, \((A^f \cup D_f', A'^f)\) and \((A'^\nu \cup D'^\nu, A'^\nu)\) say. If \( C = F \cup B \) represents a feasible cycle, then \( C \) becomes a directed cycle if all arcs in \( B \) are reversed. Therefore, construct the graph \( \tilde{G} \)
by adding all arcs in $A_l' \cup \bar{D}_l'$ and the reverse of all arcs in $A_l'' \cup \bar{D}_l''$. This construction yields the following lemma.

**Lemma 7.1**

A feasible cycle $C = F \cup B$ corresponds to a directed cycle in $\bar{G}$.

Hence, it suffices to consider directed cycles in $\bar{G}$. A simple directed cycle necessarily lies within a non-trivial strongly connected component (SCC), i.e. with at least 2 nodes. Therefore, it is sufficient to find and examine the SCCs of $\bar{G}$ to find a feasible cycle.

To determine if there is a saturating valid cycle, the arc sets $A_l'$ and $A_l''$ matters, since they are used to decide if the cycle is improving. From Proposition 7.1 it follows that a directed cycle $C$ in $\bar{G}$ corresponds to a valid cycle if and only if some arc in $\bar{C}$ associated with an SP-arc for one destination is a non-SP-arc for the other destination. This yields the following theorem as a generalization of the main theorem in [62].

**Theorem 7.1**

The SP-graphs $(A_l' \cup \bar{D}_l', \bar{A}_l')$ and $(A_l'' \cup \bar{D}_l'', \bar{A}_l'')$ induce a saturating valid cycle if and only if a strongly connected component of $\bar{G}$ (where arcs associated with $l''$ have been reversed) contains an arc $a := (i, j)$ such that $a' := (j, i) \in \bar{A}_l'$ or $a \in \bar{A}_l''$.

Theorem 7.1 translates straightforward to an algorithm for finding valid cycles. Indeed, find all SCCs, then examine the arcs in the each SCC w.r.t. the criteria in Theorem 7.1. When such an "improving" arc, $a := (i, j)$, has been found, find a cycle that contains this arc, e.g. by finding a path in the SCC from $j$ to $i$. The running time of this algorithm is $O(m + n)$ since the most expensive operations of finding the SCCs and finding a cycle runs in linear time, see e.g. [115, 209, 216]. The algorithm is given in more detailed pseudo-code in [73].

We have two remarks on this algorithm that can be very important in practice.

**Remark 7.3.** Let $a := (i, j)$ be an improving arc as described above. To form a valid cycle via $a$ and a path from $j$ to $i$, we propose to use a breadth first search since this gives a valid cycle of small cardinality.

**Remark 7.4.** In a cutting plane context it is desirable to find several violated cuts. Hence, it seems natural to use all improving arcs in all SCCs to find many valid cycles since each valid cycle gives a valid inequality. Further, from a cut selection perspective, valid cycles from different SCCs gives coordinated cuts since their normals are orthogonal, which is very desirable, see e.g. [2, 11].

Finally, consider the problem of finding non-saturating valid cycles. From Corollary 7.4 it follows that there exists a non-saturating valid cycle only if there also exists a saturating feasible cycle family. Hence, the problem of finding a non-saturating valid cycle is not really relevant in practice. For completeness, we refer to our algorithm in [73] for the solution of this problem.

### 7.2 A Characterization of the Simplicial Structure

Infeasible routing patterns arise from solutions to model (6.7), i.e. the Farkas system of (6.6), on page 71. In this section, we illustrate the connection between graph embeddings
and the class of simplicial cycle families, i.e. the combinatorial structure that induces the large class of solutions to (6.7) referred to as simplicial solutions in Chapter 6.

Recall that a simplicial cycle family can be represented as

$$C = \{ C^i \mid C^i = F^i \cup B^i, B^i \subseteq A^i \}.$$  \hspace{1cm} (7.11)

The class of simplicial cycle families is indeed comprehensible as already demonstrated in Section 7.1 for the subclass induced by valid cycles. Next, we show how an embedding of a graph can be used to form several simplicial cycle families.

### 7.2.1 Generating Simplicial Cycle Families

We consider the problem of constructing (an instance with) a simplicial cycle family. This is not directly useful in solving an ISPR problem. However, we believe that a formalized method of constructing examples, as opposed to an ad hoc procedure, is actually an important task. Indeed, this may be a very helpful tool to provide intuition, get important insights and to develop a deeper understanding of the problem at hand, or its solutions.

In our case, we get sufficiently acquainted with a very large class of infeasible ISPR structures. In the end, this translates to the concept of dependency graphs and a procedure to characterize irreducible generators. It is also possible to use this knowledge to develop combinatorial graph search algorithms for subclasses of simplicial cycle families.

To construct a simplicial solution with two destinations, i.e. a valid cycle it follows straightforward from Section 7.1 that it suffices to take an undirected cycle and assign forward and backward labels to its edges. If the cycle is simple, the valid cycle is irreducible. If all forward (and hence, backward) arcs are connected, i.e. belong to the same path segment, then the valid cycle corresponds to a subpath inconsistency conflict.

It is more complicated to construct a simplicial cycle family that involves more than two destinations. To deal solve this problem, we observe that there is a connection to oriented cycle double covers (CDC) and cellular graph embeddings. For a treatment of these subjects, see e.g. [133, 144, 145, 208, 215].

An embedding of an undirected graph $G$ on an (oriented) surface is a drawing of $G$ on the surface without edge crossings. The embedding is cellular if every face boundary is a cycle of the graph. An oriented CDC is a collection of directed cycles, such that their union covers each edge in $G$ exactly twice, once in each direction. A major conjecture in topological graph theory is that all (bridgeless) graphs has an oriented CDC, or similarly, a cellular embedding in an orientable surface, see e.g. [133, 144, 145, 208, 215].

An oriented CDC of an undirected graph gives a simplicial solution by assigning arcs to adjacent cycles as follows. First orient all edges to make them arcs and associate each directed cycle with a destination. Then, traverse each cycle in its direction and make all arcs used backward SP-arcs for the destination of the cycle being traversed. We can also construct a simplicial solution from a cellular embedding of a bridgeless undirected graph on an oriented surface since the cells along with the orientation gives an oriented CDC.

The procedure is demonstrated for two graphs in Examples 7.1 and 7.2. First, we use a planar graph, i.e. a graph embedded into the sphere. Then, a more complex example is considered where the Petersen graph is embedded into a torus.
Remark 7.5. We focus on saturating simplicial cycle families. The construction of a non-saturating cycle family requires that some directed cycle has no backward arcs. These "slack cycles" can then be discarded to obtain a non-saturating cycle family. Indeed, the total flow in these cycles in model (6.7) will be strictly negative and we may interpret the slack in the capacity constraints as a flow in these cycles. See further Remark 7.7 and the proof of Theorem 8.2.

Example 7.1

Figure 7.3: (Left) A planar embedding of a graph. (Right) The oriented CDC induced by the embedding. Each face in the embedding yields a directed cycle since the surface (the sphere) is orientable. A simplicial family of cycles is obtained from the arc orientations on the right.

The planar drawing of the graph on the left in Figure 7.3 is also called a cellular embedding into the sphere. An oriented CDC is obtained by the 2-cells/2-complexes/faces in the cellular embedding and an orientation of the surface. If the orientations of the inner faces are chosen to be counter-clockwise the directed cycles on the right in Figure 7.3, also given below in (7.12), is as an oriented CDC.

\[
C_1 = (1 \ 4 \ 5 \ 2), \quad C_3 = (4 \ 7 \ 8 \ 5), \quad C_4 = (5 \ 8 \ 9 \ 6), \quad C_0 = (1 \ 2 \ 3 \ 6 \ 9 \ 8 \ 7 \ 4). \quad (7.12)
\]

Note that the outer face cycle, \(C_0\), is the reversal of the sum of all inner cycles, i.e.

\[
-C_0 = C_1 + C_2 + C_3 + C_4. \quad (7.13)
\]

The collection of simple cycles in (7.12) is turned into a simplicial cycle family via the above procedure. First, (arbitrary) orient each edge in the graph. Here, we point all arcs to the south-west. Take an inner directed cycle. To comply with the orientation, its upper horizontal arcs must be forward arcs and the lower horizontal arcs must be backward arcs. Similarly, its left vertical arcs are forward arcs and the right vertical arcs are backward arcs. The outer cycle is handled in a similar manner. This yields a simplicial cycle family consisting of the five cycles,
The Simplicial Structure

\[ C_a = \{(1, 4), (2, 1)\} \cup \{(2, 5), (5, 4)\}, \quad C_b = \{(4, 7), (5, 4)\} \cup \{(5, 8), (8, 7)\}, \]
\[ C_c = \{(2, 5), (3, 2)\} \cup \{(3, 6), (6, 5)\}, \quad C_d = \{(5, 8), (6, 5)\} \cup \{(6, 9), (9, 8)\}, \]
\[ C_e = \{(3, 6), (6, 9), (8, 7), (9, 8)\} \cup \{(1, 4), (2, 1), (3, 2), (4, 7)\}, \]

where the left and right arc sets in the unions are forward and backward arcs, respectively. We do not consider the actual destinations associated with the cycles, cf. Remark 6.10.

Remark 7.6. It is often desirable that a constructed conflict is irreducible, since it can otherwise be replaced with a smaller conflict. The orientation used in Example 7.1 was chosen a bit carelessly and the cycle family is actually reducible. We addressed this issue further below, see Section 7.3 and 7.4, and in particular Remark 7.12.

Remark 7.7. A non-saturating simplicial cycle family can be obtained by choosing a different orientation where some cycle(s) are not associated with a backward arc. Consider for example the simplicial cycle family in (7.14) in Example 7.1. If the edges between \((2, 5)\) and \((4, 5)\) are reversed, a cycle family with

\[ C_a = \{(1, 4), (4, 5), (5, 2), (2, 1)\} \cup \emptyset, \]
\[ C_b = \{(3, 2)\} \cup \{(3, 6), (5, 2), (6, 5)\}, \]
\[ C_c = \{(4, 7)\} \cup \{(4, 5), (5, 8), (8, 7)\}, \]

and cycles \(C^d\) and \(C^e\) as above is obtained. The cycle \(C^a\) only has forward arcs so we can treat it as a slack cycle to get a (reducible) non-saturating simplicial cycle family.

Example 7.2

[Diagram of the Petersen graph]

**Figure 7.4:** A drawing of the famous Petersen graph in the plane with crossings.

Consider the Petersen graph in Figure 7.4. To obtain an oriented CDC it can be embedded into an orientable surface. It is well known that the Petersen graph is toroidal, i.e. it can be embedded into the torus, for instance as in Figure 7.5.
7.2 A Characterization of the Simplicial Structure

A toroidal embedding of the Petersen graph. To get a torus, stretch and fold the outer polygon by matching the arrows with the same labels (A and B) and the graphs edges along two arrows with the same label sequentially.

If we use this embedding and choose the orientation to be counter-clockwise, then the faces in the embedding in Figure 7.5 yields the following directed cycles,

$$
C_1 = (1 \, 6 \, 9 \, 7 \, 2), \quad C_2 = (2 \, 7 \, 10 \, 8 \, 3), \quad C_3 = (3 \, 8 \, 6 \, 1 \, 5 \, 4), \\
C_4 = (4 \, 5 \, 10 \, 7 \, 9), \quad C_5 = (5 \, 1 \, 2 \, 3 \, 4 \, 9 \, 6 \, 8 \, 10).
$$

The directed cycles in (7.16) gives a simplicial cycle family when the edges are orientated. An orientation can also be determined implicitly by assigning each edge to comply with the direction of the directed cycle associated with one of its two adjacent faces.

Assign each horizontal edge to the directed cycle below it and each non-horizontal edge to the directed cycle above it. To comply with all directed cycles, edges assigned to a given directed cycle will be used backwards for that particular cycle and edges not assigned to it will be used forwards. This yields the assignment and orientation in Figure 7.6 and the simplicial cycle family consisting of the following five cycles

$$
C^1 = \{(1, 6), (6, 9), (9, 7)\} \cup \{(2, 7), (1, 2)\}, \\
C^2 = \{(2, 7), (7, 10), (10, 8)\} \cup \{(3, 8), (2, 3)\}, \\
C^3 = \{(3, 8), (8, 6)\} \cup \{(1, 6), (5, 1), (4, 5), (3, 4)\}, \\
C^4 = \{(4, 5), (9, 4)\} \cup \{(10, 5), (7, 10), (9, 7)\}, \\
C^5 = \{(5, 1), (1, 2), (2, 3), (3, 4), (10, 5)\} \cup \{(9, 4), (6, 9), (8, 6), (10, 8)\}.
$$

where again the first arc set in the union corresponds to the set of forward arcs.

We have illustrated how cellular embeddings can be used to construct simplicial solutions. Any simplicial cycle family constructed in this manner is feasible. It is in fact a generator if the graph is connected. However, it is not guaranteed that it is irreducible, this depends on the orientation as pointed out in Remark 7.6. For the above examples it can be verified that the cycle family in Example 7.1 is reducible while the cycle family in Example 7.2 is irreducible.

After we introduce the dependency graph and Theorem 7.3, it becomes straightforward to decide whether a cycle family is reducible or not. We also obtain a post-processing measure to turn a reducible cycle family into an irreducible cycle family.
The Simplicial Structure

7.3 Dependency Graphs

Given a cycle family, we define a dependency graph to represent how its cycles are depending on each other. There is a close resemblance between our dependency graph and the intersection graph for CDCs and the dual graph.

Recall the definition of the dual graph. Given an embedding, \( I \), of an undirected graph, \( G = (V, E) \), the dual graph, \( G_I = (V^*, E^*) \), is determined as follows. For each face in the embedding of \( G \), there is a vertex in \( G_I \) and two vertices in \( G_I \) are connected if and only if their corresponding faces are adjacent in the embedding \( I \). This is well known for planar graphs, but perhaps not for non-planar graphs. Therefore, as an example, we give the dual of the Petersen graph in Figure 7.7 for the embedding in Figure 7.5.

**Figure 7.6:** The embedding of the Petersen graph on the torus where all arcs have been assigned to an adjacent face.

**Figure 7.7:** The dual graph for the embedding of the Petersen graph in Figure 7.5.
Remark 7.8. Some remarks on the dual graph are given here. (1) the dual graph may contain parallel arcs, (2) a graph can have several non-isomorphic dual graphs depending on the embedding, and (3) there is an embedding such that $(G_I^*)^* = G$.

When a cycle family has an embedding, the dual graph encodes information on the adjacency between its cycles which corresponds to a dependency relation between the cycles. An inadequacy with the dual graph, for our purposes, is that it does not tell which arcs are used backwards by which cycles. This information is crucial since it specifies exactly how cycles depend on each other, and not just that there is a dependency. This issue is resolved by using the dependency graph induced by a cycle family.

Given a simplicial cycle family $C$, there may be some cycle $C^i = F^i \cup B^i \in C$ that is not simple. The faces in an embedding correspond to simple cycles. Therefore, we let $\tilde{C}$ be a decomposition of $C$ into simple cycles, i.e.

$$\tilde{C} = \{ C^i_k \mid C^i_k = F^i_k \cup B^i_k, \ k = 1, \ldots, K^i \}, \text{ and } C^i = \bigcup_{k=1}^{K^i} C^i_k. \quad (7.18)$$

Definition 7.2

Let $C$ be a simplicial cycle family decomposed into a collection of simple directed cycles, $\tilde{C}$. The dependency graph, $\tilde{G}^*(\tilde{C}) = (N^*, \tilde{A}^*)$, associated with $\tilde{C}$ is the directed multi-graph with node set $N^* = \tilde{C}$, and arc set,

$$\tilde{A}^* = \{ (C_i, C_j) \in \tilde{C} \times \tilde{C} \mid \text{there exists } (u, v) \in C_i \cap F_i \text{ and } (v, u) \in C_j \cap B_j \}. \quad (7.19)$$

In words, a dependency graph, $\tilde{G}^*(\tilde{C})$, obtained from a simple cycle decomposition is the directed multi-graph with a node for each simple directed cycle in $\tilde{C}$ and a directed arc $(i, j) \in \tilde{A}^*$ for each arc in $\tilde{C}$ where the simple cycle corresponding to node $i \in N^*$ uses an arc in the simple cycle corresponding to node $j \in N^*$ in the forward direction. Hence, the existence of arc $(i, j)$ means that cycle $i$ depends on cycle $j$, so to speak.

If the simplicial cycle family $C$ comes with an embedding, $I$, in the underlying graph, then the dependency graph obtained via $I$ is called an embedded dependency graph and it is denoted by $G_I^*(\tilde{C})$.

Given an embedded dependency graph, $G_I^*(\tilde{C})$, it sometimes suffices to know that there is some arc from $i$ to $j$ in $\tilde{A}^*$, rather than which, i.e. these multi-arcs can be replaced by a single arc. This results in a simple graph that we refer to as the simple dependency graph and denote by $G_I^s(\tilde{C}) = (N^*, A^*)$.

The definition of embedded and simple dependency graphs are very closely connected to the dual graph. The embedded dependency graph is an orientation of the dual graph in accordance with the dependency between cycles. The simple dependency graph is a projection of the embedded dependency graph onto the underlying simple graph.

Remark 7.9. Which dependency graph to use depends on how it is to be used. The embedded dependency graph is a multigraph that encodes all necessary information. This is useful if we want to recover the simplicial cycle family that induce the dependency graph; See below. The simple dependency graph projects out some information, but captures the essential, structural information. This suffices in many cases, e.g. to characterize irreducible generators in Section 7.4.
Example 7.3

Figure 7.8: (Left) The embedded dependency graph obtained from the cycle family in Example 7.1. (Right) The corresponding simple dependency graph. Observe that some arcs are used in both directions.

Since $\tilde{G}^\gamma(\tilde{C})$ comes with an embedding, it is easy to construct the embedded dependency graph from the associated dual graph; it suffices to direct its edges to comply with the structural dependency (observe that an edge can induce two anti-parallel arcs). The simple dependency graph is then obtained by removing multiple edges. The planar embedding in Figure 7.3 in Example 7.1 yields the dependency graphs in Figure 7.8. Observe that the arcs from $C_1$ to $C_0$ and from $C_0$ to $C_4$ on the right in Figure 7.8 correspond to two edges each on the left in Figure 7.8.

We give another example. The embedding of the Petersen graph in Example 7.2 yields the dual graph in Figure 7.7. Using the orientations in Figure 7.6, we orient all edges to obtain the embedded dependency graph in Figure 7.9. In this case, the embedded dependency graph is simple and hence, it coincides with the simple dependency graph.

Figure 7.9: The dependency graph induced by the cycle family in Example 7.2.
Next, we illustrate the "duality" of a simplicial cycle family and its dependency graph. As indicated in Remark 7.9, the embedded dependency graph associated with a simplicial cycle family can be used to (almost) recover the cycle family. Alternatively, we can construct a simplicial cycle family with a desired dependency graph. This can be useful for instance if we seek an example with some particular property that can easily be expressed in terms of the dependency graph.

Let $\tilde{G}_I$ be a directed multi-graph embedded into an orientable surface. We describe how to construct a simplicial cycle family with $\tilde{G}_I$ as its embedded dependency graph. It suffices to form the dual graph of $\tilde{G}_I$ and then label all arcs to be consistent with the dependency graph $\tilde{G}_I$. The labelling can be carried out as follows. For each node $i$ in $\tilde{G}_I$, traverse the corresponding face in the dual graph clockwise. When an edge $(u, v)$ on the face boundary intersects an arc $(i, j)$ in $\tilde{G}_I$, assign $(u, v)$ to $F_i$ if it leaves the face and assign $(v, u)$ to $B_j$ if it enters the face. An example is used to illustrate the procedure.

**Example 7.4**

![Diagram](image)

**Figure 7.10:** (Left) An embedding of a digraph. (Right) The cycle family induced by this digraph. Note that the digraph is the dependency graph induced by the cycle family.

Consider the graph on the left in Figure 7.10 that acts as the desired dependency graph. When we apply the procedure above, we obtain the simplicial cycle family on the right in Figure 7.10 where the dependency graph has also been sketch. First note that the graphs are indeed dual to each other. Then, it is straightforward to check that the labellings are correct, i.e. that an arc in the cycle family is used backwards if the intersecting arc in the dependency graph enters the node associated with the cycle. It is also easy to verify that the graph on the left is indeed the dependency graph for the cycle family as embedded on the right in Figure 7.10.

Some remarks are in place.

**Remark 7.10.** When the embedded dependency graph is constructed, the information about the destinations is lost. Hence, applying the above procedure to brings back a cycle family that is structurally equivalent, but not completely recovered. This is why we used "duality" above.
The Simplicial Structure

Remark 7.1. The simple dependency graph captures the structure of a simplicial cycle family, but loses more information than the embedded dependency graph. Therefore, it cannot be used to recover the arcs in a simplicial cycle family but rather path segments which again captures the essential structure of the original simplicial cycle family.

Remark 7.12. Using the characterization of irreducibility in the next section it is straightforward to make cycle families correspond to irreducible conflicts by introducing some additional arcs in the embedded dependency graph and then (re-)apply the technique above. This results in sub-dividing an arc in the original cycle family. Actually, this procedure was carried out when Example 6.2 was continued. In Figure 6.4 on page 82 the arc (2, 5) was sub-divided to obtain the irreducible cycle family in Figure 6.5 on page 83. See also Examples 6.4 and 6.7 in [73]

The most important property of the dependency graph is that it can be used to characterize generators and irreducible simplicial solutions.

7.4 Characterization of Irreducible Generators

In this section we will assume that the lineality space contains only the origin which implies that generators are extreme rays.

Using algebraic methods, it is straightforward to decide if a solution, \( \theta \in \Theta \), to (6.7) corresponds to an extreme ray and an irreducible extreme ray, respectively. Indeed, it suffices to check the rank of the columns associated with non-zero variables to decide if a solution is extremal. To decide irreducibility, take the rows in (6.6) indexed by the negative support of \( \theta \). If there is a strict subset of these rows that forms an infeasible system of inequalities together with all inequalities in (6.6), then \( \theta \) is reducible. To decide this, it suffices to solve an LP for each subset where exactly one row in (6.6) corresponding to a negative entry in \( \theta \) is removed.

We give a purely graph theoretical characterization of (irreducible) extreme rays.

Theorem 7.2
Assume that \( \Theta^0 = \{0\} \) and consider a simplicial solution \( \theta \in \Theta \) associated with the cycle decomposition \( \hat{C} \) and the simple dependency graph \( G^*(\hat{C}) \). Then, \( \theta \) corresponds to an extreme ray if and only if \( G^*(\hat{C}) \) is strongly connected.

Proof: The solution \( \theta \) corresponds to an extreme ray if and only if there exists no ordering of the directed cycles in \( \hat{C} \) such that a strict sub-sequence induces a solution \( \theta_1 \).

Consider an arbitrary ordering of the cycles and the following procedure. Remove one cycle at a time. If all removed cycles together induce a solution \( \theta_1 \), then stop. We claim that the cycles removed can have no dependencies with the remaining cycles. Indeed, \( \theta, \theta_1 \) and thus \( \theta - \theta_1 \) only have components in \{0, ±1\}. Together with the capacity constraint that forces the total flow on an arc to be non-positive, this implies that no cycle in \( \theta - \theta_1 \) can depend on a cycle in \( \theta_1 \), or vice versa. In terms of the simple dependency graph, \( G^*(\hat{C}) \), where cycles correspond to nodes, this means that if the procedure terminates with a non-empty graph, it has effectively divided \( G^*(\hat{C}) \) into two components.

Since the procedure terminates with the empty graph for all orderings if and only if \( \theta \) corresponds to an extreme ray and there is no division of \( G^*(\hat{C}) \) into two components if and only if \( G^*(\hat{C}) \) is strongly connected, the result follows.
Remark 7.13. Consider a solution \( \theta \in \Theta \) in the case where the lineality space is not the origin. If \( \theta = \theta_1 + \theta_2 \) for some \( \theta_2 \in \Theta^0 \setminus \{0\} \), the procedure in the proof of Theorem 7.2 will find \( \theta_1 \) and \( \theta_2 \) as components of \( G^*(\hat{C}) \). Hence, to characterize generators when \( \Theta^0 \neq \{0\} \) we can find all SCCs of \( G^*(\hat{C}) \) and remove all SCCs that induce a solution in \( \Theta^0 \); if there is only one SCC left, then the solution \( \theta \) corresponds to a generator.

Since the negative support of infeasible solutions yield valid inequalities and we are interested in non-dominated inequalities, it follows that only irreducible extremal solutions matter. We give a characterization of irreducible extremal solutions. Informally, a cycle family is irreducible if no set of nodes in the dependency graph associated with a single destination forms a cut set.

Theorem 7.3
Assume that \( \Theta^0 = \{0\} \) and consider a simplicial solution \( \theta \in \Theta \) associated with the cycle decomposition \( \hat{C} \) and the simple dependency graph \( G^*(\hat{C}) \). For \( l \in L \), let \( D^l \) be the subgraph of \( G^*(\hat{C}) \) obtained by removing all cycles associated with destination \( l \). Then, \( \theta \) corresponds to an irreducible extreme ray if and only if \( D^l \) is strongly connected for all \( l \in L \).

Proof: Take \( l' \in L \) arbitrary and denote the cycles associated with \( l' \) by \( C \cup C_{l'} \) and the dependency graph resulting by removing these cycles by \( G(\hat{C}_{l'}) \).

We claim that \( \theta \) corresponds to an irreducible extreme ray if and only if there exists no strict subset of cycles \( C \cup C_{l'} \) that induces a total flow that is non-positive on all arcs except for arcs that can be used backwards for \( l' \). Indeed, let \( C \subset \hat{C} \setminus C_{l'} \) be such a subset that induces the flow \( \tilde{\theta}_a^l \) and the total flow \( \tilde{\theta}_a \), i.e.

\[
\tilde{\theta}_a^l := \begin{cases} 1, & a \in F^l \subseteq C^l \in C, \\ -1, & a \in B^l \subseteq C^l \in C, \\ 0, & \text{otherwise,} \end{cases} \quad \text{and} \quad \tilde{\theta}_a := \sum_{C^l \in C} \tilde{\theta}_a^l \begin{cases} \leq 0, & a \in A \setminus B, \\ \leq 1, & a \in B', \\ \end{cases}
\]

Then, construct \( \theta \in \Theta \) as follows,

\[
\tilde{\theta}_a^l := \begin{cases} \tilde{\theta}_a^l, & \text{if } l \neq l', \\ -\tilde{\theta}_a, & \text{if } l = l'. \end{cases}
\]

The solution \( \theta \) is feasible since the total flow is 0 on all arcs and negative only on arcs in \( B' \) for \( l \in L \). In particular, for \( l' \) we have \( \tilde{\theta}_a^{l'} = -\tilde{\theta}_a \) which, via (7.20), is negative only if \( a \in B' \).

We claim that there is a strict subset \( C \subset \hat{C} \setminus C_{l'} \) with this property if and only if \( G^*(\hat{C}_{l'}) \) is strongly connected. Indeed, this follows by an argument almost identical to the proof of Theorem 7.2.

Remark 7.14. Observe that Theorems 7.2 and 7.3 do not rely on an embedding of the simple dependency graph, only its combinatorial structure.

We use some examples to illustrate the latter theorem.
Example 7.5

Recall the cycle family constructed in Example 7.1. It was pointed out in Remark 7.6 that this cycle family is not irreducible. Using Theorem 7.3 this can easily be seen in the associated dependency graph in Figure 7.8 on page 104. If node $C_0$ is removed, the node $C_1$ becomes an SCC in the reduced graph. Now, $C_1$ and the backward arcs in $C_0$ intersecting $C_1$ yields a solution induced by the valid cycle

$$C = \{(1, 4), (2, 1)\} \cup \{(2, 5), (5, 4)\}. \quad (7.22)$$

Hence, the original cycle family is reducible as previously claimed.

On the other hand, the cycle family from the Petersen graph in Example 7.1 is irreducible. We verify this using Theorem 7.3. If a node is removed from the associated dependency graph in Figure 7.9 on page 104, the remaining graph is strongly connected.

Example 7.6

Consider the cycle family induced by the orientation and embedding of the graph in Figure 7.11. There are four destination indices involved in this conflict; cycle 1 and 3 use solid arcs as backward arcs, cycle 2 and 5 use arcs that are mixed dotted and solid, cycle 4 use dotted arcs and cycle 0 use dashed arcs. This cycle family is reducible. It is possible to find six conflicts among these arcs. For instance the two leftmost cycles can also be used to form valid cycles. Let us investigate this closer. Consider the dependency graph associated with this cycle family, it is given in Figure 7.12.

First, consider the removal of the solid nodes associated with cycle 1 and 3. This leaves cycle 2 as an SCC. If the backward arcs from cycle 1 and 3 are combined with the backward arcs in cycle 2, a valid cycle with arcs spanned by cycle 2 is obtained.

If we remove the node associated with cycle 0 this leaves cycle 1 as an SCC. If the backward arcs from cycle 0 are combined with the single backward arc in cycle 1 the valid
cycle with arcs spanned by cycle 1 is obtained. If instead the SCC from cycles 3 and 4 is considered, another conflict is obtained. The backward arcs in cycle 3 and 4 combined with some arcs in cycle 0 yields an irreducible solution.

**Figure 7.12:** The simple dependency graph associated with the cycle family in Figure 7.11.

Theorem 7.2 and 7.3 yield good characterizations of extremal and irreducible solutions, or equivalently, cycle families or routing conflicts. By this, we mean that our characterizations are given in purely graph theoretical terms that can be decided efficiently. In particular, the characterization of irreducible cycle families has important theoretical and practical consequences. An example of the former is given in Chapter 10 where we use it to characterize non-dominated valid inequalities from routing conflicts.

From a practical perspective it is important to find irreducible routing conflicts since a valid inequality associated with a reducible routing conflict is trivially dominated. Routing conflicts of small cardinality are preferred, but NP-hard to find, and a greedy algorithm is commonly used to reduce the size of a routing conflict, see Section 5.4 in [46, 49]. The greedy algorithm is based on repeatedly solving ISPR problems to decide if some part of the routing conflict can be removed, see further the second paragraph of this section about algebraic methods for deciding if solutions are extremal and irreducible.

Theorem 7.3 can be used to speed up the greedy algorithm substantially. Indeed, instead of solving a series of quite costly LPs, it suffices to construct the graph $D^l$ in Theorem 7.3 for all $l \in L$ and determine if it is strongly connected. This can be done in $O(|L|(n + m))$ time.
Cycle Basis Formulations

The Farkas systems of the inverse shortest path routing (ISPR) problems considered earlier in this thesis are very similar to linear multicommodity circulation problems. In this chapter we exploit this and propose a novel reformulation technique for the Farkas system of the partial realizability problem via fundamental cycle bases.

In a manner similar to a Dantzig–Wolfe reformulation, a cycle basis reformulation is based on enumerating variables corresponding to cycles. A major difference is that a polynomial number of variables suffices when cycle bases are used whereas the Dantzig–Wolfe reformulation in general requires an exponential number of variables. Both approaches, by construction, imply that flow conservation constraints are satisfied since the circulation aspect of the problem is in the variable encoding. Therefore, it suffices to handle capacity constraints, and possibly, commodity specific flow bounds. This implies that there are fewer constraints. However, the reformulation of the commodity variable bound constraints typically yields dense bound constraints and the potential advantage of reformulating the problem essentially disappears. In the Dantzig–Wolfe approach, the latter difficulty can be handled in the pricing problem which results in a master that only contains aggregated capacity constraints.

The good news about the partial realizability problem is that there exists a very natural choice of cycle bases that allows us to handle all commodity specific flow bounds by non-negativity constraints. Hence, the disadvantage implied by these flow bounds in a cycle basis reformulating approach disappears.

Our cycle basis approach yields a compact model for partial realizability equivalent to (6.7) in Chapter 6. As the Dantzig–Wolfe reformulation it only has aggregated capacity constraints. In contrast to the Dantzig–Wolfe reformulation, it only has a polynomial number of variables. Therefore, the resulting cycle basis model has fewer variables and fewer constraints.

By emphasizing the circulation structure of the problem, we get some important practical and theoretical insights via the cycle basis model that translates to the partial realizability models, (6.6) and (6.7). In particular, we show that under very general conditions,
a subset of constraints are (in a sense) redundant. From a practical perspective, this is very important since the Farkas system of an ISPR problem is typically quite hard to solve even though it is an LP. Indeed, all capacity constraints are binding which results in massive degeneracy. Removing the redundant constraints gives an even smaller model which suffers less from degeneracy and is often easier to solve. Our preliminary computational experiments shows that the compact cycle basis model can be solved more efficiently.

Finally, we will also consider the Farkas system of our cycle basis model. This yields a path based formulation equivalent to the partial realizability variant of ISPR. The interesting thing is that our path formulation only has a polynomial number of paths in contrast to all earlier path formulations for all inverse shortest path problems in the literature. Hence, there is no need to use a separation problem to find violated path inequalities. In a sense, we have a priori identified a polynomial subset of path constraints that suffices.

Outline We introduce cycle bases and illustrate how they can be used in modelling in Section 8.1. Then, this technique is applied to the partial realizability problem in Section 8.2. In Section 8.3, we derive some properties of partial realizability problem and the associated cycle basis formulation. Finally, in Section 8.4, the Farkas system of the cycle basis model is considered.

8.1 Modelling with Cycle Bases

Several network flow problems can be modelled via circulations. Using cycle bases and the induced relation between flow in cycles and flow on arcs, it is straightforward to develop mathematical models for these network flow problems. Some necessary definitions about the cycle space and fundamental cycle bases are introduced here. For additional results, we refer to e.g. [161], the recent survey article [151] or the text books [42, 55].

8.1.1 Oriented Circuits, Circulations and Cycle Bases

Let $G = (N, A)$ be a strongly connected digraph with incidence matrix $M_G$. An oriented circuit, $C = F \cup B \subseteq A$, is a set of forward arcs, $F$, and a set of backward arcs, $B$, such that the arcs in $F$ and the reversal of the arcs in $B$ induce a simple directed cycle. The incidence vector, $\gamma^C \in \{-1, 0, 1\}^A$, of the oriented circuit $C = F \cup B$ is defined as

$$
\gamma^C_a := \begin{cases} 
1, & a \in F, \\
-1, & a \in B, \\
0, & a \not\in C.
\end{cases}
$$

The cycle space $C_G \subset \mathbb{R}^A$ of $G$ is the vector space generated by the incidence vectors of all oriented circuits of $G$. A circulation is a point in the cycle space. The cycle space can also be defined as the null space of $M_G$. Since, rank $M_G = n - 1$, the dimension of $C_G$, or the cyclomatic number of $G$, is $\dim C_G = m - n + 1$. Therefore, a cycle basis is a set of $m - n + 1$ oriented circuits whose incidence vectors form a basis of $C_G$ and the associated cycle matrix, $\Gamma$, is the matrix formed by these incidence vectors.

There are several classes of cycle bases. For our purposes it suffices to consider fundamental cycle bases. As early as 1847, Kirchhoff presented a very elegant method to
construct a basis for the cycle space in \([152]\), see also \([43]\). The simple idea is to use a spanning tree and the oriented circuits induced by the arcs not in the tree.

### 8.1.2 Fundamental Cycle Bases

Let \(T\) be the arcs in a spanning tree in \(G\). Given an arc outside the tree, \(\bar{a} \in A \setminus T\), the fundamental cycle, \(C^T_{\bar{a}}\), is the unique cycle in \(G\) induced by \(T \cup \{\bar{a}\}\). The orientation of \(C^T_{\bar{a}}\) is determined by the single arc, \(\bar{a}\), not in the tree. The set of all fundamental cycles w.r.t. \(T\) is denoted by \(C(T) := \{C^T_{\bar{a}} : \bar{a} \in A \setminus T\}\) and is called a fundamental cycle basis of \(G\). The cycle matrix induced by the incidence vectors in \(C(T)\) is denoted by \(\Gamma^T\).

The column, \(\gamma_{\bar{a}}\), associated with \(C^T_{\bar{a}}\) in \(\Gamma^T\) is completely determined by the non-tree arc, \(\bar{a}\). Further, the only possible non-zero entries of the column are the 1 in the row associated with \(\bar{a}\) and the entries in the rows associated with tree arcs. Hence, the cycle matrix \(\Gamma^T \in \mathbb{Z}^{A \times A \setminus T}\) can be partitioned as

\[
\Gamma^T = \begin{pmatrix}
I \\
\hat{\Gamma}^T
\end{pmatrix},
\]

where the identity matrix is of order \(m - n + 1\) and \(\hat{\Gamma}^T \in \mathbb{Z}^{T \times A \setminus T}\). The rows that form the identity correspond to the arcs outside the tree and the rows that form \(\hat{\Gamma}^T\) correspond to the arcs in the tree. Clearly, \(\hat{\Gamma}^T\) forms a basis of the cycle space.

**Remark 8.1.** A word on notation. Throughout this chapter we use \(\bar{a} := (s, t) \in A \setminus T\) to denote arcs that corresponds to columns and \(a := (i, j) \in A\) to denote arcs that corresponds to rows, i.e. an arc \(\bar{a}\) induce a fundamental cycle \(C^T_{\bar{a}}\) associated with the column \(\gamma_{\bar{a}}\) and \(a\) gives the entry \(\gamma^a_a\) in \(\gamma_{\bar{a}}\) in the row associated with arc \(a\).

By assumption, \(G\) is strongly connected and hence, \(\text{rank} \ M_G = n - 1\). Therefore, we may remove a single row from \(M_G\) and partition the remaining matrix as \((M_T, M_N)\). The sub-matrices \(M_T\) and \(M_N\) contain the columns corresponding to arcs in the tree and not in the tree, respectively. This partitioning implies that \(M_T\) is invertible and also that

\[
\hat{\Gamma}^T = -M_T^{-1}M_N.
\]

**Example 8.1**

![Figure 8.1: A graph and a set of vectors in its cycle space. The columns \(C_{13}, C_{21}\) and \(C_{35}\) induce a fundamental cycle basis. The bold arcs corresponds to the tree.](image)
The graph in Figure 8.1 has 5 nodes and 7 arcs $\dim C_G = 7 - 5 + 1 = 3$. The incidence vectors, $C_0, C_{13}$ and $C_{21}$ in the table on the right in Figure 8.1 are linearly independent and form a basis for the cycle space. However, it is not fundamental since the induced cycle matrix does not contain an identity matrix of order three as a sub-matrix, see the partitioning in (8.2).

To obtain a fundamental cycle basis we (arbitrarily) choose the tree marked with thick arcs on the left in Figure 8.1. This induces the three linearly independent incidence vectors $C_{13}, C_{21}$ and $C_{35}$ in the table in Figure 8.1. Note that the rows associated with the non-tree arcs (1, 3), (2, 1) and (3, 5) form an identity matrix of order 3.

8.1.3 Modelling Circulations with Cycle Bases

We first illustrated how cycle bases can be used to model circulations. Denote the amount of flow sent in the fundamental cycle $C_a$ by $x_a$ and let $\theta \in C_G$ be the circulation induced by $x \in \mathbb{R}^{A \setminus T}$. Since $\Gamma^T$ has full column rank, see (8.2), we have the following bijective relation between $x$ and $\theta$,

$$\theta_a = \sum_{a \in A \setminus T} \gamma_a^T x_a, \quad a \in A,$$

or equivalently, using the cycle matrix,

$$\theta = \Gamma^T x. \quad (8.5)$$

Also observe that the identity matrix in the partitioning in (8.2) implies that

$$\theta_a = x_a, \quad a \in A \setminus T. \quad (8.6)$$

Given the relation between the flow in cycles and flow on arcs it is straightforward to develop mathematical models for several common network flow problems. Below, the well known the minimum cost (circulating) flow and multicommodity minimum cost (circulating) flow problems are considered.

8.1.4 The Minimum Cost Circulating Flow Problem

The minimum cost flow problem is central in combinatorial optimization. It is no restriction to consider the circulation version of the problem. Indeed, sources and sinks can be handled by adding a super source and a super sink and a set of additional arcs. The possibility to bound the flow on these new arcs also handles interval sources and sinks.

A problem formulation is as follows. Given a strongly connected graph $(G)$, arc costs $(c_a)$, lower $(l_a)$ and upper $(u_a)$ bounds, find a minimum cost circulation that satisfies the flow bounds. Let $\theta_a$ be the amount of flow on arc $a \in A$, then a common mathematical model is as follows,
minimize $\sum_{a \in A} c_a \theta_a$

subject to

$$\sum_{a \in \delta^+(i)} \theta_a - \sum_{a \in \delta^-(i)} \theta_a = 0, \quad i \in N,$$

$$l_a \leq \theta_a \leq u_a, \quad a \in A.$$  \hfill (8.7a)

$$l_a \leq \theta_a \leq u_a, \quad a \in A.$$  \hfill (8.7b)

We use (8.5), i.e. $\theta = \Gamma^T x$ to convert this model into a cycle basis formulation. Let $T$ be a spanning tree in $G$. For each $\bar{a} \in A \setminus T$, the variable $a_\bar{a}$ measures the flow in the fundamental cycle induced by $\bar{a}$. By construction, all circulation constraints are satisfied. The cost of sending one unit of flow along the fundamental cycle $C^T_{\bar{a}}$ is

$$\sum_{a \in A} \gamma_{\bar{a}}^a c_a.$$  \hfill (8.8)

To complete the model it suffices to model the flow bounds. This yields a cycle basis formulation of the minimum cost circulating flow problem in matrix form,

minimize $c^T \Gamma^T x$

subject to $l \leq \Gamma^T x \leq u.$  \hfill (8.9a)

It is insightful to consider an alternative derivation of (8.9) from (8.7). We partition the $\theta$-variables into tree (basic) and non-tree (non-basic) variables, $\theta_B$ and $\theta_N$. Then, write (8.9) in matrix form, i.e. as

minimize $c_B^T \theta_B + c_N^T \theta_N$

subject to

$$M_B \theta_B + M_N \theta_N = 0,$$

$$l_B \leq \theta_B \leq u_B,$$  \hfill (8.10a)

$$l_N \leq \theta_N \leq u_N.$$  \hfill (8.10b)

The node balance constraint (8.10a) and the relation between the incidence and the cycle matrix in (8.3) yield,

$$0 = (M_B, M_N) \begin{pmatrix} \theta_B \\ \theta_N \end{pmatrix} = (I, M_B^{-1} M_N) \begin{pmatrix} \theta_B \\ \theta_N \end{pmatrix}.$$  \hfill (8.11)

Hence, an expression for $\theta_B$ in terms of $\theta_N$ is

$$\theta_B = -M_B^{-1} M_N \theta_N = \Gamma^T \theta_N.$$  \hfill (8.12)

This implies that the objective becomes
Identifying \( \theta_N \) and \( x \) shows that the cycle basis model, (8.9), is the representation of (8.7), in the non-basic, or independent, variables associated with the basis induced by the spanning tree \( T \). Also note in (8.13) that the objective vector, \( c^T \Gamma^T \), in the cycle basis model is just the vector of reduced costs w.r.t. this basis.

Any choice of \( T \) is feasible in (8.9). A particularly interesting choice may be an optimal shortest path tree. Indeed, a basis corresponding to such an optimal tree is dual feasible in (8.9) since it is an optimal solution to the relaxation of the problem where all capacity constraints are removed. Therefore, it is a good candidate as a starting basis in the dual simplex method.

### 8.1.5 Multicommodity Minimum Cost Circulating Flow

The modelling approach above generalize straightforward to the multicommodity case. Again it is no restriction to only consider circulations. We are given a strongly connected graph \( G = (N, A) \) and a set of commodities \( K \subset N \times N \) and for each pair of commodity \( k \in K \) and arc \( a \in A \) there are also an arc cost \( (c_{ka}^k) \), individual lower \( (l_{ka}^k) \) and upper \( (u_{ka}^k) \) bound, and finally, for each arc \( a \in A \) an aggregated lower \( (l_a) \) and upper \( (u_a) \) bound. The problem is to find a circulation for each commodity that satisfies all flow bounds and minimize the total cost, i.e.

\[
\begin{align*}
\text{minimize} & \quad \sum_{k \in K} \sum_{a \in A} c_{ka}^k \theta_a^k \\
\text{subject to} & \quad \sum_{a \in \delta^+(i)} \theta_a^k - \sum_{a \in \delta^-(i)} \theta_a^k = 0, \quad i \in N, \ k \in K, \quad (8.14a) \\
& \quad l_a \leq \sum_{k \in K} \theta_a^k \leq u_a, \quad a \in A, \quad (8.14b) \\
& \quad l_a^k \leq \theta_a^k \leq u_a^k, \quad a \in A, \ k \in K. \quad (8.14c)
\end{align*}
\]

We apply the cycle basis reformulation technique to (8.14). For each \( k \in K \), let \( T^k \) be a spanning tree and \( \Gamma^k \) the associated cycle matrix. Also, denote the vector of cycle flows for \( k \in K \) by \( x^k \), hence \( \theta^k = \Gamma^k x^k \). Using this relation and the vectors \( c, l, u \) and for each \( k \in K \) also \( l^k \) and \( u^k \) for the associated given parameters yields
8.2 A Fundamental Cycle Basis Formulation

minimize \[ \sum_{k \in K} (\Gamma^k t^k)^T x^k \]
subject to
\[ l \leq \sum_{k \in K} \Gamma^k x^k \leq u, \quad (8.15a) \]
\[ l^k \leq \Gamma^k x^k \leq u^k, \quad k \in K. \quad (8.15b) \]

Observe that many constraints in (8.15) are dense. In particular, the commodity specific bounds associated with tree arcs in the partitioning in (8.2) are no longer upper and lower bounds as they were in (8.14).

8.2 A Fundamental Cycle Basis Formulation

First, we briefly repeat the formulation of the ISPR problem referred to as partial realizability. Given a strongly connected digraph \( G = (N,A) \) and a family of SP-graphs \( A^L \) for the set of destinations \( L \subseteq N \), decide if \( A^L := \{(A^l \cup \bar{D}^l, A^l)\}_{l \in L} \) is partially realizable, where \( \bar{D}^l \subset N \times N \) is the set of destination arcs, or D-arcs, see e.g. (6.3).

Denote the directed multigraph induced by \( A^L \) by \( \bar{G} = (N,\bar{A}) \) where \( \bar{A} \) consists of the ordinary arcs \( A \) and all D-arcs. We write \( \bar{A} = A \cup \bar{D} \). For each \( l \in L \), the arc set \( A \) can be partitioned as \( \bar{A} = A^l \cup \bar{D}^l \cup A^l \cup \bar{U}^l \) where \( A^l \) are SP-arcs, \( \bar{D}^l \) are D-arcs, \( \bar{A}^l \) are non-SP-arcs, and \( \bar{U}^l := \bar{A} \setminus (A^l \cup \bar{D}^l \cup \bar{A}^l) \) are unrestricted arcs.

From Chapter 6 we have that the family of SP-graphs, \( A^L \), is partially realizable if model (6.6) on page 71 is feasible. We consider the Farkas system of (6.6), i.e. model (6.7) on page 71. We repeat this model here.

\[
\text{find } \theta \\
\text{subject to } \\
\sum_{l \in L} \sum_{a \in A^l \setminus \bar{A}^l} \theta^l_a < 0, \quad (8.16a) \\
\sum_{a \in \delta^+(i)} \theta^l_a - \sum_{a \in \delta^-(i)} \theta^l_a = 0, \quad i \in N, \ l \in L, \quad (8.16b) \\
\sum_{l \in L} \theta^l_a \leq 0, \quad a \in \bar{A}, \quad (8.16c) \\
\theta^l_a \geq 0, \quad a \in \bar{U}^l, \ l \in L. \quad (8.16d)
\]

The natural interpretation of the \( \theta^l \)-variables is as the flow in \( \bar{G} \) of a commodity \( l \in L \) for the corresponding SP-graph. The node balances (8.16b) imply that all flow is circulating and (8.16c) is a capacity constraint. Also note that the non-negativity constraint (8.16d) does not apply to SP-arcs and D-arcs, i.e. reduced cost zero arcs in (6.6).
It is natural to try to exploit the circulation structure in (8.16) by alternative circulation based modelling approaches. To avoid the exponential number of variables in a Dantzig–Wolfe reformulation, we use fundamental cycle bases. The natural choice of spanning trees implies that all commodity specific flow bounds in (8.16) translate to variable bounds in our fundamental cycle basis. This is a major improvement compared to the dense constraints typically resulting when the cycle basis reformulation technique is applied, see e.g. the bound constraints (8.15b) resulting for the ordinary multicommodity problem.

To avoid dense bound constraints the spanning trees must be chosen so that the arc flow is unrestricted on tree arcs for the associated commodity. We are able to make such a choice due to the following proposition.

**Proposition 8.1**

If \((A^l \cup D^l, \widetilde{A}^l)\) is an SP-graph to destination \(l\), then the graph induced by the SP-arcs and D-arcs, i.e. \(A^l \cup D^l\), contains a spanning anti-arborescence to \(l\). Further, all arcs in \(\widetilde{D}^l\) are included in such an arborescence.

**Proof:** Since \(A^l \cup D^l\) does not induce a cycle it contains an arborescence is there is an arc emanating from each node except \(l\). This holds by the definition of \(\widetilde{D}^l\). Also, all arcs in \(\widetilde{D}^l\) are included since they are the only emanating arc from their respective nodes.

**Remark 8.2.** Proposition 8.1 and the requirement on the spanning trees to avoid dense bounds illustrate the advantage of partial realizability over compatibility; not only is partial realizability a stronger relaxation, it is also not in general possible to obtain a compact model for compatibility via cycle basis reformulation. This point is illustrated in Example 7.5 in [73] by a numerical example.

Based on the above, our fundamental cycle basis formulation follows quite naturally. For each \(l \in L\), let \(T^l \subseteq A^l \cup D^l\) be an arbitrary spanning intree rooted at \(l\). By Proposition 8.1, such an intree exists. Denote the cycle matrix of the associated fundamental cycle basis by \(\Gamma^l\). Each column in \(\Gamma^l\) is an incidence vector associated with a cycle induced by an arc not in \(T^l\). For each arc \(a \notin T^l\), the associated fundamental cycle is denoted by \(C^l a\) and the associated column by \(\gamma^l a\). The entry in \(\gamma^l a\) associated with \(a\) is \(\gamma^l a\).

Circulations are modelled by introducing flow variables for each non-tree arc and each commodity. For \(a \notin T^l\), let \(x^l a\) be the amount of flow sent in the fundamental cycle \(C^l a\). This implies, via (8.4), that the flow of commodity \(l\) on arc \(a\) is

\[
\theta^l a = \sum_{a \in A \setminus T^l} \gamma^l a x^l a, \quad a \in A, \ l \in L.
\] (8.17)

Recall that (8.17) is simplified for arcs not in the tree. The partitioning in (8.2) yields

\[
\theta^l a = x^l a, \quad a \notin T^l, \ l \in L.
\] (8.18)

By construction, all \(\theta^l\) induced by (8.17) satisfy the node balance constraints (8.16b). To fulfill a capacity constraint (8.16c), consider the amount of flow sent along an arc \(a \in A\),

\[
\sum_{l \in L} \theta^l a = \sum_{l \in L} \sum_{a \in A \setminus T^l} \gamma^l a x^l a.
\] (8.19)
This amount must be non-positive for all arcs, i.e.

$$\sum_{l \in L} \sum_{a \in A \setminus T_l} \gamma^l_a x^l_{st} \leq 0, \quad a \in A.$$  \hfill (8.20)

To guarantee that all commodity specific flow bounds (8.16d) are satisfied, the arcs are partitioned into three disjoint arc sets. For each $l \in L$, let $P^l$ be the set of arcs that are not SP-arcs, nor destination arcs, and $N^l$ the set of SP-arcs that are not in the tree, i.e.

$$P^l := \tilde{A} \setminus \left( A^l \cup \tilde{D}^l \right) \quad \text{and} \quad N^l := \left( A^l \cup \tilde{D}^l \right) \setminus T^l = A^l \setminus T^l.$$  \hfill (8.21)

This gives the partitioning, $\tilde{A} = P^l \cup N^l \cup T^l$, of all arcs in $\tilde{G}$ for all $l \in L$. Especially note that $P^l \cup N^l$ corresponds to the arcs outside the tree $T^l$ that yield fundamental cycles.

The non-negativity constraints for arcs that are not SP-arcs or destination arcs require

$$x^l_{st} \geq 0, \quad a \in P^l; \quad l \in L.$$  \hfill (8.22)

There is no non-negativity constraint for SP-arcs and no bound on $x^l_{st}$ when $\tilde{a} \in N^l \subseteq A^I$.

Consider the increment of the objective from sending flow in $C^l_a = F^l_a \cup B^l_a$. From (8.16a) it follows that all arcs but non-SP-arcs affect the objective. Hence the objective coefficient of the variable associated with arc $a$ and destination $l$ is

$$c^l_a := \sum_{a \in A \setminus A^I} \gamma^l_a = \begin{cases} |F^l_a| - |B^l_a| - 1, & \text{if } \tilde{a} \in \tilde{A}^I, \\ |F^l_a| - |B^l_a|, & \text{if } \tilde{a} \in \tilde{A} \setminus \tilde{A}^I. \end{cases}$$  \hfill (8.23)

Summarizing, our derivation implies that (8.16) is equivalent to the following fundamental cycle basis model.

\begin{align*}
\text{find} \quad & x \\
\text{subject to} \quad & \sum_{l \in L} \sum_{a \in A \setminus T^l} c^l_a x^l_{st} \leq 0, \\
& \sum_{l \in L} \sum_{a \in A \setminus T^l} \gamma^l_a x^l_{st} \leq 0, \quad a \in \tilde{A}, \\
& x^l_{st} \geq 0, \quad \tilde{a} \in P^l, \quad l \in L. \quad \hfill (8.24c)
\end{align*}

Note that $x^l_{st}$ is unrestricted for all $\tilde{a} \in N^l$ and $l \in L$.

We emphasize again that the choice of trees implies that the commodity specific flow bounds are handled implicitly by the non-negativity constraints. Another choice, where $T^l \not\subseteq A^I$, would require additional non-trivial constraints to satisfy the commodity flow bounds, potentially one for each arc and commodity, see Remark 8.2.
8.3 Properties of Partial Realizability

We derive some properties of partial realizability and models (8.16) and (8.24).

The number of variables and constraints in (8.24) is \(|L|(|m_n-n+1|)\) and \(m_n\), respectively, whereas it in (8.16) is \(|L|m_n\) and \(m_n + |L||n|\), respectively. Hence, the cycle basis model can be significantly smaller. In particular, in the practically important case where \(\tilde{G}\) is (almost) planar and \(|L| = n\) we have that the number of variables is \(O(n^2)\) in both models, but the number of constraints is \(O(n)\) for (8.24) and \(O(n^2)\) for (8.16).

By the derivation in the previous section we have that (8.16) and (8.24) are equivalent. More precisely, the relation between \(\theta\) and \(x\) defined in (8.17), is 1-to-1. Indeed, \(\Gamma^l\) has full column rank for all \(l \in L\).

**Theorem 8.1**
*There is a 1-to-1 correspondence between solutions to (8.16) and solutions to (8.24).*

The equivalence of (8.16) and (8.24) implies that all properties below can be derived without explicitly using the cycle basis model. An advantage with the latter model is that it manifests the circulation structure. We believe that it would have been harder to discover some of the results below by only considering model (8.16).

In the remainder of this section we only consider the cycle basis model. It will be convenient to write (8.24) in matrix form. To this end, let \(\phi^l\) and \(\psi^l\) be the indicator vectors of \(P^l\) and \(A^l\), respectively. Further, let \(\Phi^l\) and \(\Psi^l\) be the diagonal matrices with \(\phi^l\) and \(\psi^l\) on the diagonal, respectively, i.e.

\[
\phi^l_a = \begin{cases} 
1, & \text{if } a \in P^l, \\
0, & \text{if } a \notin P^l,
\end{cases} \quad \text{and} \quad \psi^l_a = \begin{cases} 
1, & \text{if } a \in A^l, \\
0, & \text{if } a \notin A^l.
\end{cases}
\]  

(8.25)

This implies that the objective coefficient \(c^l_a\) in (8.23) for cycle \(C^l_a = F^l_a \cup B^l_a\) is

\[
c^l_a = |F^l_a| - |B^l_a| - \psi^l_a = 1'\gamma^l_a - \psi^l_a,
\]

(8.26)

and model (8.24) can be re-written as,

\[
\begin{align*}
\text{find} & \quad x \\
\text{subject to} & \quad \sum_{l \in L} (1'\Gamma^l - 1'\Psi^l) x^l < 0, \quad \tag{8.27a} \\
& \quad \sum_{l \in L} \Gamma^l x^l \leq 0, \quad \tag{8.27b} \\
& \quad \Phi^l x^l \geq 0, \quad l \in L. \quad \tag{8.27c}
\end{align*}
\]

An important property of the model (8.27) (or any of its relatives) is that the capacity constraint (8.27b) is in a sense binding. Therefore, it may under rather general conditions be replaced with an equality constraint. This yields important practical and theoretical consequences as will be shown below.
Theorem 8.2

Let $\hat{A}^L$ be a family of SP-graphs where $|L| \geq 2$. Assume that (8.27) has a non-zero feasible solution. Then, there is a solution $\hat{x} \neq 0$ in the closure of (8.27) such that

$$\sum_{l \in L} \Gamma^l \hat{x}^l = 0. \tag{8.28}$$

Proof: Let $\hat{x}$ be a non-zero solution to (8.27) where (8.28) does not hold for $\hat{x} = \hat{x}$. Define $\gamma := \sum_{l \in L} \Gamma^l \hat{x}^l$. Let $s \geq 0$ be the vector of slack variables for the capacity constraint. Then, $\gamma + s = 0$. Since $\gamma$ is the sum of circulations, $s$ corresponds to the reverse circulation. Further, $s \geq 0$ implies that only forward arcs can be used. Hence, the slack circulation, $s$, corresponds to a directed cycle.

Take $l' \in L$. Then $s = \Gamma^{l'} x'$, for some $x'$. It follows from (8.18) that $x'_a = s_a$ for all $a \in \hat{A} \setminus T^l$. Form $\hat{x}$ from $\hat{x}$ by adding $x'$ to $\hat{x}^{l'}$. By construction, $\hat{x}$ satisfies (8.28). Since $x' \geq 0$ and $\Phi^{l'} \hat{x} \geq 0$ it follows that $\Phi^{l'} \hat{x} \geq 0$ for all $l \in L$.

Finally, $l'$ can be chosen such that $\hat{x} \neq 0$ since $|L| \geq 2$.

Remark 8.3. We assume $|L| \geq 2$ in Theorem 8.2 to avoid the pathological case where a directed cycle is "filled-in" by itself, in effect cancelling the flow and resulting in $\hat{x} = 0$.

The potential advantage of Theorem 8.2 is that the inequality in (8.27b), referred to as the capacity constraint, can be replaced by an equality constraint. To safely use equality in (8.27b), we must find conditions that guarantees that some saturating solution is improving. From the construction in the proof of Theorem 8.2 it follows that it is necessary and sufficient that the directed slack-cycle $s$ can be filled-in by an $l' \in L$ where some arc in the cycle is a non-SP-arc for $l'$.

Consider the equality formulation of model (8.27),

$$\text{find } x$$

subject to

$$\sum_{l \in L} 1^l \Phi^l x^l > 0, \tag{8.29a}$$

$$\sum_{l \in L} 1^l x^l = 0, \tag{8.29b}$$

$$\Phi^l x^l \geq 0, \quad l \in L. \tag{8.29c}$$

Observe that the objective constraint (8.29a) has been simplified by using the capacity constraint (8.29b).

Theorem 8.3

Assume that $|L| \geq 2$ and that there for every directed cycle $C$ in $G$ is an $l \in L$ where $C \cap \hat{A}^l \neq \emptyset$. Then, (8.27) is feasible if and only if (8.29) is feasible.

Proof: Trivially, if (8.27) is infeasible, then (8.29) is infeasible.

As in the proof of Theorem 8.2, construct $\hat{x}$ from $\hat{x}$ and let $C$ be the directed cycle induced by the slack. By assumption, there is an $l \in L$ such that $C \cap \hat{A}^l \neq \emptyset$. W.l.o.g.
assume that \( l \) was used for constructing \( \bar{x} \). It suffices to show that \( \bar{x} \) is improving. This follows from Proposition 6.2 on page 75 since for \( a \in C \cap \bar{A}^l \neq \emptyset \) we have \( \theta^l_a > 0 \). 

It may be cumbersome to verify the condition in Theorem 8.3. We give three stronger conditions that are easy to confirm.

**Corollary 8.1**

Assume that at least one of the following three conditions hold.

1. For each arc \( a \in A \), there is an \( l \in L \) such that \( a \in \bar{A}^l \).

2. Each node is a destination, i.e. \( L = N \).

3. Some SP-graph has been completed, i.e. \( A = A^l \cup \bar{A}^l \) and \( A^l \) does not contain a directed cycle.

Then, model (8.24) is feasible if and only if (8.29) is feasible.

**Proof:** We show that the presumption in Theorem 8.3 holds. (1) Take an arc in a directed cycle \( C \). Since there is an \( l \in L \) where \( a \in \bar{A}^l \), we have \( C \cap \bar{A}^l \neq \emptyset \). (2) Since \( L = N \), we assume w.l.o.g. that \( \delta^+(l) \subseteq \bar{A}^l \) for all \( l \in L \). This implies condition (1). Indeed, choose \( l \) for each arc \( a \in \delta^+(l) \). (3) Choose an \( l \) associated with a completed SP-graph. Since \( A^l \) does not contain a directed cycle, every cycle intersects an arc in \( \bar{A}^l \).

In the reminder of this section we assume that Theorem 8.3 applies so that we can use model (8.29). This implies that some constraints are redundant.

**Theorem 8.4**

Assume that \( |L| \geq 2 \) and that there for every directed cycle \( C \) in \( G \) is an \( l \in L \) where \( C \cap \bar{A}^l \neq \emptyset \). Then, \( n-1 \) constraints are redundant in (8.27).

**Proof:** Theorem 8.3 implies that model (8.29) can be used. The dimension of the cycle space is \( \bar{m} - n + 1 \). Hence, the rank of the constraint matrix is \( \bar{m} - n + 1 \). Since we use equalities, it follows that \( n-1 \) out of the \( \bar{m} \) constraints are redundant.
8.4 The Farkas System of the Cycle Basis Model

Eliminated. Extending this reasoning implies that all constraints associated with the non-tree arcs of the ingraph with destination \( l \) can be eliminated in (8.29). The strongest result is obtained by selecting the destination with the most arcs in \( N^l \).

**Corollary 8.2**
Assume that \( |L| \geq 2 \) and that there for every directed cycle \( C \) in \( G \) is an \( l \in L \) where \( C \cap A^l \neq \emptyset \). Then,

\[
n - 1 + \max_l |N^l|
\]

constraints can be eliminated in (8.16).

Via LP duality, there is a natural interpretation in the original partial realizability model (6.6) in Chapter 6 when we use equalities and eliminate constraints. Using equalities corresponds to allowing negative link weights in model (6.6) and eliminating constraints corresponds to fixing some link weights in (6.6). We refer to Section 7.3.1 in [73] for more details and some related results.

The property that the capacity constraints are in a sense binding can be very important in practice. If this fact is exploited by using inequalities, it induces constraint redundancy. Eliminating the redundant constraints removes some unnecessary degeneracy which can speed up computations. Also, since the capacity constraints are binding, this should be taken into account when a solution method is developed.

To illustrate the latter point we consider the following approach. It is straightforward to use a variable transformation to translate (8.16) into an ordinary multicommodity problem. However, solving the resulting problem using a method based on price-directive decomposition seems very unpromising since the success of such methods is based on few capacities being binding.

Computational experiments in a recent Master’s thesis project [202] shows that the cycle basis formulations are superior from a computational perspective. The times for solving the cycle basis formulations (using CPLEX [142]) are significantly smaller than the times for solving all other formulations considered in this thesis.

8.4 The Farkas System of the Cycle Basis Model

In this section, we derive the Farkas system of the cycle basis model in (8.24). To this end, the structure of the columns in (8.24) are first considered.

A column in (8.24) corresponds to a fundamental cycle, \( C^l_a \), determined by an intree \( T^l \) and an arc \( \bar{a} := (s,t) \notin T^l \). Let \( P^l_s \) and \( P^l_t \) be the unique paths from \( s \) and \( t \) to \( l \), respectively. Denote the induced apex by \( q \), i.e. the first common node along these paths. The paths to \( q \) are referred to as \( P_{sq} = P^l_s \setminus P^l_t \) and \( P_{tq} = P^l_t \setminus P^l_s \). This implies that \( C^l_a \) consists of two node-disjoint paths from \( s \) to \( q \), say \( P_- \) and \( P_+ \), defined as

\[
C^l_a = P_+ \cup P_- = P_{sq} \cup P_{tq} \cup \{\bar{a}\} = \bar{a} \cup (P^l_s \Delta P^l_t).
\]

Recall that the orientation of \( C^l_a \) is determined by \( \bar{a} \), i.e. the arc not in the tree. Hence, the arcs in \( P_- \) and \( P_+ \) are used backwards and forwards, respectively. They give rise to a -1 and +1 entry in the column \( \gamma_{\bar{a}} \), respectively, i.e.
\[ \gamma_a^a = \begin{cases} +1, & \text{if } a \in P_+, \\ -1, & \text{if } a \in P_- \end{cases} \]  

(8.34)

This description of columns translates to rows in the Farkas system of (8.24). Recall that the objective is \( c \) and that variables associated with arcs in \( N_l \) are unrestricted. Hence, the Farkas system of (8.24) becomes

\[
\begin{align*}
\text{find } w & \\
\text{subject to} & \\
& w_a + \sum_{a \in P_l \setminus P_{l_t}} w_a - \sum_{a \in P_l \setminus P_{l_t}} w_a \geq -c_{a_l}^{l_t}, \quad \bar{a} := (s, t) \in P_l, \ l \in L, \quad (8.35a) \\
& w_a + \sum_{a \in P_l \setminus P_{l_t}} w_a - \sum_{a \in P_l \setminus P_{l_t}} w_a = -c_{a_l}^{l_t}, \quad \bar{a} := (s, t) \in N_l, \ l \in L, \quad (8.35b) \\
& w_a \geq 0, \quad a \in \tilde{A}. \quad (8.35c)
\end{align*}
\]

The partial realizability model (6.6) in Chapter 6 and model (8.35) are equivalent. This follows from LP theory since the duals of equivalent linear programs are equivalent (alternatively, Farkas’ lemma can be used twice). To illuminate the relation more exactly, we give a constructive proof in Theorem 8.5. See also Theorem 8.6 and Figure 8.2.

To make the connection between the models more clear we use (8.23) and (8.34) to re-write the right hand side in (8.35) as

\[
-c_{a_l}^{l_t} = \begin{cases} 
-(|F| - |B|) = |P_-| - |P_+| - 1, & \text{if } \bar{a} \notin \tilde{A}^l, \\
-(|F| - |B| - 1) = |P_-| - |P_+|, & \text{if } \bar{a} \in \tilde{A}^l.
\end{cases}
\]

(8.36)

Then, substitute \( w = w + 1 \) and simplify and rename \( \bar{w} \) back to \( w \). This yields

\[
\begin{align*}
\text{find } w & \\
\text{subject to} & \\
& w_a + \sum_{a \in P_l \setminus P_{l_t}} w_a - \sum_{a \in P_l \setminus P_{l_t}} w_a = 0, \quad \bar{a} := (s, t) \in N_l, \ l \in L, \quad (8.37a) \\
& w_a + \sum_{a \in P_l \setminus P_{l_t}} w_a - \sum_{a \in P_l \setminus P_{l_t}} w_a \geq 1, \quad \bar{a} := (s, t) \in \tilde{A}^l, \ l \in L, \quad (8.37b) \\
& w_a + \sum_{a \in P_l \setminus P_{l_t}} w_a - \sum_{a \in P_l \setminus P_{l_t}} w_a \geq 0, \quad \bar{a} := (s, t) \in \tilde{U}^l, \ l \in L, \quad (8.37c) \\
& w_a \geq 1, \quad \bar{a} \in \tilde{A}. \quad (8.37d)
\end{align*}
\]

**Theorem 8.5**

*Model (6.6) is feasible if and only if (8.37) is feasible.*
8.4 The Farkas System of the Cycle Basis Model

Proof: Let \((\tilde{w}, \tilde{\pi})\) be a feasible solution to (6.6). The equality constraints in (6.6) associated with tree arcs give the distance from \(t\) to \(l\) as

\[-\tilde{\pi}_t^l = \tilde{\pi}_l^t + \sum_{a \in P_t^l} \tilde{w}_a, \tag{8.38}\]

for all \(t \in N\). Take \(l \in L\) and \(\bar{a} := (s, t) \in \tilde{A} \setminus T^l\). Consider the left hand side in (8.37),

\[
\begin{align*}
\tilde{w}_{\bar{a}} + \sum_{a \in P_t^l \setminus P_s^l} \tilde{w}_a - \sum_{a \in P_t^l \setminus P_s^l} \tilde{w}_a &= \tilde{w}_{\bar{a}} + \sum_{a \in P_t^l} \tilde{w}_a - \sum_{a \in P_t^l} \tilde{w}_a \\
&= \tilde{w}_{\bar{a}} - \tilde{\pi}_t^l + \tilde{\pi}_s^t - \tilde{\pi}_s^l = \tilde{w}_{\bar{a}} - \tilde{\pi}_t^l + \tilde{\pi}_s^t. \tag{8.39}
\end{align*}
\]

By the feasibility of (6.6), it follows that (8.37) is feasible using (8.39).

For the other direction, let \(\tilde{w}\) be a feasible solution to (8.37). An identical argument via (8.39) implies that \(\tilde{w}\) solves (6.6).

The proof of Theorem 8.5 illustrates that (8.37) is obtained when the equality constraints in (6.6) are used to eliminate the unrestricted \(\pi\)-variables, i.e. the node potentials.

The relations between the various partial realizability models considered here are summarized in Theorem 8.6 and Figure 8.2.

**Theorem 8.6**

Let \(w \in \mathbb{Q}^3\) be a weight vector. The following statements are equivalent.

1. \(w\) is part of a feasible solution to model (6.6).

2. Model (6.7) is infeasible.

3. Model (8.24) is infeasible.

4. \(w\) is a feasible solution to model (8.37).

---

**Figure 8.2:** The relation between the four partial realizability models (6.6), (6.7), (8.24) and (8.37).
The equivalence between (6.6) and (8.37) is quite interesting. It shows that only a polynomial number of paths are required in a path based model for partial realizability. This should be viewed in contrast to the exponential size path based models for the compatibility problem in the literature e.g. [25, 46]). A consequence of this equivalence is that the separation problem does not have to be solved. Alternatively, a procedure that enumerates pairs of paths induced by fundamental cycles can be used as a separation algorithm instead of solving a $k$-shortest path problem. Equivalent statements about the dual problem can be made in terms of pricing in cycles in a column generation approach for solving the natural Dantzig–Wolfe reformulation of (6.7).

**Remark 8.4.** We also note that the technique gives a polynomial size path based model for ordinary inverse shortest path problems for any norm, see [75].

In conclusion, we have presented novel formulations for partial realizability and its Farkas system based on fundamental cycle bases. The resulting models are smaller and can be solved faster. Further, emphasizing the circulation structure allowed us to deduce that all capacity constraints often are binding, which in turn implies that some of them can be eliminated.

We believe that we have shown the potential advantage of the additional structure apparent in the partial realizability problem. This gives an additional strong argument in favor of using partial realizability over compatibility, besides being a stronger relaxation of realizability. Indeed, we illustrated its beneficial practical and theoretical consequences.
Part III

A Unified Framework for Routing in IP Networks
The Shortest Path Routing Master Problem

This first chapter of the final part of the thesis serves as an introduction to shortest path routing (SPR) problems in the context of telecommunication applications. We present a model that constitutes the core of these problems, i.e., complicating side constraints required in real world applications are set aside. Once this model is presented we give a brief overview of some common modifications e.g. different objective functions, link and capacity models, routing models, etc.

The focus in this chapter is the core aspect of SPR. We introduce some notation and concepts that are used later on. In the following chapters, we delve into some selected aspects of SPR that we only treat briefly in this chapter.

Outline In Section 9.1, we formulate a minimalistic optimization in an IP network with SPR. We also give a bilinear model with administrative weights and discuss its linearization and a formulation without weights. Then, the complexity of SPR is considered in Section 9.2. Finally, various aspects of optimization problems in IP networks are reviewed in Section 9.3.

9.1 The Core of Routing Problems in IP Networks

A common property of many telecommunication problems in IP networks is that they rely on routing protocols that are based on SPR, e.g., OSPF [177] and IS-IS [76]. Since the traffic is routed along shortest paths, the flow induced by a set of administrative weights is easily determined, e.g., by Dijkstra’s algorithm. As mentioned in Part I, this is the foundation of several heuristics, but here we focus on exact solution methods for (mixed) integer linear programming (MILP) formulations of SPR problems like branch-and-cut (B&C) and combinatorial Benders’ decomposition.

The natural approach to model an SPR problem is to explicitly simulate an SPR protocol by introducing link weight variables and shortest path indicator variables. This yields
a set of bilinear constraints, see (2.22c) in model (2.22) on page 21, that has to be handled delicately. We adopt this common modelling approach and derive a model that captures the core of routing problems in IP networks below.

Throughout the chapter we only consider formulations based on arc flow. A viable option is to use path based models which corresponds to applying the Dantzig–Wolfe reformulation technique. We consider this alternative in Chapter 11 and 12.

9.1.1 Problem Formulation and a MILP Model

Let $G = (N, A)$ be a simple, strongly connected directed graph and $\mathcal{K} \subseteq N \times N$ a set of OD-pairs. For each OD-pair, $k := (o^k, d^k) \in \mathcal{K}$, the traffic demand to be routed from the origin, $o^k$, to the destination, $d^k$, is $h^k$. The set of destinations induced by all OD-pairs $\mathcal{K}$ is denoted by $L \subseteq N$, i.e.

$$L := \{ l \in N \mid \text{there exists a } k \in \mathcal{K} \text{ such that } d^k = l \}. \quad (9.1)$$

Given a destination, $l \in L$, the OD-pairs with destination $l$ is denoted by $\mathcal{K}^l$, i.e.

$$\mathcal{K}^l := \{ k \in \mathcal{K} | d^k = l \}. \quad (9.2)$$

We assume that each arc, $a \in A$, has a given capacity, $u_a > 0$. A common objective in traffic engineering problems is to find a feasible routing that minimize congestion in the network. In our basic problem formulation, we will measure congestion as the relatively maximally utilized arc, i.e. the total arc flow divided by the arc capacity. This problem formulation is considered e.g. in [46, 50]. Some alternative objectives are discussed in Section 9.3.7.

To model the core of a SPR problem as a MILP, we use three types of variables: administrative weight variables, arc flow variables and shortest path indicator variables. For each arc, $a \in A$, the administrative weight variable is $w_a \in \mathbb{Z}_+$. For each OD-pair, $k \in \mathcal{K}$, the fraction of the traffic demand, $h^k$, routed on arc $a \in A$ is denoted by $x_{a^k}$. Finally, for each destination, $l \in L$, the variable $y_{a^l}$ is used to indicate if the arc $a \in A$ is on a shortest path to $l$, i.e.

$$y_{a^l} = \begin{cases} 
1, & \text{if } a \text{ is on a shortest simple path to destination node } l, \\
0, & \text{otherwise.} 
\end{cases} \quad (9.3)$$

In SPR protocols it is required that all weights are integral and at least 1. Further, there is an upper bound, $w_{\text{MAX}}$, that in the OSPF protocol equals $w_{\text{MAX}} = 2^{16} - 1$ and in the IS-IS protocol $w_{\text{MAX}} = 2^{24} - 1$. Finally, if there are several shortest paths, the commonly accepted modelling principle is that the flow is evenly split in accordance with the equal cost multi-path (ECMP) principle. This means that the flow entering a node is divided evenly on all emanating shortest path arcs (SP-arcs) to the destination. From this problem description we obtain the bilevel model (2.22) on page 21 in Chapter 2.
minimize $\zeta$
subject to
\begin{align}
\sum_{a \in \delta^+(i)} x_a^k - \sum_{a \in \delta^-(i)} x_a^k &= b_i^k, & i \in N, k \in K, \quad (9.4a) \\
w_a + \pi_i^k - \pi_j^k &\geq 1 - y_a^l, & a = (i, j) \in A, k \in K^l, l \in L, \quad (9.4b) \\
(w_a + \pi_i^k - \pi_j^k) y_a^l &= 0, & a = (i, j) \in A, k \in K^l, l \in L, \quad (9.4c) \\
\sum_{k \in K} h_k^k x_a^k &\leq u_a \zeta, & a \in A, \quad (9.4d) \\
0 &\leq x_a^k \leq y_a^l, & a \in A, k \in K^l, l \in L, \quad (9.4e) \\
0 &\leq x_a^k - y_a^l \leq 1, & a \in \delta^+(i), i \in N, k \in K^l, l \in L, \quad (9.4f) \\
1 &\leq w_a \leq w_{MAX}, & a \in A, \quad (9.4g) \\
w &\in Z^A, v \in R^{N \times K}, x \in R^{A \times K}, y \in B^{A \times L}, \zeta \in R, \quad (9.4h)
\end{align}

where
\begin{align}
b_i^k := \begin{cases}
1, & \text{if } i = o^k \\
-1, & \text{if } i = d^k \\
0, & \text{otherwise}.
\end{cases} \quad (9.5)
\end{align}

Most of the constraints in (9.4) are common in network models. The exceptions are the SPR compatibility constraints (9.4b)-(9.4c) and the ECMP splitting constraint (9.4f). In the latter, the variable $v_a^k$, is an auxiliary splitting variable used to determine the common outflow on arcs emanating from from a given node.

An alternative to the ECMP assumption is to require that all paths are unique, i.e. the flow is unsplitable. This is referred to as unique shortest path routing (USPR). In this case, the ECMP constraints, (9.4f), are replaced by the outdegree constraints,
\begin{align}
\sum_{a \in \delta^+(i)} y_a^l &\leq 1, & i \in N, l \in L. \quad (9.4f')
\end{align}

For a node $i \in N$ where there is a demand to be sent from $i$ to destination $l$, this constraint is always satisfied at equality.

To solve models like (9.4), the bilinear constraints are typically linearized using big-$M$:s, i.e. (9.4c) is replaced by
\begin{align}
w_a + \pi_i^k - \pi_j^k &\leq My_a^l, & a = (i, j) \in A, k \in K^l. \quad (9.4c')
\end{align}

As usual, the value of $M$ has to be large enough. In the literature, $M = 2w_{MAX}$ is commonly used. However, the following proposition shows that this value of $M$ is not sufficient for general graphs.

\textbf{Theorem 9.1}
Let $d$ be the diameter of $G$. Then, $M = dw_{MAX}$ is a sufficiently large value of $M$ in
The Shortest Path Routing Master Problem

(i.e. the linearization of (9.4c). Further, there are graphs where the required value of $M$ is in $\Theta(dw_{\text{MAX}})$.

**Proof:** Sufficiency follows since, by definition, the diameter of $G$ is the longest shortest path in terms of the number of hops. Hence, no path in $G$ can be longer than $dw_{\text{MAX}}$ which is therefore an upper bound on the reduced cost of an arbitrary arc.

Necessity follows by the class of instances constructed as follows. Choose $d \geq 2$, and a weight value, $W \leq w_{\text{MAX}}$. Form the graph $G(d, W)$ by constructing a directed cycle, $C := (i_0, i_1, \ldots, i_d)$ with $d + 1$ nodes. Then add for each arc $a$ in the cycle a clique of size $W + 1$ that includes the arc $a$ and none of the nodes in the cycle $C$. Finally, add all arcs in the reversal of the cycle $C$. This graph has diameter $\lceil(d + 1)/2\rceil$. An example of the construction is shown in Figure 9.1 for $d = 3$ and $W = 3$ (the reverse arc of some arcs have not been drawn).

Given the graph $G(d, W)$, assign arc weights as follows. For each arc $(i, j)$ in the directed cycle $C$, select an $(i, j)$-path in the associated clique of length $W$ and set the weight of the arcs in this path to 1. Set the weight of all other arcs to $W$. For the example in Figure 9.1, all dashed arcs have weight 1 and all other arcs have weight $W = 3$.

By an appropriate choice of demands and arc capacities we can make the solution induced by the weights above optimal. Further, this solution can by construction not be realized by a set of weights where the weights of the arcs in the cycle are less than $W$. This implies that the reduced cost of arc $(i_0, i_1)$ w.r.t. the node potential associated with destination node $i_d$ is $(d - 1)W$. As $W$ approaches $w_{\text{MAX}}$ this goes to $\Theta(dw_{\text{MAX}})$.

**Figure 9.1:** An example of the construction of the graph $G(d, W)$ for $d = 3$ and $W = 3$. The arcs in the directed cycle, $(0, 1, 2, 3)$, are drawn with solid arcs, except for the arc $(3, 0)$, which has been replaced by the reversed arc. The paths in the cliques associated with cycle arcs are drawn with dashed arcs. The weight of a dashed arc is 1 and other arcs have weight $W = 3$.

In the light of Theorem 9.1, it is clear that the LP-relaxation of the linearized version of (9.4) can be extremely weak. Therefore, the constraints involving the administrative weights must be handled appropriately.

A common approach is to exclude the weight variables from the model, i.e. to project them out in a Benders’ decomposition fashion. Actually, since big-M constraints are involved, this is a logical/combinatorial Benders’ approach, see e.g. [80, 140, 141] for a general exposition of this methodology. This approach results in the SPR reduced cost constraints (9.4b)-(9.4c) being replaced by combinatorial Benders’ feasibility cuts that only involve binary SPR indicator variables.

We note that it is of course possible to keep the administrative weight variables and the reduced cost constraints in the model. This can help the MIP solver deep down in an
9.1 The Core of Routing Problems in IP Networks

The Core of Routing Problems in IP Networks

Enumeration tree when a lot of SPR indicator variables are binary. There, it can be more efficient to prohibit some unrealizable routing patterns by these constraints rather than to generate the associated feasibility cuts obtained by repeatedly solving an inverse shortest path routing (ISPR) subproblem. The major drawback of keeping the administrative weight variables and reduced cost constraints in the model is that the LP becomes larger, without becoming stronger, resulting in slower solution of the LP. In particular, this effect can be observed at nodes close to the root in the enumeration tree. Conceptually, when these variables and constraints are kept in the model, the Benders' feasibility cuts should rather be seen as ordinary valid inequalities and the Benders' subproblem as a (heuristic) separation problem. Therefore, the combinatorial Benders' feasibility cuts are hereafter simply referred to as valid SPR constraints.

9.1.2 A Formulation Without Administrative Weights

To obtain a model equivalent to (9.4) without the administrative weight variables they are projected out. This requires a characterization of a sufficiently large subset of the above mentioned SPR constraints. An abstract description of such a subset is given here. This description is refined in Chapter 10.

Let \((Q_0, Q_1) \subseteq (A \times L)^2\) be a pair of arc destination subsets such that the associated ISPR subproblem, i.e.

\[
\begin{align*}
\text{find } w \\
\text{subject to } \quad w_a + \pi_i^l - \pi_j^l &= 0, & a = (i, j), & (a, l) \in Q_1, \\
\text{and } w_a + \pi_i^l - \pi_j^l &\leq 1, & a = (i, j), & (a, l) \in Q_0, \\
\text{and } w_a + \pi_i^l - \pi_j^l &\geq 0, & a = (i, j), & (a, l) \in (A \times L)^2 \setminus (Q_0 \cup Q_1), \\
\text{and } w_a &\geq 1, & a \in A, 
\end{align*}
\] (9.6a)

(9.6b)

(9.6c)

(9.6d)

\[\text{is infeasible. Then, the induced SPR constraint,}\]

\[
\sum_{(a,l) \in Q_0} y_a^l + \sum_{(a,l) \in Q_1} (1 - y_a^l) \geq 1, \tag{9.7}
\]

\[\text{is valid for (9.4). Let } Q \text{ denote the family of all pairs, } (Q_0, Q_1) \subseteq (A \times L)^2, \text{ that make (9.6) infeasible, i.e.}\]

\[Q := \{ (Q_0, Q_1) \subseteq (A \times L)^2 \mid \text{model (9.6) is infeasible for } (Q_0, Q_1) \}. \tag{9.8}\]

Further, the valid inequalities induced by \(Q\) is by definition sufficient to make sure that the part of (9.4) corresponding to (9.6) is feasible. In the coming chapters, we will be particularly interested in the set \(Y^L\) of the binary vectors, \(y\), that satisfies (9.7) for all \((Q_0, Q_1) \in Q\). We define
The Shortest Path Routing Master Problem

\[ \mathcal{P}^L := \left\{ (y_a^l) \in \mathbb{R}^{A \times L} \mid \sum_{(a,l) \in Q_0} y_a^l + \sum_{(a,l) \in Q_1} (1 - y_a^l) \geq 1, (Q_0, Q_1) \in \mathcal{Q} \right\}. \quad (9.9) \]

In theory, it can be the case that the solution, \( w \), to (9.6) for \((Q_0, Q_1) \notin \mathcal{Q}\) does not satisfy (9.4g), i.e. the maximal weight is too large. Therefore, we also need to consider the largest required weight in an integral solution to (9.6) for a pair \((Q_0, Q_1) \notin \mathcal{Q}\). Denote by \( \zeta^*(Q_0, Q_1) \) the optimal value of the following problem:

\[
\begin{align*}
\text{minimize} & \quad \zeta \\
\text{subject to} & \quad (9.6a), (9.6b), (9.6c), (9.6d), \quad \zeta \geq w_a, \quad a \in A, \quad (9.10a) \\
& \quad \zeta \in \mathbb{Z}, \quad a \in A, \quad (9.10b)
\end{align*}
\]

and define \( \mathcal{Q}^* \) as the family of all pairs where the largest required weight to make (9.6) feasible is not small enough, i.e.

\[ \mathcal{Q}^* := \left\{ (Q_0, Q_1) \subseteq (A \times L)^2 \mid \zeta^*(Q_0, Q_1) > w_{\text{MAX}} \right\}. \quad (9.11) \]

Note that, by definition, \( \zeta^*(Q_0, Q_1) = \infty \) when \((Q_0, Q_1) \in \mathcal{Q}\), hence \( \mathcal{Q} \subseteq \mathcal{Q}^* \). In [45] it is shown that it is APX-hard to determine \( \zeta^*(Q_0, Q_1) \), thus we cannot expect a good characterization of the pairs \((Q_0, Q_1) \in \mathcal{Q}^* \), unless \( P = NP \). A good characterization of \( \mathcal{Q} \) was given in Chapter 6.

As above, (9.7) is a valid inequality for (9.4) also for \((Q_0, Q_1) \in \mathcal{Q}^* \setminus \mathcal{Q}\). Further, to also handle (9.4g) it suffices to include all inequalities induced by \( \mathcal{Q}^* \). Hence, the following model does not have administrative weight variables and is equivalent to (9.4).

\[
\begin{align*}
\text{minimize} & \quad \zeta \\
\text{subject to} & \quad \sum_{a \in \delta^+(i)} x_a^k - \sum_{a \in \delta^-(i)} x_a^k = b^k, \quad i \in N, \quad k \in \mathcal{K}, \quad (9.12a) \\
& \quad \sum_{(a,l) \in Q_0} y_a^l + \sum_{(a,l) \in Q_1} (1 - y_a^l) \geq 1, \quad (Q_0, Q_1) \in \mathcal{Q}^*, \quad (9.12b) \\
& \quad \sum_{k \in \mathcal{K}} h^k x_a^k \leq u_a \zeta, \quad a \in A, \quad (9.12c) \\
& \quad 0 \leq x_a^k \leq y_a^l, \quad a \in A, \quad k \in \mathcal{K}^l, \quad l \in L, \quad (9.12d) \\
& \quad 0 \leq x_a - v_i^l \leq 1 - y_a^l, \quad a \in \delta^+(i), \quad i \in N, \quad k \in \mathcal{K}^l, \quad l \in L, \quad (9.12e) \\
& \quad v \in \mathbb{R}^{N \times \mathcal{K}}, \quad x \in \mathbb{R}^{A \times \mathcal{K}}, \quad y \in \mathbb{R}^{A \times L}, \quad \zeta \in \mathbb{R}. \quad (9.12f)
\end{align*}
\]
In practice, when (9.12) is approached by B&C, the constraints associated with a pair \((Q_0, Q_1) \in Q^* \setminus Q\) are never violated (see e.g. [46]) and are therefore (at most) only separated when a new candidate for the incumbent solution is found.

9.2 Complexity of SPR Problems in IP Networks

Several results on the computational complexity of SPR problems in IP networks have been reported on in the literature, e.g. [48, 109, 192]. In [191, 192] it is shown that it is NP-complete to determine if a set of routing demands can be routed in accordance with an SPR protocol in the ECMP case given a set of arc capacities. In [109] it is shown that it is NP-hard to approximate an optimal routing given demands and a piecewise linear cost function (that implicitly handles arc capacities). Further inapproximability results are given in [48] for three problems with USPR: minimizing the maximal congestion, the capacitated and fixed cost network design problems. Here we prove that it is NP-hard to optimize over a relaxation of (9.12), i.e. to solve many SPR problems even without the arc capacity constraints.

**Theorem 9.2**

Let \(c \in \{0, \pm 1\}^{A \times L}\). Then, it is NP-hard to solve

\[
\text{maximize } \ c^t y \\
\text{subject to } \\
\sum_{a \in \delta^+ (i)} x^k_a - \sum_{a \in \delta^- (i)} x^k_a = b^k_i, \quad i \in N, \ k \in K, \quad (9.13a) \\
0 \leq x^k_a \leq y^1_a, \quad a \in A, \ k \in K^l, \ l \in L, \quad (9.13b) \\
y \in Y^L. \quad (9.13c)
\]

**Proof:** Consider an instance of the realizability problem, i.e. to decide if a family of generalized SP-graphs, \(\hat{A}_L = \{A^l, \hat{A}^l\}_{l \in L}\), is realizable. In Theorem 4.5 in Chapter 4 it was shown that realizability is NP-complete. Form the objective vector, \(c\), as follows. If \(a \in A^l\), let \(c_a = 1\) and if \(a \in \hat{A}^l\), let \(c_a = -1\), otherwise let \(c_a = 0\). Let \(z^*\) denote the optimal value of (9.13) and let \(M = \sum_{l \in L} |A^l|\), clearly \(z^* \leq M\). If \(z^* = M\), then all design variables associated with SP-arcs (arcs in \(A^l\)) must be 1 and all variables associated with non-SP-arcs (arcs in \(\hat{A}^l\)) must be 0. This implies that \(\hat{A}_L\) is realizable. Also, if \(z^* < M\), then it is not possible to set all design variables associated with SP-arcs to 1 and all variables associated with non-SP-arcs to 0, i.e. \(\hat{A}_L\) is not realizable. This proves that \(\hat{A}_L\) is realizable if and only if \(z^* = M\), and that problem (9.13) is NP-hard.

It follows immediately that problem (9.13) is NP-hard also in the USPR case since the realizability result in Chapter 4 holds also without splitting.

**Corollary 9.1**

Let \(c \in \{0, \pm 1\}^{A \times L}\). Then, problem (9.13) is NP-hard also in the USPR case, i.e. when the outdegree constraints (9.4f) are augmented to (9.13).
Note that there is a significant difference between our results and earlier NP-completeness and NP-hardness results in [48, 109, 192]. The proof in [192] indicates that the ECMP splitting and capacity part of SPR problems is the source of hardness. Indeed, the SPR protocol is actually not considered at all since only a single SP-graph is used. The splitting and capacity aspects are central also in the proof in [109], the difference is that SPR is also taken into account. The USPR problems in [48] involve arc capacities and the proofs include the SPR element.

Our results show that the routing in accordance with an SPR protocol alone is a source of hardness in SPR problems.

We conclude this chapter with a brief overview of what we believe are some of the most important aspects of optimization problems in IP networks and how they can be modelled in a MILP. The purpose is to provide a unified framework for routing in IP networks. The aspects can be thought of as a set of modules that should be put together to form a network optimization problem.

9.3 Optimization Problems in IP Networks

Traditionally, mathematical programming has been used to solve many planning problems in telecommunications. Several tasks of managing a telecommunication network fit into one of the following categories: topological design, routing and restoration.

In the long term planning, strategical and topological design issues are considered, such as node location and network dimensioning. Mid term planning involves re-design of a given topology, e.g. as a consequence of increased traffic demand, and includes dimensioning and expansion of a given network along with traffic (re-)routing. A common attribute of long and mid term strategic design problems is that the objective is often to minimize an estimated total design and routing cost. Several mathematical models have been developed for these strategic problems, and many of them also take other issues into account, e.g. survivability, robustness and fairness. These models are often based on (capacitated) facility location and multicommodity design and routing models. The latter problems have been studied extensively in the literature.

In the short term planning, also called operational planning, the objective for the network operator is to utilize the available resources as efficiently as possible without changing the actual design, i.e. only by re-configuration of the routing. This may for instance be due to increased traffic demand or hardware failure, the latter case is referred to as restoration. In operational planning problems, the objective is often to maximize a QoS measure, e.g. low congestion or link load. We refer to the operational planning problems as traffic engineering.

Below, we give short introductions to the aspects we believe are most important for telecommunication network applications and, in particular, for applications in IP networks. The main features of the (IP) network optimization problems we consider are characterized by the following elements: link load model (Section 9.3.2), capacity model (Section 9.3.3), demand model (Section 9.3.4), routing model (Section 9.3.5), some other aspects (Section 9.3.6) and finally the objective function (Section 9.3.7). To properly formulate an optimization problem in a network, these elements must be specified.
9.3 Optimization Problems in IP Networks

9.3.1 A Basis for Network Optimization Problem Formulations

To describe the elements required to formulate an optimization problem in a network we use the well known uncapacitated network design (UND) problem as a starting point, see e.g. [18, 137, 191]. One version of this problem is as follows. Let \( G \) and \( \mathcal{K} \) be as above and suppose that for each \( k \in \mathcal{K} \), the demand is \( h^k = 1 \), i.e. one unit of flow must be sent from the origin, \( o^k \), to the destination, \( d^k \). Further, an arc, \( a \in \mathcal{A} \), can only carry flow if it has been opened and the cost of opening arc \( a \) is \( c_a \). The objective is to determine a minimum cost opening of arcs that allows a feasible flow for each commodity.

A standard mathematical model for the UND problem is obtained by using a flow variable, \( x^k_a \), that represents the amount of flow sent on arc \( a \in \mathcal{A} \) for commodity, \( k \in \mathcal{K} \). Then, a binary variable, \( y_a \), is set to 1 if and only if arc \( a \) is opened. This yields the UND problem,

\[
\begin{align*}
\text{minimize} & \quad \sum_{a \in \mathcal{A}} c_a y_a \\
\text{subject to} & \quad \sum_{a \in \mathcal{A}^+ (i)} x^k_a - \sum_{a \in \mathcal{A}^- (i)} x^k_a = b^k_i, \quad i \in \mathcal{N}, \ k \in \mathcal{K}, \quad (9.14a) \\
& \quad x^k_a \leq y_a, \quad a \in \mathcal{A}, \ k \in \mathcal{K}, \quad (9.14b) \\
& \quad x^k_a \geq 0, \quad a \in \mathcal{A}, \ k \in \mathcal{K}, \quad (9.14c) \\
& \quad y_a \in \mathbb{B}, \quad a \in \mathcal{A}, \quad (9.14d)
\end{align*}
\]

where, as usual, \( b^k_i \) is the node balance, see e.g. (9.5).

The aspects of network optimization problems mentioned above are presented in the following sections where we in particular show how to adapt and extend model (9.14).

9.3.2 The Link Load Model

In the UND problem, or model (9.14), as defined above it is assumed that an arc can be used to send flow if and only if the arc is opened. Another option is to allow flow to be sent on the arc \( a = (i, j) \in \mathcal{A} \) also if the arc \( (j, i) \in \mathcal{A} \) is opened. These are two possible link load models. They are referred to as the directed and bi-directed link load model, respectively. A third option is the undirected link load model. To properly explain our usage of the terms link, link load and link load model, we first distinguish between arcs and edges. As usual, an arc, \( a = (i, j) \), is an ordered pair of nodes taking the direction into account and an edge, \( e = \{i, j\} \), is an unordered pair of nodes ignoring the direction. Given the arc set \( \mathcal{A} \), the associated set of edges denoted by \( \mathcal{E} \). The load of a link is the total flow sent on the link where the meaning of the link load models is as follows, depending on whether the link graph is directed or undirected.

**Directed** In the directed link load model, a link corresponds to an arc; a directed link can be used only in one direction. The load, \( f_a \), of link \( a \in \mathcal{A} \) is determined by the total flow on the corresponding arc. If the capacity of link \( a \) is \( u_a \) then the load must not exceed this capacity, i.e.
The Shortest Path Routing Master Problem

\[ f_a = \sum_{k \in K} x_a^k, \quad \text{and} \quad f_a \leq u_a. \]  \hspace{1cm} (9.15)

**Undirected** In the undirected link load model a link corresponds to an edge; an undirected link can be used in both directions, i.e. both arcs associated with the edge can be used. The load, \( f_e \), of link \( e = \{i, j\} \in E \) is determined by the total flow on the corresponding arcs, i.e.

\[ f_e = f_a + f_\bar{a} = \sum_{k \in K} (x_a^k + x_\bar{a}^k), \quad a = (i, j), \ \bar{a} = (j, i), \ e = \{i, j\}. \]  \hspace{1cm} (9.16a)

If the capacity of link \( e \) is \( u_e \), then this is an upper bound on the load, i.e.

\[ f_e = f_a + f_\bar{a} \leq u_e, \quad a = (i, j), \ \bar{a} = (j, i), \ e = \{i, j\}. \]  \hspace{1cm} (9.16b)

**Bi-directed** Finally, in the bi-directed link load model a link in a sense corresponds both to an arc and an edge. For the load calculation, a link is an arc; the load \( f_a \), of link \( a \in A \) is as in the directed case, i.e.

\[ f_a = \sum_{k \in K} x_a^k. \]  \hspace{1cm} (9.17a)

For the capacity restriction, a link is an edge. If the capacity of link \( e \in E \) is \( u_e \), then the load of none of the associated arcs must be greater than this capacity, i.e.

\[ f_a \leq u_e, \quad \text{and} \quad f_\bar{a} \leq u_e, \quad a = (i, j), \ \bar{a} = (j, i), \ e = \{i, j\}. \]  \hspace{1cm} (9.17b)

To ease the presentation we assume that the directed link load model, (9.15), is used in the remaining part of this chapter. The adaption to the undirected case, (9.16), and the bi-directed case, (9.17), is straightforward. It is apparent from the equations above, (9.15), (9.16b), and (9.17b), that the link load model is closely related to the capacity model. We discuss this next.

### 9.3.3 The Link Capacity Model

In the link load model above we assumed that each link had a capacity. Let this capacity be modelled via the variable \( u_a \) for link \( a \in A \). We consider some common link capacity models for how the capacity can be restricted.

**Fixed** The simplest capacity model is obtained by fixing \( u_a \) at a given value, \( \bar{u}_a \), i.e. set

\[ u_a = \bar{u}_a. \]  \hspace{1cm} (9.18)

This corresponds the scenario with pre-installed capacities and no possibility of expanding these. This is common in traffic engineering problems when the routing in an operational state is considered.
Continuous Another simple capacity model is to allow $u_a$ to take any non-negative value, i.e.

$$u_a \in \mathbb{R}_+.$$  \hspace{1cm} (9.19)

This typically means that capacity is for instance rented and the cost depends on how much a link is utilized.

Modular An important and frequently used capacity model in design of new and expansion of existing networks is the following. Assume that there is a set of modules, $M$, available for installation on any link. If module $m \in M$ is installed on link $a \in A$ it increases the link capacity by $\bar{u}^m$. Let $z^m_a$ be an integer variable modelling the number of modules of type $m \in M$ that are installed on link $a \in A$. Also, let $\bar{z}^m_a$ be an upper bound on the number of modules of type $m \in M$ that can be installed on link $a \in A$. This yields that the capacity, $u_a$, is determined by

$$u_a = \sum_{m \in M} \bar{u}^m z^m_a;$$ \hspace{1cm} (9.20a)

where

$$0 \leq z^m_a \leq \bar{z}^m_a, \text{ and } \bar{z}^m_a \in \mathbb{Z}, \text{ } m \in M.$$ \hspace{1cm} (9.20b)

In practice, the number of modules, $|M|$, is usually small and their capacities are often divisible, i.e. they are integer multiples of each other. We assume that the cost-to-capacity-ratio of modules decrease with increasing capacity.

There has been much research on network design problems with modular capacities when there is only a single module e.g. [116, 138, 176] and many more including the below. Some authors consider the problem when there are few modules e.g. [169, 129, 167]). The case with many modules has not received much attention, an exception is [15]. In particular, the polyhedral approach and the class of cutset inequalities have proven to be very successful in the solution of these problems, see e.g. [15, 21, 40, 41, 129, 168, 169, 197].

9.3.4 The Demand Model

In the UND model above, (9.14), it is assumed that the demand is fixed. In practice, it is likely that it is uncertain and that a conservative estimate is used. Often, demands are estimated using for instance link traffic measurements, see e.g. [130, 228]. To "guarantee" that the routing is feasible w.r.t. the realized demands, overestimates are used. This leads to unnecessary resource utilization (capital, energy, bandwidth allocation, etc.). Alternatively, the uncertainty can in theory be dealt with using stochastic optimization. From a computational perspective it seems to be better to use robust optimization.
Modelling Uncertainty Using Robust Optimization

Robust optimization is an emerging paradigm of mathematical programming that lately has become very popular, see e.g. [29, 33, 34, 36]. The central idea is to use an uncertainty set, \( \mathcal{A} \), to represent uncertainties in the data. Then, the deterministic problem,

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad Ax \geq b, \\
& \quad x \in \mathbb{R}^p_+ \times \mathbb{Z}^q_+,
\end{align*}
\]

is replaced by a worst case robust optimization problem, where the constraint must hold for all matrices \( A \in \mathcal{A} \). We obtain the robust counterpart,

\[
\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad Ax \geq b, \quad A \in \mathcal{A}, \\
& \quad x \in \mathbb{R}^p_+ \times \mathbb{Z}^q_+.
\end{align*}
\]

The uncertainty set, \( \mathcal{A} \), plays a crucial role in robust optimization. Already 1973, Soyster introduced robust LPs using an over-conservative box-approach for \( \mathcal{A} \). A very natural, less conservative idea primarily applicable to convex programs is to use ellipsoids as uncertainty sets, see [31, 32, 98, 99]. The drawback of ellipsoids is that the linear structure in (9.21) vanishes. In our option, the best suited and also the most popular approach for (possibly mixed integer) linear programs is to use the concept of \( \Gamma \)-uncertainty introduced in [38, 39].

Modelling Uncertain Demands

In the case of network design or routing problems, the data uncertainty lies in the demand. Several approaches have been suggested in the literature to deal with demand uncertainty. We present the most common and also the previously unexplored possibility of using conditional value-at-risk (CVaR) commonly occurring in investment problems.

We denote the demand uncertainty set by \( \mathcal{D} \subseteq \mathbb{R}^K_+ \) and use the variable \( d \in \mathbb{R}^K_+ \) to represent the vector of demands. Introducing demand uncertainty essentially only affects the capacity constraint. Recall that \( u_a \) is the capacity of link \( a \in \mathcal{A} \). This yields,

\[
\sum_{k \in K} d^k x^k_a \leq u_a, \quad \forall d \in \mathcal{D} \iff \max_{d \in \mathcal{D}} \sum_{k \in K} d^k x^k_a \leq u_a.
\]

Observe that the right capacity constraint in (9.23) is non-linear. This issue is usually overcome in one of the following ways. If \( \mathcal{D} \) is a polytope, it (by convexity) suffices to enumerate its extreme points; possibly implicitly via separation if they are exponentially many. If \( \mathcal{D} \) is a polytope with polynomially many facets it is possible to use a compact formulation based on a dualization technique, this is illustrated below. Finally, if \( \mathcal{D} \) is a
convex set given by a separation oracle, it is in principle possible to use enumeration to approximate \( D \) by linear constraints within arbitrary precision.

The following uncertainty sets are common in the literature.

**Fixed** The simplest demand model is obtained by fixing \( d \) at its estimated value, \( \bar{d} \), i.e. set \( D = \{ \bar{d} \} \). This implies that uncertainty is not taken into account. Hence, the capacity constraint is not affected, i.e. it remains to be

\[
\sum_{k \in K} a_k x_k \leq u_a, \quad a \in A.
\] (9.24)

**Hose model** The hose model is introduced in [101], see also [92]. It is based on the assumption that each node, \( i \in N \), has an upper bound on the traffic outflow, \( b_i^+ \), and inflow, \( b_i^- \). This yields that the demand uncertainty set becomes,

\[
D = \left\{ d \in \mathbb{R}_+^K \mid \sum_{k \in K} d_k \leq b_i^+, \quad \sum_{k \in K} d_k \leq b_i^-, \quad i \in N \right\}.
\] (9.25)

To incorporate this into the capacity constraint, the standard robust LP dualization trick is used; this is fruitful since the number of facets of \( D \) is small. Treating (for the moment) \( x \) as fixed it is feasible to replace the optimization problem in (9.23) by its LP dual, i.e.

\[
\begin{align}
\text{minimize} & \quad \sum_{i \in N} \left( b_i^+ \alpha_i + b_i^- \beta_i \right) \\
\text{subject to} & \quad \alpha_k + \beta_k \geq x_k, \quad k \in K, \\
& \quad \alpha_i, \beta_i \geq 0, \quad i \in N.
\end{align}
\] (9.26a)

(9.26b)

From strong duality it follows that the robust capacity constraint (9.23) is feasible if and only if the following hose model capacity constraint is feasible,

\[
\sum_{i \in N} \left( b_i^+ \alpha_i + b_i^- \beta_i \right) \leq u_a,
\] (9.27)

where also the auxiliary dual constraints (9.26) are enforced.

The hose model has been used together with IP routing, e.g. in [7, 9, 10, 149]. Its main strength is its simplicity. Indeed, it only relies on aggregate node information (which can be assumed to be available). This is also its weakness since this alone is not likely to capture the fluctuations in the point-to-point traffic data.

**\( \Gamma \)-uncertainty** In the \( \Gamma \)-uncertainty model introduced in [38, 39] it is assumed that at most fixed number, \( \Gamma \), of demand values deviate from their nominal value. More
precisely, let $\hat{d} \in \mathbb{R}_+^K$ be an estimate of the nominal demands and $\hat{d} \in \mathbb{R}_+^K$ be an estimate of the element-wise maximal deviations from $d$. Assuming that at most $\Gamma$ deviations occur simultaneously yields

$$D = \text{conv} \left\{ d \in \mathbb{R}_+^K \mid d_k = \hat{d}_k + \sigma_k \hat{d}_k, \sum_{k \in K} \sigma_k \leq \Gamma, \sigma \in \mathbb{R}_+^K \right\}. \quad (9.28)$$

Observe that for fixed $x$ the integrality constraint on $\pi$ in (9.28) can be replaced by $0 \leq \sigma \leq 1$. Then, the maximization problem in the robust capacity constraint (9.23) becomes

$$\begin{align*}
\text{maximize} & \quad \sum_{k \in K} d_k x_k + \sum_{k \in K} \sigma_k \hat{d}_k x^k \\
\text{subject to} & \quad \sum_{k \in K} \sigma_k \leq \Gamma, \\
& \quad 0 \leq \sigma \leq 1, \quad k \in K, \quad (9.29a) \\
& \quad (9.29b)
\end{align*}$$

and its dual,

$$\begin{align*}
\text{minimize} & \quad \sum_{k \in K} \hat{d}_k x_k + \Gamma \pi_a + \sum_{k \in K} p^k_a \\
\text{subject to} & \quad \pi_a + p^k_a \geq \hat{d}_k x_k, \quad k \in K, \quad (9.30a) \\
& \quad p^k_a, \pi_a \geq 0, \quad k \in K. \quad (9.30b)
\end{align*}$$

Again, it follows from strong duality it that the robust capacity constraint (9.23) is feasible if and only if the following $\Gamma$-uncertainty capacity constraint is feasible,

$$\sum_{k \in K} d_k x_k + \Gamma \pi_a + \sum_{k \in K} p^k_a \leq u_a, \quad (9.31)$$

where also the auxiliary dual constraints (9.30) are enforced.

$\Gamma$-uncertainty is quite commonly used in robust network design formulations, see e.g. [155, 154, 156].

**CVaR** The CVaR is originally a risk measure used in financial mathematics. Given a probability distribution of outcomes and a quantile-level, $\alpha$, CVaR is the expected value of the outcome in the $\alpha$-quantile. It has become a popular modelling tool in mathematical programming due to its favorable properties. We consider the important special case were the distribution is given as a finite set of scenarios (occurring
with equal probability). This implies that it is possible to minimize CVaR or setting an upper bound on CVaR using only linear constraints. See further [200]. The relation between CVaR and robust optimization is explored in [37]. In particular, the uncertainty polytope \( \mathcal{D} \) induced by an \( \alpha \) and a set of scenarios is derived. Denote the set of possible demand realizations, i.e. scenarios, by \( \{d_i\}_{i=1}^s \), where \( d_i \in \mathbb{R}_+^K \).

Using CVaR at level \( \alpha \) yields the robust capacity constraint,

\[
\zeta_a + \frac{1}{s(1-\alpha)} \sum_{i=1}^s \left[ \sum_{k \in K} d^k x^k_a - \zeta_a \right]^+ \leq u_a, \tag{9.32}
\]

To model (9.32) with linear constraints some auxiliary variables are introduced to handle the max-operator, \( [x]^+ := \max\{x, 0\} \), in the summation over scenarios. For \( a \in A \) and \( i = 1, \ldots, s \), we use \( p_{ia} \) to model this, i.e.

\[
\zeta_a + \frac{1}{s(1-\alpha)} \sum_{i=1}^s p_{ia} \leq u_a, \tag{9.33a}
\]

\[
p_{ia} \geq \sum_{k \in K} d^k x^k_a - \zeta_a, \quad i = 1, \ldots, s, \tag{9.33b}
\]

\[
p_{ia} \geq 0, \quad i = 1, \ldots, s. \tag{9.33c}
\]

Our reason for suggesting to consider CVaR instead of the robust alternatives above is that it via scenarios takes the relation between demand peaks into account. In a sense, using \( \Gamma \)-uncertainty is an implicit assumption of independence of the peaks for different OD-pairs. Of course, correlations can be handled by a variable transformation based on the Cholesky factorization of the correlation matrix, but more complicated relations cannot be captured.

There are numerous papers using CVaR to model uncertainty. To the best of our knowledge, it has however not been used in the network design or routing context, i.e. to model uncertain demands.

**Polyhedral** The concept polyhedral demand was introduced in [27] where it is assumed that \( \mathcal{D} \) is given by a polytope. Observe that this general case subsumes all demand models described above.

The maximum solution to (9.23) is attained in an extreme point of \( \mathcal{D} \). Therefore, it suffices to enumerate the constraints associated with extreme points of \( \mathcal{D} \) to model the robust capacity constraint. If \( \mathcal{D} \) has exponentially many extreme points a separation problem can be used to implicitly enumerate constraints. Let \( \hat{x} \) be a flow part of an optimal solution to a relaxation of (9.23), then the separation problem,

\[
\max_{d \in \mathcal{D}} \sum_{k \in K} d^k \hat{x}^k_a, \tag{9.34}
\]

with optimal solution and value, \( \hat{d} \) and \( u^* \), respectively, implies that the constraint
is violated and should be augmented to the formulation if and only if \( u^* > u_a \).

When \( \mathcal{D} \) has exponentially many extreme points, but only few facets, the dualization technique used above should be considered.

### 9.3.5 The Routing Model

There are two aspects of routing that must be considered. First, in the case of uncertain demands, an important question is if the routing is allowed to adapt to the realization of the demand or not, i.e. is it static or dynamic. A second question is, given a demand realization, what flow patterns are feasible?

#### Static versus Dynamic Robust Routing

As above, the uncertainty set is denoted by \( \mathcal{D} \). Given a demand realization, \( d \in \mathcal{D} \), the realized flow on arc \( a \in A \) for commodity \( k \in K \) is described by the function \( x^k_a(d) \) and the capacity constraint is

\[
\max_{d \in \mathcal{D}} \sum_{k \in K} x^k_a(d) \leq u_a. \tag{9.36}
\]

There are two routing paradigms that are used to handle the realization of uncertain demands: static and dynamic routing. We describe these and some intermediate paradigms since the latter may be too general to be applicable in practice.

**Static** In static routing, also called oblivious routing, a single routing template is used for all possible demand realizations, i.e. the end-to-end paths and splitting percentages are fixed. Using the notation above, this yields

\[
x^k_a(d) = d^k x^k_a. \tag{9.37}
\]

Observe that static routing was implicitly assumed throughout the robust demand uncertainty section, cf. Equations (9.23), (9.36) and (9.37).

Static routing is introduced in [27]. It is a paradigm that can be implemented in practice and it has good computational properties. It is the dominating routing model, see e.g. [6, 7, 10, 12, 154].

**Dynamic** A more flexible approach to linear robust optimization in general, and routing problems in particular, is to allow some variables to be adjusted once the realization of the data is known, see [16, 30]. Adjusting the variables is called recourse. In the case of routing problems, the routing variables, \( x^k_a(d) \), are adjustable and unrestricted recourse is referred to as dynamic routing. Allowing unrestricted recourse in general makes robust linear programs NP-hard [30], including routing problems [77, 131]. Another problem with dynamic routing is how to actually implement it in real-world networks, see [26].
9.3 Optimization Problems in IP Networks

Dynamic routing is obviously a relaxation of static routing. The main result in [122] is that the worst case ratio of an optimal static design compared to an optimal dynamic design is \( \Omega(\log |N|) \) even in the special case of the Hose model. See also [181] for further comparisons including routing template restrictions. Computational experiments have been carried out to investigate the actual ratio in practice, e.g. for real-world instances from SNDlib (see [182]) and Hose and \( \Gamma \)-uncertainty, in [174] and [196], respectively.

**Semi-dynamic** Since dynamic routing yields better solutions than static routing, but on the other hand is intractable, several attempts have been made to allow some restricted recourse, e.g. in [28, 207]. A theoretical comparison of the most common semi-dynamic routing principles is carried out in [195]. A drawback with many restrictions on the recourse strategies is that the resulting problem often remains hard. An exception is given in [112]. The most important exception may be affine recourse, which for robust LPs yields polynomial solvability [30].

**Affine recourse** The tractability of affine recourse makes it the most important restricted recourse principle. In our context, this is referred to as affine routing, see e.g. [17, 183, 184, 196]. Here, the function \( x^k_a(d) \) must be linear in \( d \), i.e.

\[
 x^k_a(d) = \alpha^k_a + \sum_{h \in K} \beta^{kh}_a d^h, \quad a \in A, k \in K,
\]

for some parameters \( \alpha^k_a \) and \( \beta^{kh}_a \). In [196], the affine function, \( x^k_a(d) \), is re-written as the sum of a routing template and circulations. Then, the capacity constraint (9.36) and the lower bound \( x^k_a(d) \geq 0 \) can be dualized for all \( a \in A \). This yields a polynomial (but quite large), reasonably structured LP as the robust counterpart. Hence, affine routing yields a tractable alternative in between static and dynamic routing which seems to yield very good approximations of optimal dynamic routings [196].

**Routing Template Restrictions**

The second aspect of the routing model is which flow patterns, i.e. routing templates, that are feasible. Some common restrictions will be expressed in terms of conditions on a static routing template, \( x \in R^{A \times K}_+ \). Recall that

\[
 \sum_{a \in \delta^+(i)} x^k_a - \sum_{a \in \delta^-(i)} x^k_a = b^l_i, \quad a \in A, k \in K^l, l \in N.
\]

(9.39)

where \( b^l_i \) is defined in (9.5). Modelling some routing template restrictions requires flow indicator variables. We collect all OD-pairs with the same destination in a set; for each \( l \in N \), define \( K^l = \{ i \in N \mid i = d^k \text{ for some } k \in K \} \). The reason for collecting over destinations and not origins becomes clear when the equal-split principle is considered below. For each \( l \in N \) and \( a \in A \), the binary variable \( y^l_a \) is 1 if there is flow on arc \( a \) for some \( k \in K^l \), i.e.

\[
 x^k_a \leq y^l_a, \quad a \in A, k \in K^l, l \in N.
\]

(9.40)
There is sometimes a lower bound $\epsilon > 0$ on the strictly positive flow. Hence, $y_{la}^l = 1$ if and only if $x_{ka}^k \geq \epsilon$ for some $k \in \mathcal{K}$. This can be modelled by the constraints:

$$y_{la}^l \leq \frac{1}{\epsilon} \sum_{k \in \mathcal{K}} x_{ka}^k, \quad a \in A, l \in N.$$  \hspace{1cm} (9.41)

In particular, if single-path or tree routing is used, $\epsilon$ can be set to 1. These two routing restrictions and three others are described below.

**Unrestricted** The most common routing model is to allow all routing templates. This absence of restriction implies that some routing problems can be solved as pure multi-commodity flow problems in polynomial time as LPs.

**Single-path** An important restriction is to only allow one path to be used for each OD-pair, i.e. the flow is unsplittable (also called non-bifurcated). This restriction may be very relevant in practice since it is generally much easier to implement the resulting routing templates in routers. It is also easy to model this by forcing $x$ to be binary, i.e.

$$x_{ka}^k \in \mathbb{B} \quad a \in A, k \in \mathcal{K}.$$  \hspace{1cm} (9.42)

This integrality requirement in general makes the resulting problem NP-hard.

**Tree** A special case of single-path routing is tree routing where it is required that routing templates form in-trees (or anti-arborescences). This is accomplished by forcing outdegrees to be at most 1,

$$\sum_{a \in \delta^+(i)} y_{la}^l \leq 1, \quad i \in N, l \in N.$$  \hspace{1cm} (9.43)

Observe that this constraint implicitly makes $x$ binary.

**Equal-split** In equal-split routing the flow can be split at a node $i$, but it has to be divided evenly on all arcs emanating from $i$ carrying flow. A further restriction, referred to as equal cost multipath (ECMP) splitting, is to force the same splitting for all OD-pairs with the same destination. Below, we only consider ECMP splitting. To model equal-split routing the binary destination dependent indicator variables, $y_{la}^l$, should be replaced by commodity dependent indicator variables.

In ECMP splitting, each node $i \in N$ and commodity $k \in \mathcal{K}$ has a common outflow value, $v_{ik}^k$, defined by

$$v_{ik}^k = \frac{1}{\sum_{a \in \delta^+(i)} y_{la}^l}, \quad i \in N, l \in N,$$  \hspace{1cm} (9.44)

inducing the flow to be
\[ x^k_a = y^l_a v^k_l, \quad a \in \delta^+(i), i \in N, k \in \mathcal{K}, l \in N. \]  

(9.45)

Constraints (9.44) and (9.45) can be modelled by the linear constraints,

\[ 0 \leq v^k_i - x^k_a \leq 1 - y^l_a, \quad a \in \delta^+(i), i \in N, l \in N, \]  

(9.46)

or

\[ x^k_{a'} - x^k_{a''} \leq 1 - y^l_{a''}, \quad a', a'' \in \delta^+(i), i \in N, l \in N. \]  

(9.47)

It is straightforward to show that (9.47) is the result of projecting out the auxiliary outflow variables, \( v \), in (9.46). In Section 12.1, we provide stronger extended formulations for ECMP splitting, including one that projects to the convex hull of the single node splitting polytope.

Shortest path routing In shortest path routing all routing templates must be induced by shortest paths by some common metric, \( w \in \mathbb{R}^A \). This implies that an arc \( a \in A \) may only carry flow for commodity \( k \in \mathcal{K} \) if \( a \) is on a shortest path from \( o^k \) to \( d^k \) w.r.t. \( w \). When there are multiple shortest paths, it is commonly required that ECMP splitting is adopted. An alternative is to require that shortest paths are unique, this induces tree routing. SPR is treated thoroughly in this thesis.

9.3.6 Other Aspects of Network Problems

There are many other aspects of routing in optimization problems in IP networks, but a thorough treatment is out of the scope of this thesis. Therefore, we mention only survivability, also referred to as restoration. This means to take into account the possibility that some node or link fails, e.g. due to a hardware failure. See e.g. [84, 214, 220] for more on survivability.

Some failures can be taken into account implicitly by introducing redundancy, e.g. requiring that the network is over-connected or over-capacitated, to be able to likely handle a small number of failures. An explicit option is to use failure scenarios where a reasonable subset of all failure events are enumerated, e.g. all scenarios where a single link fails or the most important links fail. Then, (possibly softened) constraints are enforced also in these failure scenarios.

One of the features of routing by SPR protocols is that they cope with failures in a very natural way; if a link fails, the new shortest path (and hence routing) is easily determined by re-calculting the shortest paths. (Of course, there is no guarantee that the link capacities are large enough to handle the induced flow unless this is explicitly taken into account.)

9.3.7 The Objective Function

In the basic UND problem formulation (9.14), the objective is to minimize the total installation cost. A general network design and routing problem is per se multi-objective.
Indeed, the natural objective is to minimize some estimated installation and operational cost while at the same time maximizing some quality of service (QoS) measure provided to the users of the network. These two objectives are typically conflicting. There are several means to handle this issue e.g. combining the measures in a single objective, maximizing QoS under a budget constraint or minimizing the cost under QoS constraints. We will not consider the multi-objective aspect; instead we focus on one objective at a time.

**Objective Functions Based Installation and Operational Costs**

The cost incurred for a network provider can have two components. There is an installation cost and an operational cost for each link. In the case of an existing network that cannot be modified, only the latter cost exists. This is referred to as a routing problem or a traffic engineering problem. If a network is expanded or setup from scratch, the former cost is typically dominating. This is referred to as a topological network design problem.

First consider the operational cost, it is commonly assumed to be proportional to the amount of traffic on the link. In some problems this cost is neglected, e.g. in (9.14) above. In long-term planning problems, i.e. topological network design problems, the operational cost must be put in a measure comparable with the installation costs. This can be handled for instance by discounting the estimated operational cost over time, i.e. by a net present value approach. We assume that the operational cost per unit of flow is equal for all commodities and denote it by $c_a$ for link $a \in A$.

The installation cost occurs only in topological network design problems. It is closely connected to the capacity model. Recall from Section 9.3.3 that the capacity is installed in modules; there may be several module sizes. Let the capacity constraint be as in (9.20) and let $g_m^a$ be the cost of installing one piece of module $m \in M$ on link $a \in A$.

This yields the total installation and operational cost,

$$\sum_{a \in A} \left( c_a f_a + \sum_{m \in M} g_m^a z_m^a \right),$$

where, as before $f_a$ is the total flow on link $a \in A$ and $z_m^a$ is the number of modules of type $m \in M$ that are installed on link $a \in A$.

**Objective Functions Based on a Common QoS Measure**

The most important responsibility for a network operator is to guarantee that the network delivers an acceptable level of the QoS in all operational states. It is possible to approach this task implicitly in failure scenarios, e.g. by robustness and survivability measures. We only focus on the normal operating state and consider QoS measures based on the following statement:

"One of the most important performance measures of a data network is the average delay required to deliver a packet from origin to destination." [35]

To model the average packet delay, some underlying probability distribution for packet sizes and inter-arrival times is required. If it is assumed that both these quantities are exponentially distributed, then the arrival times follow a Poisson process, and results in a an $M/M/1$ queueing system. This implies that the average delay for a packet is given by
where \( \lambda \) is the average arrival rate (packets per second), \( \mu \) is the average service rate (bandwidth / average packet size) and \( \rho = \lambda / \mu \) is the average utilization of the link. To optimize the QoS in the system, i.e. not on a link basis, the interaction between links is typically neglected. This is referred to as the Kleinrock independence approximation. We consider the four approaches listed in Table 9.1 to optimize the system QoS, see e.g. [35, 46, 107, 191, 198].

We use \( \lambda = f_a \), i.e. the average arrival rate equals the link load, and \( \mu = u_a \), i.e. the average service rate is proportional to the bandwidth measured as the link capacity (the average packet size is set to 1).

<table>
<thead>
<tr>
<th>Objective</th>
<th>Objective function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimize maximum utilization</td>
<td>( \min \max_{a \in A} \frac{f_a}{u_a} )</td>
</tr>
<tr>
<td>Minimize average utilization</td>
<td>( \min \sum_{a \in A} \frac{f_a}{u_a} )</td>
</tr>
<tr>
<td>Minimize sum of average packet delays</td>
<td>( \min \sum_{a \in A} \frac{1}{u_a - f_a} )</td>
</tr>
<tr>
<td>Minimize average network delay</td>
<td>( \min \sum_{a \in A} \frac{f_a}{u_a - f_a} )</td>
</tr>
</tbody>
</table>

**Table 9.1:** Common objective functions for optimizing the system QoS w.r.t. average packet delays.

The first objective function in Table 9.1, i.e. minimizing the maximum link utilization, is considered in e.g. [46, 163]. Some of its advantages are [163] that it is tractable and often implicitly performs well w.r.t. other measures. Some drawbacks [107] are that it is overly sensitive to bottleneck links and do not penalize paths with many hops.

All objective functions but the first in Table 9.1 are non-linear. Since the remaining part of the model is typically a MILP, it is common to approximate the non-linear functions by a piecewise linear function. The most common linearization is probably the Fortz-Thorup objective function \( \Phi \) introduced in [108].
Feasible Routing Pattern Polytopes

To a great extent, the purpose of the analysis in Part II was to be able to describe routing patterns that obey shortest path routing (SPR). In this chapter, we translate some results from Part II into integer linear descriptions of the polytopes that describe feasible routing patterns. These polytopes are considered further in the coming chapters.

Outline The sets related to feasible SPR patterns to be considered in this chapter are introduced in Section 10.1. Integer linear formulations of the associated polytopes are given in Section 10.2. Some of these formulations are refined in Section 10.3. Finally, in Section 10.4, the independence/transitive system view via conflict hypergraphs is introduced.

10.1 The Set of Feasible Routing Patterns

As usual, $G = (N, A)$ is a simple, strongly connected digraph and $K \subset N \times N$ a set of OD-pairs. For a node, $l \in N$, the set of origins of the OD-pairs with destination $l$ is denoted by $S^l := \{i \in N \mid k := (i, l) \in K\}$.

A feasible routing pattern corresponds to a collection of SP-graphs that is simultaneously realizable by a set of arc costs, $w \in \mathbb{Q}_+^A$. We are interested in the convex hull of the incidence vectors of feasible routing patterns. To this end we use binary shortest path indicator variables, $y^l_a$, for each destination, $l \in L$, and each arc $a \in A$, i.e.

$$y^l_a = \begin{cases} 1, & \text{if arc } a \text{ is on a shortest simple path to destination node } l, \\ 0, & \text{otherwise}. \end{cases} \quad (10.1)$$

Observe that all shortest paths are simple since the weights are strictly positive.

To characterize feasible routing patterns in an intuitive (but implicit) way we introduce the map $Y^l(w) : \mathbb{Q}_+^A \rightarrow \mathbb{R}^A$, that takes a vector of link weights to an incidence vector of a digraph as follows,
\[ y^*_a = Y^l(w)_a = \begin{cases} 
1, & \text{if arc } a \text{ is in some shortest path (w.r.t. } w) \text{ from an origin node in } S^l \text{ to the destination node } l, \\
0, & \text{otherwise}. 
\end{cases} \tag{10.2} \]

In words, \( Y^l(w) \) corresponds to a digraph that is the union of all shortest path arborescences w.r.t. \( w \) that span the nodes in \( S^l \) and have \( l \) as their root. The digraph is acyclic since \( w > 0 \). Hence, \( Y^l(w) \) induces an acyclic ingraph rooted at \( l \).

The set of all incidence vectors corresponding to ingraphs rooted at node \( l \) is denoted by \( I^l \subset \mathbb{B}^A \). Using \( Y^l(w) \), the set \( I^l \) is obtained as its image, i.e.

\[ I^l = \{ y^l \in \mathbb{B}^A \mid \text{there exist } w \in \mathbb{Q}_+^A \text{ such that } y^l = Y^l(w) \} \tag{10.3} \]

Observe that this definition allows multiple shortest paths from nodes in \( S^l \) to \( l \). In the case of unique shortest paths, that we refer to as USPR, all ingraphs must be arborescences. We denote the set of all incidence vectors corresponding to an arborescence rooted at \( l \) by \( I^l \). It is obtained from \( I^l \) by restricting the outdegree of all nodes, i.e.

\[ I^l = \left\{ y^l \in \mathbb{B}^A \mid \sum_{a \in \delta^+(i)} y^l_a \leq 1, \; i \in N \right\} \tag{10.4} \]

The collection of incidence vectors for a set of ingraphs, one to each destination in \( L \), is defined as

\[ I^L = \left\{ y = (y^l)_{l \in L} \mid y^l \in I^l \text{ for all } l \in L \right\} \tag{10.5} \]

The unique counterpart, i.e. for arborescences, is defined analogously,

\[ \mathcal{I}^L = \left\{ y = (y^l)_{l \in L} \mid y^l \in I^l \text{ for all } l \in L \right\} \tag{10.6} \]

To describe the set of feasible routing patterns it is not sufficient to model ingraphs (or arborescences). It is required that the ingraphs are simultaneously realizable as (unique) shortest paths, i.e. that all ingraphs are obtained from the same arc costs. Denote the subset of incidence vectors corresponding to feasible routing patterns by \( \mathcal{Y}^L \subset \mathcal{I}^L \). In a similar manner as above, this yields

\[ \mathcal{Y}^L = \left\{ y \in \mathcal{I}^L \mid \text{there exists } w \in \mathbb{Q}_+^A \text{ such that } y^l = Y^l(w) \text{ for all } l \in L \right\} \tag{10.7} \]

and analogously for the unique counterpart,

\[ \overline{\mathcal{Y}}^L = \left\{ y \in \overline{\mathcal{I}}^L \mid \text{there exists } w \in \mathbb{Q}_+^A \text{ such that } y^l = Y^l(w) \text{ for all } l \in L \right\} \tag{10.8} \]

Hereafter, we omit the \( L \) superscript on these sets. We are interested in the convex hulls of the incident vectors of \( \mathcal{Y} \) and \( \overline{\mathcal{Y}} \). Explicit integer programming descriptions of these polytopes are given in the next section.
10.2 Polytopes Associated with Routing Patterns

We seek integer formulations for the incidence vectors of the elements in $\mathcal{Y}$ and $\overline{\mathcal{Y}}$. To obtain this, we first consider the collections of ingraphs and arborescences, $\mathcal{I}$ and $\overline{\mathcal{I}}$, and also the collections of partially realizable SP-graphs, $\mathcal{P}$ and $\overline{\mathcal{P}}$. The associated polytopes are denoted by $\text{conv} \mathcal{Y}$, $\text{conv} \overline{\mathcal{Y}}$, $\text{conv} \mathcal{I}$, $\text{conv} \overline{\mathcal{I}}$, $\text{conv} \mathcal{P}$ and $\text{conv} \overline{\mathcal{P}}$, respectively.

The relation between the above mentioned sets, and hence polytopes, is depicted in Figure 10.1. It is described by a set-inclusion semi-lattice where the join operator is defined by taking the intersection of two sets. For example, $B^{A \times L} \supset \mathcal{P} \supset \overline{\mathcal{P}} \supset \overline{\mathcal{Y}}$. Therefore a valid inequality for $\text{conv} \mathcal{P}$ is valid also for $\text{conv} \overline{\mathcal{P}}$, $\text{conv} \mathcal{Y}$ and $\text{conv} \overline{\mathcal{Y}}$. Hence, below we only say that it is valid for $\text{conv} \mathcal{P}$ since that suffices.

![Figure 10.1](image)

**Figure 10.1:** The semi-lattice representing the relations between the sets considered in this chapter. Lines represent set-inclusion, e.g. $\mathcal{Y} \supset \overline{\mathcal{Y}}$.

At some times we will also consider the collections of partially compatible SP-graphs, $\mathcal{P}^C \supset \mathcal{P}$ and $\overline{\mathcal{P}}^C \supset \overline{\mathcal{P}}$. However, since the resulting inequalities are weaker and $\mathcal{P}^C$ and $\overline{\mathcal{P}}^C$ possess no particular advantages over $\mathcal{P}$ and $\overline{\mathcal{P}}$ the latter SP-graph collections will receive more attention.

10.2.1 Steiner Ingraph and Arborescence Polytopes

The set $\overline{\mathcal{I}}$ contains all Steiner arborescences rooted at $l$ that span the nodes in $S^l$. There are two common formulations for this set in the literature, see e.g. [118, 120, 221], a cut formulation and a flow based extended formulation. In both formulations, $y_{al}^k$ is the binary design variable from (10.1). In the latter, the variable $x_{al}^k$ denotes the flow on arc $a \in A$ from $k \in S^l$ to $l$. The extended formulation is,

$$
\begin{align*}
\sum_{a \in \delta^-(i)} x_{al}^k - \sum_{a \in \delta^+(i)} x_{al}^k &= b_i^k, & i \in N, \ k \in S^l, \ (k, l) \in K, \\
& x_{al}^k \leq y_{al}^l, & a \in A, \ k \in S^l, \ (k, l) \in K, \\
& x_{al}^k \geq 0, & a \in A, \ k \in S^l, \ (k, l) \in K, \\
& y_{al}^l \in B, & a \in A,
\end{align*}
$$

(10.9a)

(10.9b)

(10.9c)

(10.9d)

where as usual $b_i^k$ is the node balance for origin $k$ and destination $l$, see e.g. (9.5) on page 131. Projecting out the flow variables in (10.9) gives the cut formulation.
\[ \sum_{a \in \delta^+(S)} y_a^l \geq 1, \quad S \cap S^l \neq \emptyset, \quad S \subseteq N \setminus \{l\}, \]  
(10.10a)

\[ y_a^l \in \mathbb{B}, \quad a \in A. \]  
(10.10b)

Note that models (10.9) and (10.10) are actually not completely accurate to model $\mathcal{I}$. Indeed, they rather model the dominant of $\mathcal{I}$. This is not a problem when the objective is non-negative since the minimum will be obtained by a solution in $\mathcal{I}$ anyway. In our setting, the objective can have negative components, e.g. in a branch-and-cut-and-price (B&C&P) framework the dual prices of the constraints generated in the master problem may induce negative coefficients in the Steiner arborescence subproblem.

To be accurate we propose a minor modification in the outdegree constraints. We also use an alternative modelling approach based on cycles instead of cuts. This results in,

\[ \sum_{a \in \delta^+(i)} y_a^i = 1, \quad i \in S^l, \]  
(10.11a)

\[ \sum_{a \in \delta^+(i)} y_a^i \leq \sum_{a \in \delta^-(i)} y_a^i, \quad i \notin S^l \setminus \{l\}, \]  
(10.11b)

\[ \sum_{a=(i,j) : i,j \in S} y_a^i \leq |S| - 1, \quad |S| \geq 2, \quad S \subseteq N \setminus \{l\}, \]  
(10.11c)

\[ y_a^i = 0, \quad a \in \delta^+(l), \]  
(10.11d)

\[ y_a^i \in \mathbb{B}, \quad a \in A. \]  
(10.11e)

In (10.11), the first two constraints make sure that a correct number of arcs leaves each node. To model $\mathcal{I}$ using (10.9) or (10.10), these constraints should be augmented. The third constraints in (10.11) are (lifted) cycle prohibition constraints that replace the cut constraints in (10.10) ensuring flow from nodes in $S^l$ to $l$.

Optimizing over $\mathcal{I}$ is NP-hard since it subsumes the Steiner arborescence problem. In the important special case where $S^l = N \setminus \{l\}$, the problem is easy and all formulations above are equivalent and, in particular, they are integral.

Which models to use is not obvious. Models (10.9) and (10.10) are equivalent (see e.g. [221] and [205, Chapter 52]) but incomparable to (10.11). In particular, for objective functions with some negative coefficients there may be an optimal solution to the LP relaxation of (10.10) that is not in the LP relaxation of (10.11).

From a practical point of view the model choice is not clear either. Model (10.9) is polynomial in size but soon becomes quite large while the separation problems for (10.10) and (10.11) can be solved as easy max flow problems. As an example, the computational experiments in [24] shows that for medium and large networks (50-100 nodes) and not to many OD-pairs (< 1000), model (10.9) performs well, while it becomes to large and performs poor (or runs out of memory) when there are many OD-pairs (> 5000). When the flow variables are already in the problem, model (10.9) is the natural choice.

Allowing multiple shortest paths induces the set $\mathcal{I}$ instead of $\mathcal{I}$. It contains all acyclic Steiner digraphs rooted at $l$ that span the nodes in $S^l$. To derive an integer formulation
for \( \mathcal{I} \), model (10.11) is a suitable starting point. There are two differences compared to \( \mathcal{I} \): the outdegrees can be larger than 1 and the cycle prohibition inequalities can no longer be lifted (they are already facets, see Section 12.4.1). This yields,

\[
\sum_{a \in \delta^+(i)} y^l_a \geq 1, \quad i \in S^l, \tag{10.12a}
\]
\[
y^l_a \leq \sum_{a \in \delta^-(i)} y^l_a, \quad \bar{a} \in \delta^+(i), \quad i \not\in S^l \setminus \{l\}, \tag{10.12b}
\]
\[
\sum_{a \in C} y^l_a \leq |C| - 1, \quad C \in \mathcal{D}^l, \tag{10.12c}
\]
\[
y^l_a = 0, \quad a \in \delta^+(l), \tag{10.12d}
\]
\[
y^l_a \in \mathbb{B}, \quad a \in A, \tag{10.12e}
\]

where \( \mathcal{D}^l \) is the collection of all directed cycles in \( G \) not containing node \( l \).

To model the set of feasible routing patterns, \( \mathcal{Y} \) and \( \overline{\mathcal{Y}} \), using \( \mathcal{I} \) and \( \mathcal{I} \), we must guarantee that there is a some arc cost vector, \( w \), that simultaneously gives all ingraphs, or arborescences, cf. (10.7) and (10.8). This is accomplished by prohibiting the routing conflicts presented in Part II of this thesis.

### 10.2.2 Compatible and Partially Realizable SP-Graph Polytopes

Recall that compatibility (and hence, partial realizability) is a necessary condition for realizability. In Proposition 5.2 on page 64 it was shown that it is also sufficient when each SP-graph spans its associated origin nodes. This implies that the set of feasible routing patterns can be described as the intersection of Steiner arborescences/graphs and partially compatible, or partially realizable, SP-graphs, i.e.

\[
\mathcal{Y} = \mathcal{I} \cap \mathcal{P} = \mathcal{I} \cap \mathcal{P}^C \quad \text{and} \quad \overline{\mathcal{Y}} = \mathcal{I} \cap \overline{\mathcal{P}} = \mathcal{I} \cap \overline{\mathcal{P}}^C. \tag{10.13}
\]

The collections of partially realizable SP-graphs, \( \mathcal{P} \) and \( \overline{\mathcal{P}} \), are modelled below. Take a vector \( y \in [0, 1]^{A \times L} \) and let \( A_0(y) \) and \( A_1(y) \) be the index sets of arc destination pairs where \( y \) is 0 and 1, respectively, i.e.

\[
A_0(y) = \{(a, l) \in A \times L \mid y^l_a = 0\} \tag{10.14}
\]

and

\[
A_1(y) = \{(a, l) \in A \times L \mid y^l_a = 1\}. \tag{10.15}
\]

Denote the part of \( A_0(y) \) and \( A_1(y) \), respectively, associated with destination \( l \) by \( A'_0(y) \) and \( A'_1(y) \), i.e.

\[
A'_0(y) = \{a \in A \mid a \in A_0(y)\} \quad \text{and} \quad A'_1(y) = \{a \in A \mid a \in A_1(y)\}. \tag{10.16}
\]

Also, define the set of destination arcs, \( \mathcal{D}^l(y) \), for destination \( l \) in the natural manner,
\[ \mathcal{D}^l(y) = \{(i, l) : \delta^+(i) \cap A^1_l(y) = \emptyset\}, \tag{10.17} \]
and let
\[ \mathcal{D}(y) = \bigcup_{l \in L} \mathcal{D}^l(y) \quad \text{and} \quad \tilde{A}(y) = A \cup \mathcal{D}(y). \tag{10.18} \]

These arcs are referred to as D-arcs. Using model (5.13) on page 63 in Part II yields the partial realizability instance induced by \( y \),

\[
\begin{align*}
&\quad w_{ij} + \pi^i_l - \pi^j_l = 0, \quad (i, j) \in A^1_l(y) \cup \mathcal{D}^l(y), \ l \in L, \tag{10.19a} \\
&\quad w_{ij} + \pi^i_l - \pi^j_l \geq 1, \quad (i, j) \in A^0_l(y), \ l \in L, \tag{10.19b} \\
&\quad w_{ij} + \pi^i_l - \pi^j_l \geq 0, \quad (i, j) \in \tilde{A}(y), \ l \in L, \tag{10.19c} \\
&\quad w_{ij} \geq 1, \quad (i, j) \in \tilde{A}(y). \tag{10.19d}
\end{align*}
\]

This gives an implicit definition of \( \mathcal{P} \) as

\[ \mathcal{P} = \{(A_0(y), A_1(y)) \subset (A \times L)^2 \mid (10.19) \text{ is feasible for } A_0(y) \text{ and } A_1(y)\} \tag{10.20} \]

To define the unique counterpart, \( \mathcal{P}^\circ \), it suffices to restrict all outdegrees, i.e.

\[ \mathcal{P}^\circ = \{(A_0(y), A_1(y)) \in \mathcal{P} \mid |A^1_k(y) \cap \delta^+(i)| \leq 1 \text{ for all } i \in N, l \in L\} \tag{10.21} \]

The analogues for \( \mathcal{P}^C \) and \( \mathcal{P}^\circ C \) are obtained by using \( \tilde{D}^l = \emptyset \).

Remark 10.1. Observe that it is not appropriate to define a set similar to \( \mathcal{P} \) in terms of the values of the \( y \) variables, i.e. as \( \{y \in [0, 1]^{A \times L} \mid (10.19) \text{ is feasible for } A_0(y) \text{ and } A_1(y)\} \).

Indeed, this set is open and its closure is \([0, 1]^{A \times L}\).

Integer linear descriptions of \( \mathcal{P} \) and \( \mathcal{P}^\circ C \) are obtained from the implicit descriptions by enumerating the (irreducible) infeasible subsystems of (10.19) and prohibiting them. Let \((Q_0, Q_1) \subseteq (A \times L)^2\) be a pair of arc destination subsets such that (10.19) is infeasible when \(A_0 := Q_0, A_1 := Q_1\) and \( \tilde{D} \) is obtained from \( A_1 \). Then, the inequality,

\[ \sum_{(a, l) \in Q_0} y^l_a + \sum_{(a, l) \in Q_1} (1 - y^l_a) \geq 1, \tag{10.22} \]

is valid for \( \text{conv} \mathcal{P} \). Further, enumerating all (irreducible) infeasible subsystems suffices. Denote the collection of all arc destination subset pairs that corresponds to an infeasible subsystem of (10.19) by \( \mathcal{Q} \). Then, the set of partially realizable SP-graphs, \( \mathcal{P} \), corresponds to the feasible solutions to the following integer linear system,

\[
\begin{align*}
\sum_{(a, l) \in Q_0} y^l_a + \sum_{(a, l) \in Q_1} (1 - y^l_a) \geq 1, \quad (Q_0, Q_1) \in \mathcal{Q}, \tag{10.23a} \\
y^l_a \in \mathbb{B}, \quad a \in A, \ l \in L. \tag{10.23b}
\end{align*}
\]
Using (10.23) and (10.21) it is trivial to obtain an integer linear system for $\mathcal{P}$:

$$
\sum_{(a,l) \in Q_0} y^l_a + \sum_{(a,l) \in Q_1} (1 - y^l_a) \geq 1, \quad (Q_0, Q_1) \in \mathcal{Q}, \quad (10.24a)
$$

$$
\sum_{a \in \delta^+(i)} y^l_a \leq 1, \quad i \in N, \ l \in L, \quad (10.24b)
$$

$$
y^l_a \in \mathbb{B}, \quad a \in A, \ l \in L. \quad (10.24c)
$$

Similar results for the compatibility case, i.e. $\mathcal{P}_C$ and $\mathcal{P}^{C}_C$, are given in e.g. [46, 50, 218].

### 10.2.3 Realizable SP-Graph Polytopes

We do not write out the trivial integer linear descriptions of $\mathcal{Y}$ and $\overline{\mathcal{Y}}$ induced by (10.13) and the integer linear descriptions from the previous subsections. Instead, we give a formulation of $\mathcal{Y}$ similar to the formulation of $\mathcal{P}$ in (10.23) based on realizability.

Let $(Q^R_0, Q^R_1) \subseteq (A \times L)^2$ be a pair of arc destination subsets whose induced SP-graphs are not realizable, i.e. model (4.4) on page 46 is infeasible for $y$. Then,

$$
\sum_{(a,l) \in Q^R_0} y^l_a + \sum_{(a,l) \in Q^R_1} (1 - y^l_a) \geq 1, \quad (Q^R_0, Q^R_1) \in \mathcal{Q}^R, \quad (10.25)
$$

is valid for $\text{conv} \ \mathcal{Y}$. Observe that (10.25) is in general not valid for $\text{conv} \ \mathcal{P}$.

**Remark 10.2.** Consider a "routing conflict" caused by the size of the weights, i.e. $w_{\text{MAX}}$ is not large enough in model (3.7) on page 36. Such conflicts can be handled by (10.25). Indeed, this constraint can be used to prohibit an arbitrary face (including a vertex) where some variables are fixed at 0 or 1.

Further, denote by $Q^R$ the collection of all inclusion-wise minimal arc destination subset pairs $(Q^R_0, Q^R_1) \subseteq (A \times L)^2$ that correspond to SP-graphs that are not realizable. Then, $Q^R$ is sufficient to describe $\mathcal{Y}$ as

$$
\sum_{(a,l) \in Q^R_0} y^l_a + \sum_{(a,l) \in Q^R_1} (1 - y^l_a) \geq 1, \quad (Q^R_0, Q^R_1) \in Q^R, \quad (10.26a)
$$

$$
y^l_a \in \mathbb{B}, \quad a \in A, \ l \in L. \quad (10.26b)
$$

This follows from (10.13). Indeed, the inequalities in the descriptions of $\mathcal{I}$ and $\mathcal{P}$ are subsumed by the inequalities in (10.26). Further, this implies that the LP-relaxation of (10.26) is a stronger formulation of $\mathcal{Y}$ then the one induced by (10.13), (10.11) and (10.23). The unique counterpart analogues are omitted.

The drawback of model (10.26) is that it is very unlikely (due to the complexity of realizability) that there exists a good characterization of $Q^R$, i.e. the pairs of subsets $(Q^R_0, Q^R_1)$ that make (10.25) a valid inequality for $\text{conv} \ \mathcal{Y}$. It is also hard to separate such inequalities.
Because of this complication, we limit the scope here to routing conflicts that arise from partial unrealizability. This implies that the combinatorial characterization of infeasible routing patterns in Chapter 6 can be used. This yields a comprehensive and combinatorial description of the pairs \((Q_0, Q_1) \in Q\) where (10.22) constitute a valid inequality for \(\text{conv } P\), and thus also for \(\text{conv } \mathcal{Y}\). Note that this important subfamily of \(Q_R\) is not complete. However, it is sufficient as long as the intersection with the set \(I\) is considered since realizability and partial realizability is equivalent when all ingraphs span the required set of nodes.

Next, we refine the descriptions of \(\text{conv } P\) and \(\text{conv } \mathcal{Y}\) and their unique counterparts by using a combinatorial characterization of valid inequalities based on routing conflicts.

### 10.3 A Characterization of Valid Inequalities

By translating the results in Chapter 6 we obtain an explicit integer linear description of \(P\) and thus also of \(P, \mathcal{Y}\) and \(\mathcal{Y}\). We use the notation and terminology from Chapter 6 and, in particular \(Q\) is the collection of all arc destination subset pairs that correspond to an infeasible subsystem of (10.19).

A pair \((Q_0, Q_1) \in Q\) corresponds 1-to-1 to a feasible family of cycles \(C\), where

\[
C = \{C^i\}_{i \in L},
\]

and

\[
C^i = \left\{ C_k^i \subset \tilde{A} \mid C_k^i = F_k^i \cup B_k^i, B_k^i \subseteq A^i \cup \tilde{D}^i \right\}.
\]

To express valid inequalities concisely, we collect the indices of the forward and backward arcs associated with a feasible cycle family \(C\). Let

\[
\tilde{F}(C) = \left\{ (a, l) \in \tilde{A} \times L \mid a \in F_k^i \text{ for some } k \right\},
\]

and

\[
\tilde{B}(C) = \left\{ (a, l) \in \tilde{A} \times L \mid a \in B_k^i \text{ for some } k \right\}.
\]

Also define the arc sets associated with a certain destination,

\[
\tilde{F}^d(C) = \left\{ a \in \tilde{A} \mid (a, l) \in \tilde{F}(C) \right\},
\]

and

\[
\tilde{B}^d(C) = \left\{ a \in \tilde{A} \mid (a, l) \in \tilde{B}(C) \right\}.
\]

Note that a cycle in \(C\) may contain D-arcs and also that there are no design variables associated with them. The following notation is used when the restriction to ordinary arcs is considered. Define

\[
B(C) = \tilde{B}(C) \cap (A \times L) \quad \text{and} \quad F(C) = \tilde{F}(C) \cap (A \times L),
\]

(10.33)
and also let
\[ B^l(C) = \tilde{B}^l(C) \cap A \quad \text{and} \quad F^l(C) = \tilde{F}^l(C) \cap A. \] (10.34)

Equations (10.33) and (10.34) can be seen as a projection of the feasible cycle families onto the underlying set of arc destination pairs. This projection may significantly improve the quality of affected valid inequalities. It also causes some technical complications. See e.g. Example 5.3 on page 66 and Section 10.3.2 and 10.4.2, and the examples therein.

Remark 10.3. When the D-arcs are omitted, the collections of arc destination pairs, \( B(C) \) and \( F(C) \), no longer induce cycles. This implies that the structure is not as clearly revealed as in Chapter 6 before the projection. In the compatibility case, there are no D-arcs. Hence, each cycle family \( C \) implies that the collections of arc destination pairs, \( B(C) \) and \( F(C) \), induce cycles.

### 10.3.1 Valid Inequalities from Cycle Families

We derive the valid inequalities arising from cycle families. Let \( y \in \mathbb{B}^{A \times L} \) be a binary vector of shortest path indicator variables and \( C \) cycle family. If all indices in \( y \) associated with the backward arcs in \( C \) are 1, then there is a potential routing conflict induced by \( y \). Whether the point \( y \) belongs to \( \text{conv} \mathcal{P} \) depends on if the cycle family is improving or not. When it is improving, all design variables associated with the backward arcs in \( C \) must simultaneously be 1.

**Proposition 10.1**

Let \( C \) be an improving family of cycles. Then, the inequality
\[ \sum_{l \in L} \sum_{a \in B^l(C)} (1 - y^l_a) \geq 1 \] (10.35)
is valid for \( \text{conv} \mathcal{P} \).

Proposition 6.1 on page 75 implies that a non-saturating cycle family is improving.

**Corollary 10.1**

Let \( C \) be a non-saturating family of cycles. Then, inequality (10.35) is valid for \( \text{conv} \mathcal{P} \).

In the saturating case, Proposition 6.2 on page 75 states that a saturating solution to (6.2), say \( \theta^l \), is improving if and only if there is a commodity \( l \) and an arc \( a \) such that \( a \in A^l_0(y) \) and \( \theta^l_a > 0 \). This means that no such arc must be in \( A^l_0(y) \) and therefore we must force them to be in \( A^l_1(y) \). This can be put in terms of cycle families and design variables; if \( C \) is saturating and all backward arc design variables are 1, then all forward arc design variables must also be 1.

**Proposition 10.2**

Let \( C \) be a saturating family of cycles. Then, the inequalities
\[ y^l_{a'} + \sum_{l \in L} \sum_{a \in B^l(C)} (1 - y^l_a) \geq 1, \quad (a', l') \in F(C), \] (10.36)
are valid for \( \text{conv} \mathcal{P} \).
Proposition 10.1 and 10.2 imply that a necessary pair \((Q_0, Q_1) \in \mathcal{Q}\) always satisfies \(|Q_0| \leq 1\). In terms of the general SPR inequality (10.22), this means that all terms but at most 1 have the same sign.

To obtain an integer linear inequality description of \(\mathcal{P}\) it is sufficient to enumerate all feasible families of cycles and include the valid inequalities from Proposition 10.1 and 10.2. Let \(Q_S\) and \(Q_{NS}\) be the collection of feasible saturating and non-saturating cycles families, respectively.

**Proposition 10.3**

The incidence vector \(\mathbf{y} \in \mathbb{B}^{A \times L}\) corresponds to an element in \(\mathcal{P}\) if and only if it is a feasible solution to the system

\[
\sum_{a \in B(\mathcal{C})} (1 - y_a) \geq 1, \quad \mathcal{C} \in Q_{NS}, \ l \in L, \quad (10.37a)
\]

\[
y_{a'} + \sum_{l \in L} \sum_{a \in B(\mathcal{C})} (1 - y_a) \geq 1, \quad (a', l') \in F(\mathcal{C}), \ \mathcal{C} \in Q_S. \quad (10.37b)
\]

To formulate \(\mathcal{P}\) in terms of \(\mathcal{P}\) it suffices to add constraints that restrict outdegrees to be at most 1, i.e. to augment model (10.37) by the constraints

\[
\sum_{a \in \delta^+(i)} y_a \leq 1, \quad i \in N, \ l \in L. \quad (10.38)
\]

Similar results for the compatibility case, i.e. \(\mathcal{P}^C\) and \(\mathcal{P}^C\), are given in e.g. [46, 50, 218].

### 10.3.2 Non-Dominated Valid Inequalities

To refine the descriptions of \(\mathcal{P}^C\) and \(\mathcal{P}^C\) further, we consider which of the inequalities in (10.37) that are non-dominated. As indicated earlier, it suffices to consider inequalities associated with irreducible cycle families.

**Proposition 10.4**

Let \(\mathcal{C}\) be a family of cycles and (10.36) an induced valid inequality. If \(\mathcal{C}\) is associated with a reducible solution, then all inequalities induced by \(\mathcal{C}\) are dominated.

**Proof:** When \(\mathcal{C}\) is reducible, Definition 6.10 on page 83 implies that there exists a family of cycles \(\overline{\mathcal{C}}\), such that \(B(\overline{\mathcal{C}}) \subset B(\mathcal{C})\). An inequality associated with \(B(\mathcal{C})\) is dominated by an inequality associated with \(B(\overline{\mathcal{C}})\). Indeed, the former inequality is the sum of the latter inequality and the upper bound constraints for the variables in \(B(\overline{\mathcal{C}}) \setminus B(\mathcal{C})\). \(\square\)

For non-saturating cycle families, Theorem 8.2 and 8.3 in Chapter 8 imply that there is an improving saturating cycle family. This gives a stronger dominance result similar to Proposition 10.4 for non-saturating cycle families. The consequence is that all necessary inequalities associated with non-saturating solutions arise from directed cycles.

**Proposition 10.5**

Let \(\mathcal{C}\) be a non-saturating family of cycles. Then, (10.35) is a valid inequality for \(\text{conv} \ \mathcal{P}\) that is non-dominated only if \(\mathcal{C}\) corresponds to a simple directed cycle.
Proof: We show that the inequality induced by \( \mathcal{C} \) is dominated unless it is associated with a directed cycle. Theorem 8.2 on page 121 states that a non-saturating solution induce a saturating solution. By Theorem 8.3, this solution is improving if there for every cycle, \( \mathcal{C} \), is a destination \( l \) where some arc, \( a \in \mathcal{C} \), is a non-SP-arc. This condition is satisfied for all \( l \in L \) by all points in \( \mathcal{P} \) due to the cycle inequality

\[
\sum_{l \in L} \sum_{a \in \mathcal{C}} (1 - y_a^l) \geq 1. \tag{10.39}
\]

Hence, is suffices to prohibit the saturating solution. Since Theorem 8.3 requires that there are two SP-graphs in the conflict, \( \mathcal{C} \) can only induce a non-dominated inequality if it is a directed cycle.

Let \( \mathcal{Q}_I \) be the collection of irreducible and feasible saturating cycle families and \( \mathcal{Q}_C \), the collection of non-saturating cycle families associated with simple directed cycles not containing node \( l \) (this suffices since there is no arc out from a destination node).

Unfortunately, irreducibility of cycle families is not equivalent to non-domination for the associated valid inequalities. We give an explanation to this (at first) counter intuitive behavior. Take two irreducible cycle families \( \mathcal{C}, \tilde{\mathcal{C}} \in \mathcal{Q}_I \). By irreducibility, \( \mathcal{C} \) and \( \tilde{\mathcal{C}} \) are incomparable when seen as cycle families, i.e.

\[
\tilde{B}(\tilde{\mathcal{C}}) \not\subseteq \tilde{B}(\mathcal{C}) \quad \text{and} \quad \tilde{B}(\mathcal{C}) \not\subseteq \tilde{B}(\tilde{\mathcal{C}}). \tag{10.40}
\]

In the derivation of valid inequalities arising from \( \mathcal{C} \) and \( \tilde{\mathcal{C}} \), the D-arcs are projected out. This may result in one of the cycle families containing the other, so to speak, i.e.

\[
B(\tilde{\mathcal{C}}) \subset B(\mathcal{C}) \quad \text{or} \quad B(\mathcal{C}) \subset B(\tilde{\mathcal{C}}). \tag{10.41}
\]

When this occurs, i.e. if \( B(\tilde{\mathcal{C}}) \subset B(\mathcal{C}) \), then some or all of the inequalities associated with \( \mathcal{C} \) are dominated by inequalities associated with \( \tilde{\mathcal{C}} \). We make this more precise below, first for the USPR case and then for the ECMP case.

Non-Dominated Valid Inequalities in the USPR Case

Formulating \( \overline{\mathcal{P}} \) from \( \mathcal{P} \) by restricting outdegrees to be at most 1 is a very naive approach that does not take advantage of uniqueness. In fact, uniqueness is a crucial aspect in the derivation of strong inequalities.

In the USPR case, almost all cycle families are improving and it turns out that only relatively few cycle families that are required have a D-arc. To see this, and to take advantage of it, we partition the irreducible cycle families as

\[
\mathcal{Q}_I = \mathcal{Q}_I \cup \tilde{\mathcal{Q}}_I = (\mathcal{Q}_U \cup \mathcal{Q}_D) \cup (\tilde{\mathcal{Q}}_S \cup \tilde{\mathcal{Q}}_P \cup \tilde{\mathcal{Q}}_D) \tag{10.42}
\]

where \( \mathcal{Q}_I \) and \( \tilde{\mathcal{Q}}_I \) are the cycle families not containing, and containing respectively, D-arcs. The partitioning of \( \mathcal{Q}_I \) and \( \tilde{\mathcal{Q}}_I \) is as follows.

The cycle families without D-arcs, \( \mathcal{Q}_U \), have been divided into sub-collection depending on whether some cycle associated with \( l \in \mathcal{L} \) induce a path to \( l \) or not. For \( \mathcal{C} \in \mathcal{Q}_I \), let \( N(B^l(\mathcal{C})) \) be the nodes spanned by \( B^l(\mathcal{C}) \). Then, \( \mathcal{C} \in \mathcal{Q}_U \) if \( l \not\in N(B^l(\mathcal{C})) \) for all \( l \in L \) and \( \mathcal{C} \in \mathcal{Q}_D \) if \( l \in N(B^l(\mathcal{C})) \) for some \( l \in L \).
The cycle families with D-arcs, \( \tilde{Q}_I \), have been divided into sub-collections as follows. Define \( \tilde{Q}^l_S \) as the collection of cycle families that correspond to valid cycles induced by two \((i, l)\)-paths, \( P^l_i \) and \( \tilde{P}^l_i \), where \( \tilde{P}^l_i \) consists only of a destination arc, \( \tilde{a} := (i, l) \). Then, collection \( \tilde{Q}_S \) is the union of all \( \tilde{Q}^l_s \). The collection \( \tilde{Q}_P \) consists of cycle families that correspond to valid cycles induced by two \((i, l)\)-paths, \( P^l_i \) and \( \tilde{P}^l_i \), where \( \tilde{P}^l_i \) consists of an arc \( \tilde{a} := (i, j) \) and a destination arc \( \tilde{a} := (j, l) \), i.e. \( \tilde{P}^l_i = \{\tilde{a}, \tilde{a}\} \). Hence, the collection \( \tilde{Q}_D \) must consist of all cycle families with a destination arc that do not fit into one of the above collections, i.e. \( \tilde{Q}_D = \tilde{Q} \setminus (\tilde{Q}_S \cup \tilde{Q}_P) \). We give examples of cycle families in these collections in Example 10.1 and Figure 10.2.

**Figure 10.2:** Two cycle families induced by paths from 1 to 3. The dashed arcs are associated with destination 3 and the dotted arcs are destination arcs. Denote the collection above collections, i.e. \( e \in D \) must consist of all cycle families with a destination arc that do not fit into one of the above collections, i.e. \( \tilde{Q}_D = \tilde{Q} \setminus (\tilde{Q}_S \cup \tilde{Q}_P) \). We give examples of cycle families in these collections in Example 10.1 and Figure 10.2.

For each part in (10.42), we determine which of the inequalities in the general forms (10.35) and (10.36) that are required to describe \( \tilde{P} \), i.e. non-dominated. The first result states that almost all cycle families are improving which implies that the number of required inequalities is significantly reduced.

**Proposition 10.6**

Let \( C \in Q_I \) be a saturating family of cycles. Then,

\[
\sum_{l \in L} \sum_{a \in B^M(C)} (1-y^a_l) \geq 1 \tag{10.43}
\]

is valid for \( \text{conv} \tilde{P} \) if and only if \( C \notin \tilde{Q}_S \).

**Proof:** It suffices to observe that \( \tilde{Q}_S \) corresponds 1-to-1 to non-improving cycle families in the USPR case. Recall from Remark 6.6 on page 76 that all non-improving cycle families are of this form, i.e. they are induced by a path and a destination arc. \( \square \)

To analyze the non-improving cycle families, take \( C \in \tilde{Q}_S \). We write the valid cycle associated with \( C \) as \( C = F \cup B = P^l_i \cup \tilde{P}^l_i \). Let \( \tilde{a}(C) \) be the first arc in \( P^l_i \) and let \( \tilde{l}'(C) \) be the destination different from \( l \). We claim that every inequality in (10.36) is dominated unless it is associated with \( \tilde{a}(C) \). Indeed, for \( a' \neq \tilde{a}(C) \), there is a cycle family \( \tilde{C} \) induced by the subpath of \( \tilde{P}^l_i \) to \( l \) starting with \( a' \). The inequality associated with \( a' \) and \( \tilde{C} \) dominates the inequality associated with \( a' \) and \( C \) since \( B^M(C) \) = \( B^M(C) = \{a\} \) and \( B^M(\tilde{C}) \subset B^M(\tilde{C}) \). See also Example 10.1.

**Proposition 10.7**

Let \( C \in \tilde{Q}_S \) induce the valid cycle \( C = F \cup B = P^l_i \cup \{\tilde{a}\} \). The only non-dominated inequality in (10.36) associated with \( C \) is
it corresponds to a valid cycle 

Remark 10.4. Observe that Proposition 10.7 applies both to the USPR and ECMP cases.

We analyze the improving families of cycles. First take a cycle family without D-arcs. To show that no cycle family in \(Q_D\) is required, we construct an improving cycle family with a D-arc that induces a dominating inequality. This shows the benefit of using partial realizability, i.e. D-arcs, in the sense that several families of cycles that induce compatibility conflicts yield valid inequalities that are dominated by inequalities that stem from partial realizability conflicts.

Proposition 10.8
Take \(C \in Q_D\). In the USPR case, the inequality (10.43) associated with \(C\) is dominated.

Proof: Take \(l \in L\) such that \(l \in N(B(C))\). Then, there is an \((i, l)\)-path \(\tilde{P}_l^i \subseteq B(C)\). If \(C\) does not correspond to a conflict arising from two \((i, l)\)-paths, then construct \(\tilde{C} \in Q_D\) from \(C\) by replacing \(\tilde{P}_l^i\) by the D-arc \(\tilde{a} := (i, l)\). If \(C\) corresponds to a conflict from two \((i, l)\)-paths, \(\tilde{P}_l^i\) and \(\tilde{P}_l^j\) \(\subseteq B(C)\). Then, construct \(\tilde{C} \in Q_P\) from \(C\) by replacing \(\tilde{P}_l^i\) by the first arc, say \(a := (i, j)\), in \(\tilde{P}_l^i\) and a D-arc, \(\tilde{a} := (j, l)\), i.e. by the path \(\{a, \tilde{a}\}\). Since \(B(C) \subseteq B'(C)\) and \(B'(C) = B'(\tilde{C})\) for all \(l \in L \setminus \{l\}\), the result follows.

Remark 10.5. The inequality associated with \(\tilde{C}\) in the proof of Proposition 10.8 is often dominated, i.e. if \(\tilde{C} \in Q_D\). By recursion, or the derivation of Proposition 10.9, it follows that there is a \(\tilde{C}' \in Q_P\) that induces a non-dominated inequality.

We continue to analyze improving cycle families. Take \(C \in Q_I \setminus Q_S\). If \(C \in Q_P\), then it corresponds to a valid cycle \(\tilde{C} = F \cup B = \tilde{P}_l^i \cup \tilde{P}_l^j\), where \(\tilde{P}_l^i = \{a, \tilde{a}\} = \{(i, j), (j, l)\}\). Then, \(\tilde{C}\) induces the valid inequality

\[
(1 - y_{\tilde{a}}^i) + \sum_{\tilde{a} \in B'(C)} (1 - y_{\tilde{a}}^j) \geq 1. 
\]

(10.45)

Take another improving cycle family with a D-arc, \(\tilde{C} \in Q_D\), where all original backward arcs in \(C\) also belong to \(\tilde{C}\), i.e. that \(B'(C) = \{\tilde{a}\} \subseteq B'(\tilde{C})\) and \(B'(C) = \tilde{P}_l^i \subseteq B'(\tilde{C})\). Then, the inequality associated with \(\tilde{C}\) is dominated by the inequality associated with \(\tilde{C}\). See Figure 10.2 and Example 10.1.

Proposition 10.9
Take \(C \in Q_D\). In the USPR case, the inequality (10.43) associated with \(C\) is dominated.

Summarizing, the partitioning in (10.42) and Proposition 10.6-10.9 yield a significant improvement of the formulation of \(\overline{P}\) over the trivial adaption of \(P\) in Proposition 10.3. We give the resulting formulation of \(\overline{P}\) as a theorem.

Theorem 10.1
The incidence vector \(y \in B^{A \times L}\) corresponds to an element in \(\overline{P}\) if and only if it is a feasible solution to the system
\[ \sum_{a \in \delta^+(i)} y_a^l \leq 1, \quad i \in N, \quad l \in L, \]  
(10.46a)

\[ \sum_{a \in B^l(C)} (1 - y_a^l) \geq 1, \quad C \in \mathcal{Q}_C^l, \quad l \in L, \]  
(10.46b)

\[ \sum_{l \in L} \sum_{a \in B^l(C)} (1 - y_a^l) \geq 1, \quad C \in \mathcal{Q}_U \cup \mathcal{Q}_P, \]  
(10.46c)

\[ y_a^l + \sum_{a \in B^l'(C)} (1 - y_a'^l) \geq 1, \quad \bar{a} := a(C), \quad l' := l'(C), \quad C \in \mathcal{Q}_S. \]  
(10.46d)

The compatibility analogues, \(\mathcal{P}_C^C\) and \(\mathcal{P}_P^C\), can easily be described as a byproduct of the partitioning (10.42) of \(\mathcal{Q}_J\). Indeed, it suffices to omit cycle families involving D-arcs.

**Remark 10.6.** Note that the formulation of \(\mathcal{P}_C^C\) induced by the partitioning in (10.42) is weaker than the relaxation of (10.46) where constraint (10.46d) is dropped. Indeed, it suffices to show that the inequalities induced by \(\mathcal{Q}_D\) are satisfied if the inequalities induced by \(\mathcal{Q}_P\) are satisfied. This is Remark 10.5.

**Example 10.1**

Recall Figure 10.2. Let the destinations be 3 and \(l \not\in \{1, 2, 3, 4, 5\}\) and consider the cycle families based on the following valid cycles,

\[ C_{P1} = F \cup \bar{B} = \{(2, 3)\} \cup \{(2, 3)\}, \]  
(10.47a)

\[ C_{P2} = F \cup \bar{B} = \{(1, 2), (2, 3)\} \cup \{(1, 3)\}, \]  
(10.47b)

\[ C_L = F \cup \bar{B} = \{(1, 2), (2, 3)\} \cup \{(1, 4), (4, 3)\}, \]  
(10.47c)

\[ C_R = F \cup \bar{B} = \{(1, 2), (2, 3)\} \cup \{(1, 4), (4, 5), (5, 3)\}. \]  
(10.47d)

The cycle families on the left and right, respectively, in Figure 10.2 are induced by the valid cycles \(C_L\) and \(C_R\). In addition, the cycle families associated with the paths 2-3 and 1-2-3, respectively, are induced by the valid cycles \(C_{P1}\) and \(C_{P2}\). We have emphasized that \(B(C)\) originally contains a D-arc (to destination 3) by writing \(\bar{B}\). The inequalities induced by the cycle families in (10.47) are,

\[ y_{23}^l \leq 0 + y_{23}^3, \]  
(10.48a)

\[ y_{12}^l + y_{23}^l \leq 1 + y_{23}^3, \quad \text{dominated by (a)}, \]  
(10.48b)

\[ y_{12}^l + y_{23}^l \leq 1 + y_{12}^3, \]  
(10.48c)

\[ y_{12}^l + y_{23}^l + y_{14}^3 \leq 2, \]  
(10.48d)

\[ y_{12}^l + y_{23}^l + y_{14}^3 + y_{45}^3 \leq 3, \quad \text{dominated by (d)}. \]  
(10.48e)

Observe that \(C_{P1}\) and \(C_{P2}\) induce non-improving cycle families in \(\mathcal{Q}_S^l \subset \mathcal{Q}_S\) which yield inequalities of form (10.36). In accordance with Proposition 10.7, inequality (10.48b)
is dominated. The cycle families associated with \( C_L \) and \( C_R \) belong to \( \mathcal{Q}_P \) and \( \mathcal{Q}_D \), respectively. They are improving, see Proposition 10.6 and of form (10.43). In accordance with Proposition 10.8, the inequality (10.48e) associated with the latter cycle family is dominated (by the inequality associated with former).

### Non-Dominated Valid Inequalities in the ECMP Case

From the general analysis in Section 10.3.2, Proposition 10.5 implies that the only non-dominated inequalities arising from non-saturating cycle families correspond to directed cycles. From the analysis of the USPR case, Proposition 10.7 applies also to the ECMP case. Hence, the non-dominated inequalities associated with paths, i.e. where \( \mathcal{C} \in \mathcal{Q}_S \), have also been characterized. The analysis of the domination aspect for the other parts in the partitioning (10.42) differs from the USPR case.

Take an irreducible cycle family \( \mathcal{C} \in \mathcal{Q}_I \). The inequalities (10.36) in Proposition 10.2 associated with \( \mathcal{C} \) are repeated here for convenience,

\[
y''_{a;l'} + \sum_{l \in L} \sum_{a \in B'(\mathcal{C})} (1 - y''_a) \geq 1, \quad (a', l') \in F(\mathcal{C}) .
\]  \hspace{1cm} (10.49)

Given an arc destination pair \((a, l) \in F(\mathcal{C})\), we define the set \( \mathcal{D}'_{a}(\mathcal{C}) \) of cycle families that induce some inequality of form (10.49) that dominates the inequality associated with \((a, l)\) in (10.49) as

\[
\mathcal{D}'_{a}(\mathcal{C}) := \{ \mathcal{C}' \in \mathcal{Q}_I | (a, l) \in F(\mathcal{C}') \text{ and } B(\mathcal{C}') \subset B(\mathcal{C}) \} .
\]  \hspace{1cm} (10.50)

The interesting inequalities in (10.49) are associated with arc destination pairs where \( \mathcal{D}'_{a}(\mathcal{C}) = \emptyset \) since they are exactly the non-dominated inequalities. We define

\[
\mathcal{N}(\mathcal{C}) := \{ (a, l) \in A \times L | \mathcal{D}'_{a}(\mathcal{C}) = \emptyset \} .
\]  \hspace{1cm} (10.51)

#### Proposition 10.10

Take \( \mathcal{C} \in \mathcal{Q}_I \) and \((a, l) \in F(\mathcal{C})\). There exist a \( \mathcal{C}' \in \mathcal{Q}_I \) such that the inequality in (10.49) associated with \( \mathcal{C} \) is dominated by the inequality in (10.49) associated with \( \mathcal{C}' \) if and only if \( \mathcal{D}'_{a}(\mathcal{C}) \neq \emptyset \).

**Proof:** Observe that the existence of a \( \mathcal{C}' \in \mathcal{Q}_I \) where \( B(\mathcal{C}') \subset B(\mathcal{C}) \) is necessary and sufficient for domination. Hence, the result follows. \( \square \)

#### Proposition 10.11

Take \( \mathcal{C} \in \mathcal{Q}_I \). Then, \( \mathcal{N}(\mathcal{C}) = F(\mathcal{C}) \).

**Proof:** Take \((a, l) \in F(\mathcal{C})\) and assume \((a, l) \notin \mathcal{N}(\mathcal{C})\). A cycle family \( \mathcal{C}' \in \mathcal{D}'_{a}(\mathcal{C}) \) has \( B(\mathcal{C}') \subset B(\mathcal{C}) \). Since \( \mathcal{C} \) is irreducible, \( B(\mathcal{C}) \subset B(\mathcal{C}') \) and therefore the projection must cause \( B(\mathcal{C}') \subset B(\mathcal{C}) \). Hence, there is an \( a' \in B(\mathcal{C}') \setminus B(\mathcal{C}) \) which is necessarily a D-arc to some \( l' \in L \), i.e. \( a' \in B'(\mathcal{C}') \cap \delta^{-}(l') \). Further, a D-arc is the last arc in a path segment, and the flow has to turn at \( l' \), so to speak, hence there is an \( a'' \in B'(\mathcal{C}) \cap \delta^{-}(l') \) and \( a''' \in B'(\mathcal{C}) \cap \delta^{-}(l') \) since no arc
can be added to \( C \) to take care of \( a'' \) so to speak. This implies \( l' \in N(B^l(C)) \) which contradicts \( C \in \overline{Q}_U \).

To determine \( N(C) \) for \( C \in Q_I \), we define critical arcs as follows. A cycle family \( C \in Q_I \) where \( l \in N(B^l(C)) \) for some \( l \in L \) induces a path in \( B^l(C) \) to \( l \). A critical arc is an arc \( a \in B^l(C) \) in a path \( P \subseteq B^l(C) \) to \( l \) with no successor in \( B^l(C) \), i.e. it is a last arc in the subpath \( P \cap B^l(C) \). Define \( L^l(C) \) to be the set of critical arcs for \( l \in L \) and \( C \in Q_I \) and \( \mathcal{L}(C) \) to be the set of all critical arcs, i.e.

\[
\mathcal{L}(C) := \bigcup_{l \in L} L^l(C). \tag{10.52}
\]

**Theorem 10.2**

Take \( C \in Q_I \). Then,

\[
N(C) = \begin{cases} 
F(C), & \text{if } |\mathcal{L}(C)| = 0, \\
(a, l), & \text{if } |\mathcal{L}(C)| = 1, \\
\emptyset, & \text{if } |\mathcal{L}(C)| > 1,
\end{cases} \tag{10.53}
\]

where \( a \) and \( l \) induce the unique critical arc in the case \( |\mathcal{L}(C)| = 1 \), i.e. \( a \in L^l(C) \).

**Proof:** If there is no critical arc, i.e. \( |\mathcal{L}(C)| = 0 \), then there is no \( l \in L \) with a path in \( B^l(C) \) to \( l \) that has some arc in \( B^l(C) \). This implies \( C \in Q_U \) and, by Proposition 10.11, \( N(C) = F(C) \).

Let \( a := (i, j) \in L^l(C) \) be the unique critical arc in the case \( |\mathcal{L}(C)| = 1 \), and let \( \bar{a} := (j, l) \in B^l(C) \) be the D-arc after \( a \) in the path to \( l \). Take \( (a', l') \in F(C) \setminus \{(a, l)\} \).

Construct \( C' \) from \( C \) by replacing the path \( \{a, \bar{a}\} = \{(i, j), (j, l)\} \) by the D-arc \( (i, j) \). Then, \( C' \in D^l_{a'}(C) \). If \( (a', l') = (a, l) \), then the argument in the proof of Proposition 10.11 can be used to show \( (a, l) \in N(C) \).

When there are several critical arcs, i.e. \( |\mathcal{L}(C)| > 1 \), it is always possible to construct some \( C' \in D^l_{a'}(C) \) for an arbitrary \( (a', l') \in F(C) \) as in the case \( |\mathcal{L}(C)| = 1 \). \( \square \)

An example is given to clarify the definition of critical arcs and to illustrate the idea in the proofs.

**Example 10.2**

![Figure 10.3: Two cycle families where dashed arcs are associated with destination node 0 and the destination of the solid arcs is unspecified. The dotted arc is a D-arc.](image)

First consider the cycle family \( C \in \overline{Q}_U \subset Q_I \) on the left in Figure 10.3. Since \( C \in \overline{Q}_U \), i.e. the cycle family is not connected to one of its destinations, there is no critical arc. Hence, \( N(C) = F(C) \) and no associated inequality is dominated. The inequalities are
10.3 A Characterization of Valid Inequalities

\( y_{34}^0 + y_{45}^0 + y_{51}^0 + y_{32}^i + y_{21}^i \leq 4 + y_{12}^0, \) \hspace{1cm} (10.54a)
\( y_{34}^0 + y_{45}^0 + y_{51}^0 + y_{32}^i + y_{21}^i \leq 4 + y_{12}^0, \) \hspace{1cm} (10.54b)
\( y_{34}^0 + y_{45}^0 + y_{51}^0 + y_{32}^i + y_{21}^i \leq 4 + y_{14}^0, \) \hspace{1cm} (10.54c)
\( y_{34}^0 + y_{45}^0 + y_{51}^0 + y_{32}^i + y_{21}^i \leq 4 + y_{15}^0, \) \hspace{1cm} (10.54d)
\( y_{34}^0 + y_{45}^0 + y_{51}^0 + y_{32}^i + y_{21}^i \leq 4 + y_{11}^0, \) \hspace{1cm} (10.54e)

Instead consider the cycle family \( C \in \tilde{Q}_D \subset \tilde{Q}_I \) on the right in Figure 10.3. The only critical arc is \((4, 5)\) and \(N(C) = (45, 0)\) and the only non-dominated inequality is the last out of the following five inequalities

\( y_{34}^0 + y_{45}^0 + y_{32}^i + y_{21}^i + y_{10}^i \leq 4 + y_{12}^0, \) \hspace{1cm} (10.55a)
\( y_{34}^0 + y_{45}^0 + y_{32}^i + y_{21}^i + y_{10}^i \leq 4 + y_{21}^0, \) \hspace{1cm} (10.55b)
\( y_{34}^0 + y_{45}^0 + y_{32}^i + y_{21}^i + y_{10}^i \leq 4 + y_{10}^0, \) \hspace{1cm} (10.55c)
\( y_{34}^0 + y_{45}^0 + y_{32}^i + y_{21}^i + y_{10}^i \leq 4 + y_{14}^0, \) \hspace{1cm} (10.55d)
\( y_{34}^0 + y_{45}^0 + y_{32}^i + y_{21}^i + y_{10}^i \leq 4 + y_{15}^0. \) \hspace{1cm} (10.55e)

By constructing a cycle family \( C' \in Q_I \) for each dominated inequality as described in the above proof, i.e. by using an appropriate subset of the ordinary arcs in Figure 10.3 and replacing some arcs with a destination arc to node 0, we obtain the associated dominating valid inequalities,

\( y_{32}^i + y_{21}^i + y_{10}^i \leq 2 + y_{12}^0, \) \hspace{1cm} (10.56a)
\( y_{21}^i + y_{10}^i \leq 1 + y_{21}^i, \) \hspace{1cm} (10.56b)
\( y_{10}^i \leq 0 + y_{10}^i, \) \hspace{1cm} (10.56c)
\( y_{34}^0 + y_{32}^i + y_{21}^i + y_{10}^i \leq 3 + y_{14}^0. \) \hspace{1cm} (10.56d)

**Figure 10.4:** A cycle family where all associated valid inequalities are dominated. The dashed arcs correspond to design variables associated with destination node 0 and the destination of the solid arcs is now 2. The dotted arc is a D-arc.

Assume that we modify the cycle family above to become as in Figure 10.4 and at the same time set the unspecified destination to 2 resulting in \( C \in \tilde{Q}_I \subset Q_I \). This family of cycles has two critical arcs; as before \((4, 5)\) for destination 0, and also \((3, 2)\) for destination 2. Since there are two critical arcs, all inequalities associated with \( C \) are dominated.
Summarizing, using Proposition 10.5 and 10.7 together with the definition of critical arcs gives an improved description of $\mathcal{P}$.

**Theorem 10.3**

The incidence vector $y \in \mathbb{B}^{A \times L}$ corresponds to an element in $\mathcal{P}$ if and only if it is a feasible solution to the system

\[
\sum_{a \in \delta^+(i)} y^1_a \leq 1, \quad i \in N, \; l \in L, \quad (10.57a)
\]

\[
\sum_{a \in \delta^-(C)} (1 - y^0_a) \geq 1, \quad C \in \mathcal{Q}_L', \; l \in L, \quad (10.57b)
\]

\[
y^0_a + \sum_{a \in \delta^+(C)} (1 - y^0_a) \geq 1, \quad \bar{a} := \bar{a}(C), \; l' := l'(C), \; C \in \bar{\mathcal{Q}}_S, \quad (10.57c)
\]

\[
y^0_{a'} + \sum_{l \in L} \sum_{a \in \delta^+(C)} (1 - y^0_a) \geq 1, \quad (a', l') \in \mathcal{N}(C), \; C \in \mathcal{Q}_I \setminus \bar{\mathcal{Q}}_S, \quad (10.57d)
\]

Some formulations above make it clear that there are close connections to independence systems. We elaborate on this in the next section.

### 10.4 Independence and Transitive Systems

Independence systems have been studied extensively in the combinatorial optimization literature. Several solution methods, exact and heuristics, have been developed and much is known about the facial structure of the associated polytope, see e.g. [81, 96, 111, 153, 170, 179]. A (less known) generalization of independence systems introducing transitive elements is transitive systems [178, 206]. Both these structures and their related optimization problems are introduced below.

**Definition 10.1**

Let $V$ be a finite base set and $\mathcal{E} \subseteq 2^V$ a collection of subsets of $V$. The pair $(V, \mathcal{E})$ is an independence system if

\[
\emptyset \in \mathcal{E} \quad \text{and} \quad I \subseteq E, \; J \subseteq I \Rightarrow J \in \mathcal{E}. \quad (10.58)
\]

**Remark 10.7.** Given a nonnegative matrix $A$ and a nonnegative vector $b$, the binary solutions to the system $Ax \leq b$ induce an independence system.

The elements in $\mathcal{E}$ are called independent sets and a subset $H \subseteq V$ such that $H \notin \mathcal{E}$ is called a dependent set. The maximal (w.r.t. set inclusion) independent sets are called the bases and the minimal dependent sets are called the circuits of $(V, \mathcal{E})$.

A convenient description of an independence system is as a vertex packing in a hypergraph. Denote by $\bar{\mathcal{E}}$ the set of circuits in the independence system $(V, \mathcal{E})$. The conflict hypergraph, $\mathcal{H}(V, \mathcal{E}) = (V, \bar{\mathcal{E}})$, is the hypergraph with a vertex for each element in $V$ and a hyperedge for each circuit in $\bar{\mathcal{E}}$. A vertex packing in the hypergraph corresponds 1-to-1 to an independent set in $(V, \mathcal{E})$. 
Given a set of weights, $c_i$ for each item $i \in V$, the independent set problem is to find an independent set $I \subseteq V$ of maximal weight. This is equivalent to finding a maximum weight vertex packing in the corresponding hypergraph. Denote the incidence matrix of an arbitrary hypergraph $H$ by $A_H$ and let $p_H$ be the vector where element $i$ is the number of positive entries in row $i$ of $A_H$, i.e. $p_H = 1'(A_H)^+$. This yields the following model for the independent set problem,

$$\text{maximize } w'x$$

$$\text{subject to } A_Hx \leq p_H - 1,$$

$$x \in \mathbb{B}^V,$$

where the variable $x_i \in \mathbb{B}$ is 1 if item/vertex $i \in V$ is selected.

**Remark 10.8.** Observe that any 0/1 matrix can be interpreted as a hypergraph incidence matrix (w.l.o.g assume that there are no empty edges and no loops since these cases can trivially be reduced).

**Remark 10.9.** By construction, all hyperedges in $H(V,E)$ correspond to circuits of the independence system and the hypergraph is a so called clutter, i.e. no hyperedge is contained in another. Therefore, no row of $A_H$ is trivially dominated in (10.59).

In [178, 206], transitive packings are introduced as a unifying concept in combinatorial optimization. They generalize hypergraph vertex packings by allowing transitive elements. This implies that the incidence matrix can be almost any $0/\pm 1$ matrix in (10.59). A myriad of combinatorial optimization problems fit into this framework.

Let us formally introduce the transitive packing problem in hypergraphs, a minor deviation from the presentation in [178] is made to cover arbitrary $0/\pm 1$ matrices. This implies that even more combinatorial optimization problems fit into the framework, e.g. the set covering, partitioning and packing problems and also acyclic Steiner digraphs problems induced by the set $I$ introduced earlier in this chapter. A price we pay for this generality is that the associated polytope is no longer necessarily full-dimensional.

**Definition 10.2 (cf. Definition 2.1 in [178])**

An extended hypergraph is a triple, $H = (V,E,\text{tr})$, consisting of a set of vertices, $V$, a multiset of hyperedges, $E$, and a transitivity mapping, $\text{tr} : E \rightarrow 2^V$. It also holds for each hyperedge $E \in E$ that $E \subseteq V$ and $\text{tr}(E) \subseteq V \setminus E$.

Note that multiple and empty edges are allowed, this implies that any $0/\pm 1$ matrix can be interpreted as an incidence matrix of an extended hypergraph.

**Definition 10.3 (cf. Definition 2.1 in [178])**

Let $H = (V,E,\text{tr})$ be an extended hypergraph. The vertex subset $S \subseteq V$ is a transitive packing in $H$ if for every $E \in E$ where $E \subseteq S$ there is a vertex $i \in S \cap \text{tr}(E)$.

**Definition 10.4 (cf. Section 2 in [178])**

Let $H = (V,E,\text{tr})$ be an extended hypergraph and $w \in \mathbb{Q}^V$ a weight vector. The maximum weight transitive packing problem is to find a transitive packing in $H$ of maximum
weight. The transitive packing polytope is the convex hull of all incidence vectors that correspond to transitive packings.

Given an extended hypergraph, $\mathcal{H}$, the obvious generalization of the notation associated with model (10.59) implies that (10.59) is a valid formulation of the maximum weight transitive packing problem. A less compact, but more comprehensible, formulation is

$$\text{maximize } \sum_{i \in V} w_i x_i$$

subject to

$$\sum_{i \in E} x_i - \sum_{i \in \text{tr}(E)} x_i \leq |E| - 1, \quad E \in \mathcal{E}, \quad (10.60a)$$

$$x_i \in \mathbb{B}, \quad i \in V. \quad (10.60b)$$

Remark 10.10. If there are no transitive elements, i.e. $\text{tr}(E) = \emptyset$ for all $E \in \mathcal{E}$, then the transitive packing problem coincides with the hypergraph vertex packing problem and a transitive packing is simply an independent set.

Remark 10.11. Note that restricting the hyperedge cardinality to be at least 2 implies that for instance the models associated with $\mathcal{I}$, $\mathcal{P}$ and $\mathcal{Y}$ do not fit into the transitive packing framework. To also include these models, we allow empty hyperedges and loops.

Several results about the transitive packing polytope are given for the case where the edges in $\mathcal{H}$ have cardinality at least 2 in [178]. In particular, cutting plane proofs are given for very large classes of valid inequalities for the transitive packing polytope, e.g. cycle, clique, anti-hole and anti-web inequalities. These classes are generalizations of valid inequalities that are commonly used in other combinatorial optimization problems, e.g. the Möbius ladder and fence inequalities for the acyclic subgraph polytope in [126, 160], for details see [178].

Our main interest in transitive packings lies in the special case where all edges are transitively mapped to singletons, i.e. for all $E \in \mathcal{E}$ there is some $u \in V$ such that $\text{tr}(E) = \{u\}$. This restriction gives what we refer to as 1-transitive packings. Refining model (10.60) to 1-transitive packings and re-writing yields,

$$\text{maximize } \sum_{i \in V} w_i x_i$$

subject to

$$x_u + \sum_{i \in E} (1 - x_i) \geq 1, \quad \{u\} = \text{tr}(E), \quad E \in \mathcal{E}, \quad (10.61a)$$

$$x_i \in \mathbb{B}, \quad i \in V. \quad (10.61b)$$

It should be clear that many models from above fit into the transitive system framework, e.g. (10.11) and (10.12) when $S^l = N \setminus \{l\}$, and (10.26), (10.46), (10.57) for all $S^l$. In particular, model (10.46) and (10.57), i.e. $\mathcal{P}$ and $\mathcal{P}$, describe 1-transitive packings. In the compatibility version of the USPR case, there are no transitive elements. Hence, $\mathcal{P}^C$ induce an independence system.
10.4 Independence and Transitive Systems

10.4.1 Independence and Transitive System Interpretations

There is a close resemblance between the models associated with feasible routing patterns and independence systems and transitive systems. In fact, all collections introduced in this chapter can be seen as transitive systems and several as either 1-transitive systems or independence systems.

**Proposition 10.12**

*All the collections $I, \mathcal{I}, \mathcal{P}^C, \mathcal{P}^G, \mathcal{P}, \mathcal{Y}$ and $\mathcal{Y}$ induce transitive systems.*

**Proof:** It suffices to show that all constraints in the models associated with the collections can be written as

$$\sum_{i \in I} x_i - \sum_{i \in \bar{I}} x_i \leq |I| - 1, \quad (10.62)$$

or can be obtained as valid inequalities from constraints of the form in (10.62). We show this for the non-trivial cases, i.e. (10.11b), (10.12a) and equality constraints.

**Constraint (10.11b).** To accurately describe $\mathcal{I}$ we forced the outdegree of a node $i \in S^I$ to be less than the indegree, see (10.11b). Since $\mathcal{I} \subset \mathcal{I}$, the corresponding constraint (10.12b) is valid. An induction argument shows that (10.11b), i.e.

$$\sum_{a \in \delta^+(i)} y^l_a \leq \sum_{a \in \delta^-(i)} y^l_a, \quad (10.63)$$

is obtained from (10.12b), i.e.

$$y^l_a \leq \sum_{a \in \delta^-(i)} y^l_{\bar{a}}, \quad \bar{a} \in \delta^+(i), \quad (10.64)$$

and the outdegree constraint via a Chvátal-Gomory cutting plane proof (a $\{0, \frac{1}{2}\}$-cut even). Indeed, for $I \subset \delta^+(i)$ and $a \in \delta^+(i) \setminus I$, combining and rounding yields,

$$\frac{1}{2} y^l_a + \sum_{a \in \delta^+(i)} y^l_a \leq \sum_{a \in \delta^-(i)} y^l_a, \quad (10.65a)$$

$$\frac{1}{2} y^l_a \leq \sum_{a \in \delta^-(i)} y^l_a, \quad (10.65b)$$

$$\frac{1}{2} y^l_a + \sum_{a \in \delta^+(i)} y^l_a \leq 1, \quad (10.65c)$$

$$y^l_a \leq \sum_{a \in \delta^+(i)} y^l_a \leq \sum_{a \in \delta^-(i)} y^l_a, \quad (10.65d)$$

**Constraint (10.12a).** Set $I = \emptyset$ and $\bar{I} = \delta^+(i)$ in (10.62) to obtain
\[ - \sum_{a \in \delta^{+}(i)} y_{a} \leq -1, \]  

which is equivalent to (10.12a).

Equality constraints. Since all right hand sides in equality constraints involve outdegrees they are 1. These constraints can be seen as the combination of an outdegree clique constraints and a covering constraint (i.e. of type (10.66)) both with right hand side 1.

For most of the collections in Proposition 10.12 it is possible to specialize further and give independence or 1-transitive system interpretations.

**Proposition 10.13**
The collections \( P_{C} \), \( \overline{P} \) and \( \mathcal{P} \) induce 1-transitive systems.

**Proposition 10.14**
The collection \( \overline{P}_{C} \) induce an independence system.

It is easy to see that any of collections not mentioned as an independence system in the above proposition in general is not an independence system. A simple proof is by providing two maximal elements with different rank. As an example take \( \mathcal{I} \) and a spanning arborescence and a Steiner arborescence that is not spanning.

If we consider the special cases where a non-negative objective is to be minimized or the arborescences or ingraphs are required to be spanning, it is possible to obtain some additional results about the interpretation as independence systems.

**Proposition 10.15**
When \( w \in \mathbb{Q}_{+}^{A} \), the problem of minimizing \( w'y \) over \( y \in \mathcal{I} \) can be solved over an independence system.

**Proof:** When \( w \in \mathbb{Q}_{+}^{A} \), the outdegree constraint required to be accurate is redundant, hence model (10.10) can be used. Complementing variables, i.e. using \( x := 1 - y \), induces an independence system or a maximum weight independent set problem. (We are simply using the affine equivalence between set covering and independence systems.)

The most important results are obtained for the special case where all arborescences are spanning, i.e. when \( S^{l} = N \setminus \{l\} \) for all \( l \in L \).

**Remark 10.12.** The requirement \( S^{l} = N \setminus \{l\} \) for all \( l \in L \) is not as restrictive as it may seem. Every solution must be augmentable to a solution for an extended problem, i.e. with more OD-pairs, where this property is satisfied. Indeed, a weight certificate associated with a feasible routing pattern induces a routing pattern that spans all nodes.

**Proposition 10.16**
When \( S^{l} = N \setminus \{l\} \) for all \( l \in L \), the elements in \( \mathcal{I} \) correspond 1-to-1 to the bases of an independence system that is the intersection of two matroids.

This result and the matroid view below are well known, see e.g. [205]. The matroids are the partitioning matroid and the cycle matroid in the underlying undirected graph. The matroid intersection interpretation implies that the associated optimization problem is polynomially solvable. Polynomial algorithms, not relying on matroid theory, include [54, 78, 97, 217].
Proposition 10.17
When $S^i = N \setminus \{l\}$ for all $l \in L$, the elements in $\mathcal{Y}$ correspond 1-to-1 to the bases of the independence system induced by $\mathcal{P}^C$.

Proof: The only issue with the formulations of $\mathcal{Y}$ stems from non-improving cycle families corresponding to paths, i.e. constraints of the form (10.46d) that have a single variable with a negative coefficient. However, since $S^i = N \setminus \{l\}$, all outdegree constraints are equalities that can be used to replace this single variable with the other variables associated with emanating arcs from the same node. Hence, the elements in $\mathcal{Y}$ correspond to the bases of some independence system. Since the bases of the independence system induced by $\mathcal{P}^C$ corresponds to spanning arborescences that are partially compatible, they are also realizable. Therefore, $\mathcal{P}^C$ induce the sought independence system.

We will elaborate further on this result and the interpretations as independence and transitive systems in the next section using conflict hypergraphs.

10.4.2 The Conflict Hypergraph for Routing Patterns

In this section, we explicitly consider the hypergraph aspect of hypergraph (transitive) packing formulations for the two sets of feasible routing patterns, $\mathcal{P}$ and $\mathcal{P}$, considered above. In particular, we focus on the associated conflict hypergraph similar to the conflict hypergraph for independence systems. We believe that this is a suitable representation. Especially, and most importantly, it can be helpful when classes of valid inequalities are to be derived based on the rich hypergraph structures considered in the literature for hypergraph (transitive) packings e.g. cycle, clique, anti-hole, anti-web, and odd partition inequalities, and generalizations thereof. There is a vast literature on this subject, see e.g. [81, 96, 111, 153, 170, 178, 179].

A conflict hypergraph approach similar to ours is used in [46]. There, the path formulation for the compatibility version of the USPR version is considered. We consider the arc formulations for the partial realizability version of SPR without splitting, i.e. USPR, and with splitting, i.e. ECMP. This gives some significant differences. We begin by considering the USPR case and then the ECMP case.

The USPR Case

In view of Proposition 10.17 and Remark 10.12 it seems natural to focus on $\mathcal{P}^C$. In a sense this suffices, but some inequalities for $\mathcal{Y}$ are easier to obtain by instead considering $\mathcal{P}$, or the relaxation of $\mathcal{P}$ mentioned in Remark 10.6.

Due to the non-improving cycle families in $\mathcal{Q}_S$ associated with paths, $\mathcal{P}$ is not an independence system, see constraint (10.46d). To maintain the independence system structure, we consider a relaxation referred to as $\mathcal{P}'$ of of $\mathcal{P}$ where these constraints are not taken into account. This can be useful since it makes it easier to analyze the facial structure. The relaxation $\mathcal{P}'$ becomes
\[
\sum_{a \in \delta^+(i)} y_a^l \leq 1, \quad i \in N, \quad l \in L,
\]
(10.67a)
\[
\sum_{a \in B_l^+(C)} (1 - y_a^l) \geq 1, \quad C \in Q^l_C, \quad l \in L,
\]
(10.67b)
\[
\sum_{l \in L} \sum_{a \in B_l^+(C)} (1 - y_a^l) \geq 1, \quad C \in Q^l \cup \tilde{Q}_P.
\]
(10.67c)

As pointed out in Remark 10.6, \( \overline{P} \) is a stronger relaxation than \( \overline{P}' \). Also, when \( S^l = N \setminus \{i\} \) for all \( l \in L \), the bases of this independence system corresponds to the bases of the independence system associated with \( \overline{P}' \) and hence the elements in \( \overline{Y} \).

We define the conflict hypergraphs, \( \mathcal{H}^C = (V, \mathcal{E}^C) \) and \( \mathcal{H}^I = (V, \mathcal{E}^I) \), associated with \( \overline{P}' \) and \( \overline{P} \), respectively. Both hypergraphs have a vertex, \( v \in V \), for each arc destination pair \( (a, l) \in A \times L \). The hyperedge sets differ, \( \mathcal{H}^C \) has a hyperedge for each irreducible cycle family without a D-arc, i.e. for each \( C \in Q^l_U \) and \( \mathcal{H}^I \) has a hyperedge for the irreducible cycle families without D-arcs in \( \overline{Q}_U \) and a hyperedge for the irreducible cycle families with a D-arc in \( \overline{Q}_P \). In addition, both \( \mathcal{H}^C \) and \( \mathcal{H}^I \) have a hyperedge for each directed cycle and for each pair vertices corresponding to two arcs emanating from the same node with the same destination. This yields

\[
V = A \times L,
\]
(10.68a)
\[
\mathcal{E}^C = \mathcal{E}' \cup \mathcal{E}'' \cup \mathcal{E}(Q_U) \cup \mathcal{E}(Q_D),
\]
(10.68b)
\[
\mathcal{E}^I = \mathcal{E}' \cup \mathcal{E}'' \cup \mathcal{E}(Q_U) \cup \mathcal{E}(Q_P),
\]
(10.68c)

where

\[
\mathcal{E}(Q) = \left\{ E \subseteq 2^V \left| E = \bigcup_{l \in L} \bigcup_{a \in B_l^+(C)} \{(a, l)\}, \ C \in Q \right. \right\},
\]
(10.68d)

and

\[
\begin{align*}
\mathcal{E}' & = \left\{ E \subseteq 2^V \left| E = \bigcup_{a \in B_l^+(C)} \{(a, l)\}, \ C \in Q^l_C, \ l \in L \right. \right\}, \\
\mathcal{E}'' & = \left\{ E \subseteq 2^V \left| E = \{(v^1, v^2), v^1 = (a^1, l), v^2 = (a^2, l)\}, \ a^1, a^2 \in \delta^+(i), \ i \in N, l \in L \right. \right\}.
\end{align*}
\]
(10.68f)

\( \mathcal{E}'' \)

Remark 10.13. Note that \( \mathcal{H}^C \not\subset \mathcal{H}^I \), but \( \mathcal{H}^C \subset \overline{\mathcal{H}}^I \) for some \( \overline{\mathcal{H}}^I \supset \mathcal{H}^I \) that induces the same bases as \( \mathcal{H}^I \), i.e. \( \overline{\mathcal{H}}^I \) is obtained from \( \mathcal{H}^I \) by adding hyperedges with redundant elements to some hyperedges in \( \mathcal{H}^I \) corresponding to circuits.
A clutter is a hypergraph where no hyperedge properly contains another, i.e. for two different hyperedges \( E_1 \neq E_2 \in \mathcal{E} \), we have \( E_1 \nsubseteq E_2 \).

Observe that a hypergraph that is not a clutter contains some redundant hyperedges. In terms of an integer formulation of an independence system given via a hypergraph, this corresponds to redundant (surrogate) inequalities.

**Proposition 10.18**

\( \mathcal{H}^I \) is a clutter.

**Proof:** Since all cycle families in \( \mathcal{Q}_I = \mathcal{Q}_U \cup \mathcal{Q}_D \) and \( \mathcal{Q}^P_C \) are irreducible, the only potential redundancy stem from hyperedges from \( \mathcal{E}' \). However, since no irreducible cycle family can contain two emanating arcs from the same node, a hyperedges in \( \mathcal{E}' \) cannot be contained in a hyperedge in \( \mathcal{E} \), or vice versa.

Due to the partitioning in (10.42) and our efforts in Proposition 10.6-10.9, the hypergraph \( \mathcal{H}^I \) is also a clutter.

**Proposition 10.19**

\( \mathcal{H}^I \) is a clutter.

**Proof:** Since \( \mathcal{H}^I \) is a clutter, the sub-hypergraph of \( \mathcal{H}^I \) without the hyperedges in \( \mathcal{E}(P) \) is a clutter. Assume that \( \mathcal{H}^I \) is not a clutter, by irreducibility, the only possibility of destroying the clutter structure is by the projection operation, i.e. if \( \mathcal{C} \in \mathcal{E}(\mathcal{Q}_P) \) and \( \mathcal{C} \in \mathcal{E}(\mathcal{Q}_D) \) and \( B(C) \subseteq B(\mathcal{C}) \) or \( B(\mathcal{C}) \subseteq B(C) \). Since \( \mathcal{C} \) does not have a D-arc, the only possibility is \( B(\mathcal{C}) \subseteq B(C) \). Similar to the proof of Proposition 10.11, we will conclude \( \mathcal{C} \in \mathcal{Q}_D \).

The arcs in \( B(\mathcal{C}) \) are necessarily D-arcs and there must be some \( l \in L \) and an \( \tilde{a} \in B(L) \). By irreducibility, \( B(\mathcal{C}) \not\subseteq B(C) \), and from above \( B(\mathcal{C}) \subseteq B(C) \). This implies that \( \mathcal{C} \) is obtained from \( \mathcal{C} \) by removing \( \tilde{a} \) and some other arcs. Therefore, \( l \in N(B(C)) \) and \( \mathcal{C} \in \mathcal{Q}_D \). A contradiction.

**Remark 10.14.** Had we not excluded the irreducible cycle families in \( \mathcal{Q}_D \) in the description of \( \mathcal{P}^C \) in (10.67), then \( \mathcal{H}^C \) would not have been a clutter as seen in the above proof.

**Remark 10.15.** As in the case of the path formulation analyzed in [46], it is easy to see that none of the independence systems \( \mathcal{P}' \) or \( \mathcal{P}^C \) are in general matroids. Recall that a matroid has the following property: if \( A \) and \( B \) are independent sets and \( A \) is larger than \( B \), then there is an element in \( A \) that can be added to \( B \) to obtain a larger independent set. Consider the two sets \( A \) and \( B \) defined as \( A = \{(13, 2), (13, 3), (21, 3)\} \) and \( B = \{(12, 3), (23, 3)\} \). Then, \( A \) and \( B \) are independent sets in either both \( \mathcal{P}' \) and \( \mathcal{P}^C \). However, adding an element from \( A \) to \( B \) does not give an independent set. Indeed, adding \((13, 2)\) creates a valid cycle, adding \((13, 3)\) induces splitting and adding \((21, 3)\) creates a directed cycle. Hence, the independence systems cannot be matroids.

**Proposition 10.20**

Neither \( \mathcal{P}' \), nor \( \mathcal{P}^C \), is in general a matroid.
Consider the independence system induced by compatibility for the graph on the left in Figure 10.5. Assume that there are two destinations, $L = \{l_0, l_1\}$. The conflict hypergraph is given in Figure 10.6. First consider conflicts involving a single destination, i.e., due to directed cycles and splitting. The associated (hyper)edges in Figure 10.6 are solid if only two arc destination pairs are involved and dashed if three arc destination pairs are involved. The cycle families corresponding to directed cycles are given by,

\begin{align}
\{(12, l), (21, l)\},
\{(12, l), (23, l), (31, l)\},
\end{align}

and the cycle families arising from splitting are given by,

\begin{align}
\{(21, l), (23, l)\}, \{(21, l), (24, l)\}, \{(23, l), (24, l)\}, \{(31, l), (34, l)\},
\end{align}

where $l \in L$.

\textbf{Example 10.3}

Consider the independence system induced by compatibility for the graph on the left in Figure 10.5. Assume that there are two destinations, $L = \{l_0, l_1\}$. The conflict hypergraph is given in Figure 10.6. First consider conflicts involving a single destination, i.e., due to directed cycles and splitting. The associated (hyper)edges in Figure 10.6 are solid if only two arc destination pairs are involved and dashed if three arc destination pairs are involved. The cycle families corresponding to directed cycles are given by,

\begin{align}
\{(12, l), (21, l)\},
\{(12, l), (23, l), (31, l)\},
\end{align}

and the cycle families arising from splitting are given by,

\begin{align}
\{(21, l), (23, l)\}, \{(21, l), (24, l)\}, \{(23, l), (24, l)\}, \{(31, l), (34, l)\},
\end{align}

where $l \in L$.

\textbf{Figure 10.5:} Two graphs used in Example 10.3 and 10.4.

\textbf{Figure 10.6:} The conflict hypergraph associated with the independence system induced by compatibility and the graph on the left in Figure 10.5. The destination of nodes is indicated by a solid or a dashed vertex. Solid edges correspond to conflicts involving two arcs, i.e., splitting conflicts and directed cycles with two nodes. Dashed hyperedges correspond to directed cycle conflicts on three nodes. Dotted hyperedges correspond to valid cycle conflicts on three or four nodes.
Conflicts with two destinations are associated with valid cycles and correspond to dashed hyperedges in Figure 10.6. Their cycle families belong to \( \overline{Q}_I \) and are given by,

\[
\begin{align*}
\{(23, l'), (31, l'), (21, l'')\}, & \quad \text{(10.70a)} \\
\{(23, l'), (34, l'), (24, l'')\}, & \quad \text{(10.70b)} \\
\{(21, l'), (34, l'), (24, l''), (31, l'')\} & \quad \text{(10.70c)}
\end{align*}
\]

where \( l', l'' \in L \) and \( l' \neq l'' \).

**Example 10.4**

We illustrate that the conflict hypergraph can by useful to obtain strong inequalities. Take the graph on the right in Figure 10.5 and consider destination \( l = 1 \) and some arbitrary destination \( l' \). The conflict hypergraph for the independence system associated with \( \overline{P}' \) is given in Figure 10.7.

![Figure 10.7: The conflict hypergraph induced by destinations \( l \) and \( l' \) for the graph on the right in Figure 10.5. Black vertices correspond to arcs in the path 4-3-2-1 for destination \( l' \) and grey vertices correspond to arcs emanating from node 4 associated with destination \( l = 1 \). Observe that a hyperclique is given by \( \{45, 46, 47, \{21, 43, 32\}\} \).](image)

Conflicts associated with splitting gives dashed edges in Figure 10.7. The induced clique corresponds to the outdegree constraint

\[
\sum_{a \in \delta^+(4)} y_a^l = y_{43}^1 + y_{45}^1 + y_{46}^1 + y_{47}^1 \leq 1.
\] (10.71)

The cycle families corresponding to conflicts associated with the path 4-3-2-1 and an arc emanating from node 4 all belong to \( \overline{Q}'_P \). They yield the hyperedges,

\[
\begin{align*}
\{(21, l'), (32, l'), (43, l'), (45, l)\}, & \quad \text{(10.72a)} \\
\{(21, l'), (32, l'), (43, l'), (46, l)\}, & \quad \text{(10.72b)} \\
\{(21, l'), (32, l'), (43, l'), (47, l)\}, & \quad \text{(10.72c)}
\end{align*}
\]

which together with the edges

\[
\{(45, l), (46, l)\}, \{(45, l), (47, l)\}, \{(46, l), (47, l)\},
\]

forms the hyperclique \( \{45, 46, 47, \{21, 43, 32\}\} \) (destinations have been omitted for readability). Hence the inequality

\[
\]
178

Feasible Routing Pattern Polytopes

\[ y_{45}^1 + y_{46}^1 + y_{47}^1 + y_{43}^l + y_{32}^l + y_{21}^l \leq 1 + 3(1 - 1) + (3 - 1) = 3 \] (10.73)
is valid for \( \text{conv} \mathcal{P} \). (3 parts in the clique has 1 vertex and contributes (1-1) to the right hand side and the last part has 3 vertices and contributes (3-1) to the right hand side.)

Writing out the inequalities associated with the (hyper)edges, i.e.

\[ y_{45}^1 + y_{43}^l + y_{32}^l + y_{21}^l \leq 3, \] (10.74a)
\[ y_{46}^1 + y_{43}^l + y_{32}^l + y_{21}^l \leq 3, \] (10.74b)
\[ y_{47}^1 + y_{43}^l + y_{32}^l + y_{21}^l \leq 3, \] (10.74c)
\[ y_{45}^1 + y_{46}^1 \leq 1, \] (10.74d)
\[ y_{45}^1 + y_{47}^1 \leq 1, \] (10.74e)
\[ y_{46}^1 + y_{47}^1 \leq 1, \] (10.74f)

it is easy to prove the validity of (10.73) using a \{0, \frac{1}{2}\} Chvátal-Gomory cutting plane proof in two steps (recall that the Chvátal-Gomory rank for cliques is two less the size of the clique, i.e. 4-2=2 in our case).

Let us now apply the same approach for the ECMP case.

### The ECMP Case

To describe the conflict hypergraph in the ECMP case, we start with the formulation (10.57) of \( \mathcal{P} \) in Theorem 10.3. This system fits into the 1-transitive packing framework, see Proposition 10.13. Recall that the collection of all feasible and irreducible cycle families is denoted by \( \mathcal{Q} \), which can be be partitioned as in (10.42).

A conflict hypergraph is defined in a similar manner as in the USPR case. The conflict hypergraph, \( \mathcal{H}^\mathcal{P} = (V, \mathcal{E}) \) has a vertex each arc destination pair \((a, l) \in A \times L\) and a set of hyperedges for each each irreducible cycle family in \( \mathcal{Q} \). More precisely, for each \( C \in \mathcal{Q} \) and each \((a, l) \in \mathcal{N}(C)\), there is an \( E \in \mathcal{E} \) with \( \text{tr}(E) = \{(a, l)\} \).

The conflict hypergraph, \( \mathcal{H}^\mathcal{P} \), in general has multi-hyperedges in the ECMP case. The projection onto the simple underlying hypergraph (without multi-hyperedges) does not yield a clutter even though \( \mathcal{Q} \) is "appropriately" defined. It is more convenient to present this underlying simple hypergraph than \( \mathcal{H}^\mathcal{P} \). Moreover, the transitive elements is in a sense "clear" from the context anyway.

#### Example 10.5

Consider a problem where the graph on the left in Figure 10.8 is a subgraph. The restriction to only design variables associated with these arcs and two destinations, an unspecified destination, say 0, and destination 1, gives the (projection of the) restricted conflict hypergraph \( \mathcal{H}^\mathcal{P} \), on the right in Figure 10.8. There are 6 arcs in the original subgraph and two destinations are considered. The arc \((1, 3)\) emanates from a destination, i.e. node 1. Therefore, there are \( 11 = 2 \cdot 6 - 1 \) hyperedges in \( \mathcal{H}^\mathcal{P} \).
10.4 Independence and Transitive Systems

Figure 10.8: (Left) A subgraph of the underlying graph. (Right) The (projection of the) restricted conflict hypergraph (right) for the part of the instance induced by the subgraph and destination 0 and 1. The hypernodes are labelled with the arc. A solid black node is associated with destination 0 and a grey node is associated with destination 1. Each hyperedge corresponds to at least one potential SPR conflict.

There are several (potential) conflicts which all induce at least one hyperedge. In short, the dashed hyperedges correspond to directed cycles and are therefore not associated with transitive elements. The dotted hyperedges represent routing conflicts (valid cycles) without D-arcs and cannot involve node 1. There is one hyperedge for each arc in the conflict. The solid hyperedges represent routing conflicts (valid cycles) that depend on D-arcs and must therefore involve node 1. There is only one hyperedge associated with each of these conflicts since there is a unique critical arc. A more detailed description in terms of the valid inequalities follows.

**Directed cycles** correspond to dashed hyperedges. This yields 5 hyperedges without transitive elements and 5 valid inequalities. Namely,

\[
\begin{align*}
y_{0}^{3} + y_{3}^{0} & \leq 1, \\
y_{2}^{0} + y_{0}^{2} & \leq 1, \quad \text{and} \quad y_{2}^{0} + y_{3}^{2} + y_{4}^{3} & \leq 2, 
\end{align*}
\]

(10.75)

for destination 0, and

\[
\begin{align*}
y_{1}^{3} + y_{3}^{1} & \leq 1, \\
y_{2}^{1} + y_{3}^{2} + y_{4}^{3} & \leq 2.
\end{align*}
\]

(10.76)

for destination 1.

**Routing conflicts with no destination arc** between two SP-graphs correspond to dotted hyperedges. This yields 2 hyperedges with 3 transitive elements each and 6 valid inequalities in total. From hyperedge \{(34, 0), (32, 1), (24, 1)\} we get,
10 Feasible Routing Pattern Polytopes

and the hyperedge \{(24, 0), (32, 0), (34, 1)\} yields

\[
\begin{align*}
y_{0}^{24} + y_{0}^{0} + y_{1}^{0} & \leq 2 + y_{24}^{0}, \\
y_{0}^{24} + y_{0}^{1} + y_{1}^{1} & \leq 2 + y_{24}^{1}, \\
y_{0}^{24} + y_{0}^{0} + y_{1}^{0} & \leq 2 + y_{24}^{1}.
\end{align*}
\]

Routing conflicts with destination arcs between two SP-graphs correspond to solid hyperedges and yield one inequality for each hyperedge (smaller hyperedges may be contained in the original hyperedge which yields more non-dominated inequalities).

From the hyperedge \{(24, 0), (31, 0), (43, 0)\} and its sub-hyperedges, we get

\[
\begin{align*}
y_{0}^{24} + y_{0}^{0} + y_{1}^{0} & \leq 2 + y_{24}^{0}, \\
y_{0}^{24} + y_{0}^{1} + y_{1}^{1} & \leq 2 + y_{24}^{1}, \\
y_{0}^{24} + y_{0}^{0} + y_{1}^{0} & \leq 2 + y_{24}^{1}.
\end{align*}
\]

Hyperedge \{(32, 1), (24, 1), (31, 0)\} and its sub-hyperedges yield,

\[
\begin{align*}
y_{1}^{24} + y_{1}^{1} + y_{0}^{0} & \leq 2 + y_{24}^{0}, \\
y_{1}^{24} + y_{0}^{1} & \leq 1 + y_{24}^{1}, \\
y_{1}^{24} & \leq 0 + y_{24}^{1}.
\end{align*}
\]

Finally, the last hyperedge \{(34, 1), (24, 0), (31, 0)\} and its sub-hyperedges yield,

\[
\begin{align*}
y_{0}^{24} + y_{1}^{1} + y_{0}^{0} & \leq 2 + y_{24}^{0}, \\
y_{1}^{24} + y_{0}^{0} & \leq 1 + y_{24}^{0}, \\
y_{1}^{24} & \leq 0 + y_{24}^{0}.
\end{align*}
\]

The hypergraph description may be very fruitful in the derivation of classes of valid inequalities based on routing conflicts, see e.g. Example 10.4, and lifting techniques. We will not investigate this here but believe that it is a promising area for future research.
Dantzig-Wolfe Reformulations in Unique Shortest Path Routing

The Dantzig–Wolfe reformulation technique is a standard approach to (structured) mathematical programs. When applied to a mixed-integer linear program (MILP), the aim is often to obtain an equivalent formulation of the original problem that results in an LP-relaxation that is either stronger or easier to solve. The method is particularly well suited to utilize problem structure by decomposition.

To solve the Dantzig–Wolfe reformulation, two techniques referred to as Branch-and-Price (B&P) and Branch-and-Cut-and-Price (B&C&P) are frequently used. In this chapter we apply the Dantzig–Wolfe reformulation technique to the unique shortest path routing (USPR) problem considered earlier. In particular, we describe means to deal with the most important issues arising in the B&P and B&C&P context, i.e. branching, pricing and cutting.

Outline We begin by considering a Dantzig–Wolfe reformulation of a relaxation of a USPR problem in Section 11.1 which naturally yields a Dantzig–Wolfe reformulation of the original USPR problem in Section 11.2. Then, the branching, pricing and cutting aspects are handled in Section 11.3. In Section 11.4, we show how to translate a cut in the original space into a stronger cut in the extended space. Finally, we conclude in Section 11.5 by discussing some practical considerations related to the column generation nature of the Dantzig–Wolfe reformulation approach for our problem.

11.1 A Relaxation of a USPR Problem

As the starting point for deriving Dantzig–Wolfe reformulations in this chapter we take a relaxation of a USPR problem obtained by ignoring the shortest path routing (SPR) aspect. First, we will make an assumption. Throughout the chapter it is assumed that every destination, \( l \in L \), satisfies
\[ K_l = N \setminus \{l\}, \quad (11.1) \]
i.e. for destination \( l \), there is an OD-pair associated with every origin. Hence, the induced anti-arborescences are spanning. Note that this is not much of a restriction since it is possible to set the demand to 0 for an OD-pair not in \( K \). An important consequence of this assumption is that the relation between the \( x \)- and \( y \)-variables becomes stronger. Indeed,

\[
\sum_{a \in \delta^+(i)} y^l_a = \sum_{a \in \delta^+(i)} x^l_k = 1, \quad i = o^k, \ k \in K_l, \ l \in L, \quad (11.2)
\]

and

\[
x^l_k \leq y^l_a, \quad a \in A, \ k \in K_l, \ l \in L, \quad (11.3)
\]

which implies that

\[
x^l_k = y^l_a, \quad a \in \delta^+(i), \ i = o^k, \ k \in K_l, \ l \in L, \quad (11.4)
\]

Hence, all \( y \)-variables can be replaced by some \( x \) variable. This substitution is omitted since it would completely destroy the readability. However, we strongly encourage the inclusion of the constraints in (11.4) so that the preprocessor can do this reduction; it greatly affects the size of a USPR model.

Let us consider the above mentioned relaxation. Take the minimalistic core problem (9.4) in Section 9.1 and drop the requirement that routing is conducted via shortest paths. Actually, this is ambiguous since it can be interpreted as single-path routing and as tree-routing. In the former case, a standard single-path multicommodity flow problem is obtained, but with a less common objective, i.e. to minimize the congestion measured as the maximally utilized link,

\[
\begin{align*}
&\text{minimize} & \zeta \\
&\text{subject to} & \sum_{k \in K} h^k x^k_a \leq u_a \zeta, & a \in A, \quad (11.5a) \\
& & \sum_{a \in \delta^+(i)} x^k_a - \sum_{a \in \delta^-(i)} x^k_a = b^k_i, & i \in N, \ k \in K, \quad (11.5b) \\
& & x^k_a \in \mathbb{B}, & a \in A, \ k \in K. \quad (11.5c)
\end{align*}
\]

where again \( b \) denotes the node balance vector, see e.g. (9.5) on page 131.

In the latter case, i.e. when the traffic is routed along trees, but the SPR constraints are dropped, the resulting problem is,
minimize $\zeta$
subject to
\[
\sum_{k \in K} h^k x^k_a \leq u_a \zeta, \quad a \in A,
\]
\[
\sum_{a \in \delta^+(i)} x^k_a = \sum_{a \in \delta^-(i)} x^k_a = b^k_i, \quad i \in N, k \in K,
\]
\[
\sum_{a \in \delta^+(i)} y^k_a \leq 1, \quad i \in N, l \in L,
\]
\[
0 \leq x^k_a \leq y^k_a, \quad a \in A, k \in K^l, l \in L,
\]
\[
x^k_a \in B, \quad a \in A, k \in K^l, l \in L,
\]
\[
y^k_a \in B, \quad a \in A, l \in L.
\]
Both these relaxations lend themselves natural to Dantzig–Wolfe reformulations. First consider the single-path relaxation (11.5).

The flow conservation constraint (11.5b) induces paths. For each $l \in L$ and each $k \in K^l$, denote by $\mathcal{P}^{kl}$ the collection of all $(o^k, l)$-paths. Given a path, $p \in \mathcal{P}^{kl}$, and an arc, $a \in A$, define $\beta$ to indicate whether the arc is in the path $p$ or not, i.e.
\[
\beta^k_{ap} := \begin{cases} 1, & \text{if } a \text{ is in path } p, \\ 0, & \text{otherwise}. \end{cases}
\]
Also, $\gamma^k_{ap}$ is used to denote the flow on arc $a$ induced by the demands, $h^k$, and a path $p \in \mathcal{P}^{kl}$, i.e.
\[
\gamma^k_{ap} := h^k \beta^k_{ap}, \quad a \in A, p \in \mathcal{P}^{kl}, k \in K^l, l \in L.
\]

In our Dantzig–Wolfe reformulation of (11.5), we use $\lambda^k_p$ as a convexity variable, i.e. $\lambda^k_p = 1$ if path $p \in \mathcal{P}^{kl}$ is selected for OD-pair $k \in K^l$ and 0 otherwise. This gives $x$ in terms of $\lambda$,
\[
x^k_a = \sum_{p \in \mathcal{P}^{kl}} \beta^k_{ap} \lambda^k_p, \quad a \in A, k \in K^l, l \in L.
\]
The Dantzig–Wolfe reformulation of (11.5) becomes
minimize $\zeta$
subject to
\[
\sum_{k \in K} \sum_{a \in \mathcal{P}^{kl}} \gamma^k_{ap} \lambda^k_p \leq u_a \zeta, \quad a \in A,
\]
\[
\sum_{p \in \mathcal{P}^{kl}} \lambda^k_p = 1, \quad l \in L,
\]
\[
\lambda^k_p \in B, \quad p \in \mathcal{P}^{kl}, k \in K^l, l \in L.
\]
Let \( \mu_a \leq 0 \) and \( \eta^k \), respectively, be the dual variables for the capacity constraint (11.10a) and the convexity constraint. This implies that the reduced cost of the column associated with path \( p \in P^{kl} \) is

\[
\bar{z}_p^k = -\eta^k - \sum_{a \in A} \gamma^k_{ap} \mu_a.  \tag{11.11}
\]

To find the most negative reduced cost for a given \( k \in K \), a shortest path problem with arc costs \( -h^k \mu_a \geq 0 \) can be solved. Note that it suffices to solve a single shortest problem with arc costs \( -\mu_a \geq 0 \) to find the column with the most negative reduced cost for all \( k \in K \). The reduced cost is then obtained via (11.11). Hence, the pricing problem can be solved in \( O((|L|(m + n \log n)) \) time.

Let us also consider a Dantzig–Wolfe decomposition of the tree-routing relaxation, i.e. model (11.6). Note that constraints (11.6b)-(11.6f) describe the relative flow (\( x \)-variables) uniquely determined by spanning anti-arborescences (\( y \)-variables).

Denote the collection of spanning anti-arborescences rooted at \( l \) by \( A_l \). For \( p \in A_l \), let \( \alpha^l_{ap} \) be 1 if arc \( a \in A \) is in anti-arborescence \( p \). Since an anti-arborescence induce a unique path from each node \( k \in K_l \) to \( l \), we define \( \beta^l_{ap} \) analogously with (11.7), i.e.

\[
\beta^l_{ap} := \begin{cases} 
1, & \text{if } a \text{ is in the path from } o^k \text{ to } l \text{ induced by anti-arborescence } p, \\
0, & \text{otherwise.} 
\end{cases} \tag{11.12}
\]

Also note that

\[
\sum_{k \in K^l} h^k \beta^l_{ap} \lambda_p^k = \sum_{p \in A^l} \alpha^l_{ap} \lambda_p^l, \quad a \in A, \ l \in L. \tag{11.16}
\]

As above, \( \lambda_p^l \) is a convexity variable, i.e. \( \lambda_p^l = 1 \) if anti-arborescence \( p \in A^l \) is used for destination \( l \in L \). The relations between \( x, y \) and \( \lambda \) is

\[
\begin{align*}
\bar{x}_a^k &= \sum_{p \in A^l} \beta^l_{ap} \lambda_p^k, & a \in A, & k \in K^l, & l \in L, \\
\bar{y}_a^l &= \sum_{p \in A^l} \alpha^l_{ap} \lambda_p^l, & a \in A, & l \in L. \tag{11.14}
\end{align*}
\]

Hence, the Dantzig–Wolfe reformulation becomes
11.2 A Dantzig–Wolfe Reformulation

Based on the relaxation and the models in the previous section it is straightforward to derive Dantzig–Wolfe formulations for the USPR problem. Indeed, it suffices to augment models (11.10) and (11.17) with SPR conflict constraints.
Recall from Chapter 10 that \( \mathcal{C} \) is the collection of all circuits in the independence system describing feasible routing patterns. Hence, \( \mathcal{C} \subseteq \mathcal{C} \) is a minimal SPR conflict that induces a collection of arc sets, i.e.

\[
\mathcal{C} := \{ \mathcal{C} \}_{l \in L} \subseteq A \times L.
\]

(11.20)

Given a collection of arc sets, we denote the induced set of destinations by

\[
L(C) := \{ l \in L \mid \mathcal{C} \neq \emptyset \}.
\]

(11.21)

Using this notation, the SPR circuit constraint associated with \( \mathcal{C} \subseteq \mathcal{C} \) is

\[
\sum_{l \in L(C)} \sum_{a \in C^l} y_a^l \leq \sum_{l \in L(C)} |\mathcal{C}^l| - 1.
\]

(11.22)

Hence, if either of the models (11.10) or (11.17) is augmented by the constraint (11.22) for all \( \mathcal{C} \subseteq \mathcal{C} \), it appropriately models a USPR problem.

More generally, we are also interested in stronger inequalities, in particular rank inequalities, e.g. obtained by lifting circuit inequalities. For an arbitrary collection of arc sets, \( \mathcal{C} := \{ \mathcal{C} \}_{l \in L} \subseteq A \times L \), the rank, \( r(C) \), of \( \mathcal{C} \) is defined as the maximal number of arcs in \( \mathcal{C} \) that can simultaneously be used in a unique shortest path system (USPS).

Hence, the rank inequality

\[
\sum_{l \in L} \sum_{a \in C^l} y_a^l \leq r(C), \quad \mathcal{C} := \{ \mathcal{C} \}_{l \in L},
\]

(11.23)

is valid and generalizes (11.22).

Dantzig–Wolfe reformulations of USPR problems are obtained based on these SPR rank inequalities by expressing them in the path or arborescence variables.

First consider the path formulation. We use (11.4), (11.9) to re-write (11.23) as

\[
\sum_{l \in L} \sum_{a \in C^l} y_a^l = \sum_{l \in L} \sum_{k \in K^l} \sum_{a \in \delta^+(a^k) \cap \mathcal{C}^l} x_a^k = \sum_{l \in L} \sum_{k \in K^l} \sum_{a \in \delta^+(a^k) \cap \mathcal{C}^l} x_a^k =
\]

\[
= \sum_{l \in L} \sum_{k \in K^l} \sum_{a \in \delta^+(a^k) \cap \mathcal{C}^l} \sum_{r \in P_{kl}} \beta_{ap}^k \lambda_p = \sum_{l \in L} \sum_{k \in K^l} \sum_{p \in P_{kl}} \beta_p^C \lambda_p^k,
\]

(11.24)

where

\[
\beta_p^C := \sum_{a \in \delta^+(a^k) \cap \mathcal{C}^l} \beta_{ap}^k = \max_{a \in \delta^+(a^k) \cap \mathcal{C}^l} \beta_{ap}^k,
\]

(11.25)

i.e. the path based rank inequality is

\[
\sum_{l \in L(C)} \sum_{k \in K^l} \sum_{p \in P_{kl}} \beta_p^C \lambda_p^k \leq r(C), \quad \mathcal{C} := \{ \mathcal{C} \}_{l \in L} \subseteq A \times L,
\]

(11.26)

and for an arc set based on a circuit \( \mathcal{C} \subseteq \mathcal{C} \), the rank inequality is
\[
\sum_{l \in L(C)} \sum_{k \in K^l} \sum_{p \in P^{kl}} \beta_p^C \lambda_p^k \leq \sum |C^l| - 1, \quad C := \{C^l\}_{l \in L}, \ C \in \mathcal{C}.
\] (11.27)

**Remark 11.1.** The last equality in (11.25) follows since at most one arc can emanate from a node in a simple path. Hence, \(\beta_p^C \in B\) and (11.26) is a rank inequality.

A path based Dantzig–Wolfe reformulation of USPR is obtained by adding constraint (11.26) to model (11.10) for a suitable subset of arc sets, e.g. for all circuits. For each \(C \subset A \times L\), let \(\nu_C \leq 0\) be the dual variable associated with its induced rank inequality. This implies that the reduced cost of the augmented column associated with path \(p \in P^{kl}\) becomes

\[
\hat{\zeta}_p^k = -\eta^k - \sum_{a \in A} \alpha_{ap}^k \mu_a - \sum_{C \in \mathcal{C}^l} \sum_{a \in \delta^+(o^k) \cap C^l} \bar{\beta}_p^C \nu_C = -\eta^k - \sum_{a \in A} \beta_{ap}^k \omega_a^k,
\] (11.28)

where

\[
\omega_a^k = \begin{cases} 
  h^k \mu_a + \sum_{C \in \mathcal{C}, a \in C^l} \nu_C, & \text{if } a \in \delta^+(o^k), \\
  h^k \mu_a, & \text{if } a \notin \delta^+(o^k).
\end{cases}
\] (11.29)

Hence, a column with most negative reduced cost for a given \(k \in K\) is found by solving a shortest path problem with arc costs \(-\omega_a^k \geq 0\). Thus, the pricing problem can be solved in \(\mathcal{O}(|K|(m + n \log n))\) time. We present a commonly used branching rule that does not affect the structure of this pricing problem from [22] in Section 11.3.1.

To obtain an arborescence based Dantzig–Wolfe formulation, again consider the rank inequality (11.23). From (11.15) it follows that

\[
\sum_{l \in L} \sum_{a \in C^l} y_a^l = \sum_{l \in L} \sum_{a \in C^l} \sum_{p \in A^l} \alpha_{ap}^l \lambda_p^l = \sum_{l \in L} \sum_{p \in A^l} \tilde{\alpha}_p^C \lambda_p^l
\] (11.30)

where

\[
\tilde{\alpha}_p^C = \sum_{a \in C^l} \alpha_{ap}^l,
\] (11.31)

i.e. the arborescence inequality corresponding to (11.23) is

\[
\sum_{l \in L} \sum_{p \in A^l} \tilde{\alpha}_p^C \lambda_p^l \leq r(C), \quad C \in \mathcal{C}.
\] (11.32)

When model (11.17) is augmented by (11.32) it becomes a Dantzig–Wolfe formulation of USPR based on arborescences. To see how the resulting pricing problem is affected we use the dual variable \(\bar{\nu}_C \leq 0\) for each \(C \subset A \times L\). This implies that the reduced cost of the augmented column associated with arborescence \(p \in A^l\) becomes
\[ \hat{\zeta}_p = -\bar{\eta}_l - \sum_{a \in A} \bar{\gamma}_{ap} \bar{\mu}_a - \sum_{C : (c^l) \in L} \sum_{a \in C} \alpha_{ap} \bar{\nu}_C = \]

\[ = -\bar{\eta}_l - \sum_{a \in A} \sum_{k \in K} h_k \bar{\mu}_a \bar{\beta}_{ap} - \sum_{C : (c^l) \in L} \sum_{a \in C} \bar{\nu}_C \alpha_{ap}. \]  \tag{11.33}

Setting

\[ \bar{\nu}_a := \sum_{C : (c^l) \in L \cap C \in \mathcal{C}, a \in C} \bar{\nu}_C, \]  \tag{11.34}

the most negative reduced cost for a fixed \( l \in L \) is found by solving the pricing problem,

\[
\begin{align*}
\text{minimize} & \quad -\bar{\eta}_l - \sum_{a \in A} \sum_{k \in K} h_k \bar{\mu}_a x^k_a - \sum_{a \in A} \bar{\nu}_a y^l_a \\
\text{subject to} & \quad \sum_{a \in \delta^+(i)} x^k_a - \sum_{a \in \delta^-(i)} x^k_a = b^k_i, \quad i \in N, \ k \in K^l, \quad \tag{11.35a} \\
& \quad x^k_a \leq y^l_a, \quad a \in A, \ k \in K^l, \quad \tag{11.35b} \\
& \quad x^k_a \geq 0, \quad a \in A, \ k \in K^l, \quad \tag{11.35c} \\
& \quad y^l_a \in \mathbb{B}, \quad a \in A. \quad \tag{11.35d}
\end{align*}
\]

As noted above, this problem can be solved in polynomial time since the LP relaxation is integral. An option is to solve it by a primal-dual or a dual ascent method obtained by adapting the procedure in [221] for Steiner arborescences. Yet another option is to approach it heuristically by solving a shortest path problem where the common arc cost \( \bar{\nu} \) is somehow shared. Our experiments showed that whether such a heuristic performs well (i.e., produces near-optimal solutions) depends on the ratio between the fixed and linear costs, i.e., \( \bar{\nu} \) and \( \bar{\mu} \), and not much on the size of the problem. In particular, when the fixed cost is small, in comparison to the linear cost, the heuristic solution is near-optimal. We observed that solving shortest path problems instead of the LP speeds up computations several orders of magnitudes. Since the optimal solution of the pricing problem is not necessary as long as some improving column is found, this observation is in our opinion a strong argument in favor of solving the pricing problem by a heuristic based on an algorithm for the shortest path problem.

### 11.3 Solving the Dantzig–Wolfe Formulations

The Dantzig–Wolfe reformulation typically has an exponential number of variables and are therefore solved by the column-generation approach. Further, since there are integrality constraints, the approach is typically embedded in a branch-and-bound proce-
dure which yields the B&P framework, see e.g. [23, 166]. If there is also an exponential number of constraints, the more general B&C&P framework is commonly used, see e.g. [22, 166]. There are essentially two components in a B&P scheme that has to be specified: pricing and branching. In B&C&P, cutting must also be taken into account.

### 11.3.1 Branching Rules

There are four desirable properties of branching rules in B&P and B&C&P schemes, see e.g. [166]. A branching rule should preferably:

- **B1** cut off the LP-relaxation,
- **B2** partition the solution space,
- **B3** balance the enumeration tree,
- **B4** preserve the structure of the subproblem.

It is possible to branch on path or arborescence variables, i.e. the $\lambda$-variables. However, this approach suffers from two major drawbacks since it implies that property B3 and B4 become violated, i.e. the enumeration tree becomes very unbalanced and the structure of the subproblem is completely destroyed. A common approach to avoid these issues is to branch on the original variables. In the case of paths, this rule would violate property B4. Indeed, in a branch where $x^k_a = 1$ for some arcs $a \in B$, this means that a path using all arcs $a \in B$ must be found (which is a hard problem).

We provide branching rules that satisfies most of the desirable properties below.

### Branching in Path Formulations

In path formulations, it is undesirable to branch on the original variables. The standard branching rule for unsplittable flow problems, proposed in [22], is as follows.

Take an OD-pair with fractional path variables. Find a divergence node, $i \in N$, where some arcs in $\delta^+(i)$ take fractional values. Denote these arcs by $a_1$ and $a_2$. Arbitrary partition $\delta^+(i)$ into two sets, $A_1$, and $A_2$, such that $a_1 \in A_1$, and $a_2 \in A_2$. Then, create two branches where the arcs in $A_1$, and $A_2$ respectively, are not allowed in the path for this OD-pair.

A drawback with this rule is that it does not have property B2, i.e. the solution space is not partitioned, since solutions where the path considered do not use the divergence node at all occurs in both sub-trees. Since the optimal LP solution induce the usage of the divergence node, this is likely of little importance in practice.

### Branching in Arborescence Formulations

For arborescence formulations it is possible to branch on the original variables. Indeed, for $l \in L$ and $a := (i, j) \in A$, setting $y^l_a = 0$ corresponds to removing arc $a$ and setting $y^l_a = 1$ corresponds to removing all arcs in $\delta^+(i) \setminus \{a\}$. This natural branching rule has all the desirable properties mentioned above (B3 can be challenged in the sense that there are branching schemes that achieve better balancing).
To improve the balancing of the tree, we note that an outdegree constraint gives a special ordered set (SOS). Hence, for a node $i \in N$, partition $\delta^+(i) = S_1 \cup S_2$. Then, it is possible to use SOS branching which is generally a very desirable branching rule when applicable. Observe the similarity of this rule with the path branching rule, but note that SOS branching does not lack property B2.

### 11.3.2 Pricing and Cutting

Pricing in the root node was covered above. Observe that the branching rules only fix variables at 0 which does not affect the structure of the pricing problem. Hence, the same pricing problem can be used in all nodes in the enumeration tree.

When it comes to cutting, it is straightforward to use (11.9) or (11.15) and use a separation routine in the space of original variables. We consider some separation problems in Chapter 13.

Ideally, cuts in the $\lambda$-space should be separated since they tend to be much stronger. However, characterizing such cuts and separating them can be quite hard. Also, adding such cuts typically destroys the structure of the pricing problem, see Section 11.5. A general technique to translate cuts in the space of original variables to stronger cuts in the space of $\lambda$-variables is described in the next section.

Finally we also mention the possibility of taking cutting into account via branching, i.e. by only using the ISPR problem for pruning and possibly also for propagation based on variable fixations. The advantage of this is its simplicity, it results in smaller sub-problems that are also easier to solve. The disadvantage is that the quality of bounds decrease.

### 11.4 Dantzig–Wolfe Formulation Strength

An advantage of the Dantzig–Wolfe reformulation approach is that it can give a formulation that is tighter than the original formulation. This follows since the convex hull of a substructure of the original formulation is modelled. However, in our case, the substructures correspond either to paths or arborescences and the convex hulls of these substructures are already modelled in the original formulation. Since the rank inequalities in the $\lambda$-space are direct translations of rank inequalities in the $y$-space, no additional strength is gained.

**Proposition 11.1**

Let $z_{1,LP}, z_{2,LP}, z_{3,LP}$ and $z_{4,LP}$ denote the optimal values of the LP-relaxations of (11.5), (11.10), (11.6) and (11.17), respectively. Then,

$$z_{1,LP} = z_{2,LP} \leq z_{3,LP} = z_{4,LP}.$$  \hspace{1cm} (11.36)

This seems like a negative result that indicates that the only gain in considering the Dantzig–Wolfe reformulation approach is to potentially be able to solve the LP-relaxation faster. This is in our opinion not the case. Indeed, the path and arborescence formulations have variables that encode much more information than the original variables. This fact
can be used to derive valid inequalities in the higher dimensional λ-space that are much stronger. See Example 11.1.

Below, we consider a general procedure that translates cuts in the original space to possibly stronger cuts in the extended space.

### 11.4.1 Stronger Cuts in the Extended Space

The procedure described here can be applied in any B&C&P approach. We illustrate it for the arborescence formulation, but it is straightforward to adapt it to an arbitrary problem.

Consider a valid inequality in the original formulation. For notational convenience, assume that it only involves the \( y \)-variables,

\[
\sum_{l \in L} \sum_{a \in A} \theta^l_a y^l_a \leq b, \tag{11.37}
\]

where \( \lambda \in \mathbb{Z}^{A \times L} \). In the \( \lambda \)-space, this translates via the relation (11.15) between \( y \) and \( \lambda \) to

\[
\sum_{l \in L} \sum_{p \in A^l} \theta^l_p \lambda^l_p \leq b, \tag{11.38}
\]

where

\[
\theta^l_p := \sum_{a \in p} \alpha^l_{ap} \tilde{\theta}^l_a, \quad p \in A^l, \quad l \in L, \tag{11.39}
\]

and \( \alpha^l_{ap} \) was defined on page 184 to be 1 if the arc, \( a \), is in the arborescence \( p \in A^l \).

To give the intuition behind the procedure that translates cuts in the original space to stronger cuts in the extended space, we begin by an example.

---

**Example 11.1**

**Figure 11.1:** A subpath inconsistency conflict where the inequality is not violated in the original space but in the extended space. SP-arcs to destination \( l_0 \) and \( l_1 \) are represented by solid and dashed arcs, respectively.

The conflict in Figure 11.1 induce the circuit inequality

\[
y^l_{21} + y^l_{13} + y^l_{23} + y^l_{34} \leq 3 \tag{11.40}
\]

in the original space.

Suppose that there is an arborescence \( A_1 \in A^{l_0} \) that contains the solid path 2-1-4 and an arborescence \( B_1 \in A^{l_1} \) that contains the dashed path 2-3-4. If \( \lambda^l_{10} \) and \( \lambda^l_{11} \) are the variables in the extended space associated with \( A_1 \) and \( B_1 \), respectively, then the associated inequality induced by (11.40) in the extended space is
\[ \lambda_{10}^b + \lambda_{11}^b \leq 1, \quad (11.41) \]

which states that at most one of the arborescence \( A_1 \) and \( B_1 \) can be used.

Assume that the value of the arborescence variables are 0.7, which implies that the associated arc destination variables in the original space also have value 0.7, i.e.

\[ \lambda_{00}^b = \lambda_{01}^b = 0.7, \quad \text{and} \quad y_{21}^b = y_{14}^b = y_{23}^b = y_{34}^b = 0.7. \quad (11.42) \]

Using these variables values in the inequalities (11.40) and (11.41) shows that the inequality in the original space is not violated (0.7 + 0.7 + 0.7 = 2.1), whereas the inequality in the extended space is violated (0.7 + 0.7 = 1.4 \( \not\leq 1 \)).

The advantage of the inequality in the extended space becomes more apparent when there are several small valued arborescence variables and the inequality is first lifted. Suppose that there are instead four arborescences,

\[
\begin{align*}
A_1 & \subseteq \{(1, 4), (2, 1), (3, 4)\}, \\
A_2 & \subseteq \{(1, 4), (2, 1), (3, 2)\}, \\
B_1 & \subseteq \{(1, 2), (2, 3), (3, 4)\}, \\
B_2 & \subseteq \{(1, 4), (2, 3), (3, 4)\},
\end{align*}
\quad (11.43)
\]

where \( A_1, A_2 \in A^b \) contains the solid path 2-1-4 and \( B_1, B_2 \in A^t \) contains the dashed path 2-3-4. As before \( \lambda_{10}^b, \lambda_{20}^b, \lambda_{11}^t \) and \( \lambda_{21}^t \) are the variables in the extended space associated with \( A_1, A_2, B_1 \) and \( B_2 \), respectively.

It is possible to lift the inequality (11.41) in the extended space to

\[ \lambda_{10}^b + \lambda_{20}^b + \lambda_{11}^t + \lambda_{21}^t \leq 1. \quad (11.44) \]

The validity of (11.44) can be motivated in several ways, e.g. by an ad hoc argument that at most one of the arborescence can be selected. We will later see the basic inequality (11.41) as a cover inequality and (11.44) as an extended cover inequality.

Suppose that arborescence variables in (11.41) all take the value 0.35. Then, all original variables again equal 0.7, i.e.

\[ \lambda_{10}^b = \lambda_{20}^b = \lambda_{11}^t = \lambda_{21}^t = 0.35, \quad \text{and} \quad y_{21}^b = y_{14}^b = y_{23}^b = y_{34}^b = 0.7. \quad (11.45) \]

In the original space, the situation is unchanged and (11.40) is not violated. In the extended space, the non-lifted cover inequality (11.41) is also not violated, whereas the extended cover inequality is violated.

This example illustrates that the inequalities in the extended space can be much stronger than the inequalities in the original space.
To derive stronger inequalities, we observe that the inequality (11.38) in the extended space is a (dense) knapsack constraint. When the convexity constraints in the Dantzig–Wolfe formulation are also taken into account it induces the multiple-choice knapsack polytope below. See e.g. [147, 194, 210] for a general treatment of this version of the knapsack problem. In [147] it is shown that the problem can be brought into a standard form where all coefficients are non-negative. Hence, we can assume $\theta \in \mathbb{Z}^{A \times L}_+$. 

\begin{align}
\sum_{l \in L} \sum_{p \in A^l} \theta_p^l \lambda_p^l &\leq b, \quad (11.46a) \\
\sum_{p \in A^l} \lambda_p^l &= 1, \quad l \in L, \quad (11.46b) \\
\lambda_p^l &\in \mathbb{B}, \quad p \in A^l. \quad (11.46c)
\end{align}

Stronger inequalities can be obtained directly from (11.46) via (extended) cover inequalities. Taking the convexity constraints into account significantly improves the quality of the resulting cuts and also helps in the analysis of the lifting. To derive minimal covers and to be able to easily lift the associated constraints we define for each $l \in L$ and $n \in \mathbb{Z}_+$ the set of arborescences, $A^l_n \subseteq A^l$ that have $\theta_p^l \geq n$, i.e.

$$A^l_n := \{ p \in A^l \mid \theta_p^l \geq n \}. \quad (11.47)$$

Based on the arborescence sets, $A^l_n$, we derive minimal covers for (11.46a). Further, based on the definition of $A^l_n$, the resulting cover inequalities will be easy to lift.

Remark 11.2. If the original inequality (11.37) is a circuit inequality, then the coefficient $\theta_p^l$ in the inequality (11.38) in the extended space counts the number of arcs that are in the arborescence $p \in A^l$. Hence, $p \in A^l_n$ if $p$ contains at least $n$ of the arcs associated with $l$ in the original circuit inequality.

We use $\{n^l\} := \{n^{l_1}, \ldots, n^{l_{|l|}}\} \in \mathbb{Z}_+^L$ to denote a set of non-negative integers, where each element $n^l \in \{n^l\}$ is associated with some $l \in L$. The set of destinations associated with strictly positive integers in $\{n^l\}$ are denoted by $L_+$. A $b$-cover partition is defined as a set of non-negative integers whose sum exceeds $b$.

**Definition 11.1**

Take $\{n^l\}$. Then, $\{n^l\}$ is a $b$-cover partition if

$$\sum_{l \in L} n^l \geq b + 1. \quad (11.48)$$

Further, if equality holds, we say that $\{n^l\}$ is a minimal $b$-cover partition.

**Definition 11.2**

Take $\{n^l\}$. Then, $L_+ := \{ l \in L \mid n^l > 0 \}$.

**Proposition 11.2**

Let $\{n^l\}$ be a $b$-cover partition. For each $l \in L_+$ take $p^l \in A^l_{n^l}$ arbitrarily. Then,
\[ P := \{ p^l \}_{l \in L^+} \]  

induces a cover of (11.46a).

**Proof:** The indices in \( P \) induces a cover since

\[
\sum_{p^l \in P} \theta^l_p \geq \sum_{l \in L} n^l \geq b + 1. \tag{11.50}
\]

**Corollary 11.1**

Let \( \{ n^l \} \) be a \( b \)-cover partition and define \( P \) as in (11.2). Then,

\[
\sum_{p^l \in P} \lambda^l_p \leq |P| - 1, \tag{11.51}
\]

is a valid inequality for (11.46).

Using the common concept of an extended cover, or equivalently, by sequentially lifting the inequality in (11.51), we obtain a much stronger inequality.

**Corollary 11.2**

Let \( \{ n^l \} \) be a \( b \)-cover partition. Then,

\[
\sum_{l \in L^+} \sum_{p \in \mathcal{A}^{l^+}_{n^l}} \lambda^l_p \leq |L^+| - 1, \tag{11.52}
\]

is a valid inequality for (11.46).

**Proof:** Since for each \( l \in L^+ \) and each \( p \in \mathcal{A}^{l^+}_{n^l} \), the coefficient \( \theta^l_p \geq n^l \) it follows that inequality (11.52) is an extended cover inequality.

It should be noted that inequality (11.52) is a rank inequality for a set that typically has very large cardinality and low rank. This indicates that it is quite strong and in particular (potentially) much stronger than (11.38).

The special case where the inequality in the original space, i.e. (11.37), is a cover inequality is very relevant for us (and presumably in many other cases) since for instance all circuit inequalities fall into this category. In particular, this implies that \( \theta \in B^{A \times L} \).

When arborescences are considered many such constraints are associated with a minimal conflict \( C \in \mathcal{C} \) where \( \mathcal{C} := \{ C^l \}_{l \in L} \). Then, \( \theta^l_p = |p \cap C^l| \). See Remark 11.2.

**Proposition 11.3**

Let \( \mathcal{C} := \{ C^l \}_{l \in L} \) be a conflict associated with the minimal cover inequality (11.37) is in the original space and with (11.38) in the \( \lambda \)-space. Then, \( \{ |C^l| \}_{l \in L(C)} \) is a \( b \)-cover partition and the inequality

\[
\sum_{l \in L(C)} \sum_{p \in \mathcal{A}^{l^+}_{|C^l|}} \lambda^l_p \leq |L(C)| - 1, \tag{11.53}
\]

is valid for (11.46). Further, it induces a facet for the polytope induced by (11.46).
Proof: Since $C$ induces a cover, $\{C^j\}_{j \in L(C)}$ is a b-cover partition and validity follows from Corollary 11.2. To show that the inequality induces a facet use a maximal lifting argument, see e.g. [180].

First observe that (11.53) induces a facet of the restricted polytope where variables not in the inequality (11.53) are set to 0. Then, considered variables not in (11.53) for lifting. First take a variable $\lambda_p^l$ where $l \in L \setminus L(C)$ and $p \in A^l$. Setting such a variable to 1 is not in conflict with any variable already in the inequality, i.e. it is feasible w.r.t. (11.46) to set $\lambda_p^l = 1$ and any $|L(C)| - 1$ variables in (11.53) to 1. Hence, the maximal lifting coefficient of $\lambda_p^l$ is 0 for $l \in L \setminus L(C)$ and $p \in A^l$. Instead consider a variable $\lambda_p^l$ where $l \in L(C)$ and $p \notin A^l(C)$. This variable has $\theta_p^l < |C^l|$, therefore we can choose $|L(C)| - 1$ variables to set to 1 from arborescences associated with the remaining destinations, i.e. $L(C) \setminus \{l\}$. Hence, this maximal lifting coefficient must also be 0.

Remark 11.3. In the case where $C \in C$ is a circuit, the facet defining inequality (11.53) is the only non-trivial facet of (11.46).

Example 11.2

Figure 11.2: A valid cycle conflict where the inequality is not violated in the extended path-space but in the extended arborescence-space. SP-arcs to destination $l_0$ and $l_1$ are represented by solid and dashed arcs, respectively.

We compare the inequalities induced by the valid cycle in Figure 11.2 for the path and arborescence formulations. Throughout, entities without a bar, e.g. $A_1$, refers to the path case and entities with a bar, e.g. $\overline{A}_1$, refers to the arborescence case. Consider some paths $A_1, A_2, B_1, B_2$ and some arborescences $\overline{A}_1, \overline{B}_1$ that contain the following arcs,

$$A_1 := \{(1, 4), (4, l_0)\},$$
$$A_2 := \{(3, 2), (2, l_0)\},$$
$$B_1 := \{(1, 2), (2, l_1)\},$$
$$B_2 := \{(3, 4), (4, l_1)\},$$
$$\overline{A}_1 := A_1 \cup A_2 = \{(1, 4), (4, l_0), (3, 2), (2, l_0)\},$$
$$\overline{B}_1 := B_1 \cup B_2 = \{(1, 2), (2, l_1), (3, 4), (4, l_1)\}.

In the extended space for paths, all paths contains one conflict arc, hence

$$A^{l_0} = \{A_1, A_2\}, \quad A^{l_1} = \{B_1, B_2\}.

This implies, via Corollary 11.2 that the inequality

$$\lambda^{l_0}_1 + \lambda^{l_0}_2 + \lambda^{l_1}_1 + \lambda^{l_1}_2 \leq 3,$$

is valid, i.e. not all four paths associated with $A_1, A_2, B_1, B_2$ can be chosen.
Instead consider the extended space for arborescences, where both $\bar{A}_1$ and $\bar{B}_1$ contain two conflict arcs, hence

$$\bar{A}^{lo} = \{\bar{A}_1\}, \quad \text{and} \quad \bar{A}^{hi} = \{\bar{B}_1\},$$

(11.57)

and Corollary 11.2 implies that the inequality

$$\lambda_{lo}^l + \lambda_{hi}^l \leq 1,$$

(11.58)

is valid, i.e. not all both arborescences $\bar{A}_1$ and $\bar{B}_1$ can be chosen.

It is straightforward to verify that a solution in the extended path space given by $l_0^l = l_2^l = 0$, $l_1^l = l_2^l = 0$ corresponds to a solution in the extended arborescences space where $\lambda_{lo}^l = \lambda_{hi}^l = 0.7$. In this case, the above inequality is violated in the arborescence case and not violated in the path case.

Example 11.1 and 11.2 illustrate that the inequalities in the extended space can be much stronger than the inequalities in the original space. Also, when conflicts involve more than two path segments (or more than two destinations) the rank, i.e. right hand side, gives a significant difference between paths and arborescences. In short, the additional information encoded in the variables in the extended spaces can be very useful.

Every $b$-cover partition, $(n^l)_{l \in L}$, yields the reasonably strong valid inequality (11.52) via Corollary 11.2. A very relevant question is how to find a good $b$-cover partition. We derive an efficient separation algorithm that given an inequality in the original space finds the $b$-cover partition that corresponds to a most violated inequality in the $\lambda$-space.

Let $\bar{\lambda}$ be a fractional solution to the B&C&P master problem. Define

$$s(n, l) := \sum_{p \in \bar{A}_n^l} \lambda_{n}^p, \quad n \in \mathbb{Z}_+, \ l \in L,$$

(11.59)

and

$$s(n, l) := \begin{cases} s(n, l) - 1, & n \geq 1, \\ 0, & n = 0 \end{cases}, \quad n \in \mathbb{Z}_+, \ l \in L,$$

(11.60)

i.e. $s(n, l)$ is the sum over all arborescence variables that have $\theta \geq n$ and $s(n, l)$ is an auxiliary function. If the original inequality is a rank inequality, $s(n, l)$ evaluates the sum over variables associated with arborescences that contain at least $n$ of the conflict arcs for destination $l$.

Remark 11.4. The function $s(n, l)$ is bounded above by 0 and non-increasing for $n \geq 1$.

To obtain a most violated inequality, we must find a $b$-cover partition that maximizes the sum of the induced $s(n, l)$ minus the size of the partition. This can be accomplished by solving the following knapsack problem.
maximize \( \sum_{l \in L} s(n^l, l) \)
subject to
\[
\sum_{l \in L} n^l \geq b + 1, \quad (11.61a)
\]
\[
n^l \in \mathbb{Z}_+, \quad l \in L. \quad (11.61b)
\]

Problem (11.61) can be solved by dynamic programming. Observe that such an algorithm is polynomial in the special case where the original inequality is a rank inequality.

**Proposition 11.4**
Let \( z^* \) denote the optimal value of (11.61). There is a minimal \( b \)-cover partition such that the inequality (11.52) is violated if and only if \( z^* > -1 \). Further, the violation is \( 1 + z^* \).

**Proof:** A feasible solution \( \{n^l\} \) is a \( b \)-cover partition and
\[
\sum_{l \in L} \sum_{p \in A^l_{n^l}} \bar{x}^l_p = \sum_{l \in L} s(n^l, l). \quad (11.62)
\]
Hence, the quantity
\[
\sum_{l \in L} \sum_{p \in A^l_{n^l}} \bar{x}^l_p - |L_+| + 1 = \sum_{l \in L} \left( \sum_{p \in A^l_{n^l}} \bar{x}^l_p - 1 \right) + 1 = \sum_{l \in L} s(n^l, l) + 1 \quad (11.63)
\]
measures the violation and there is a violated inequality if and only if \( z^* > -1 \). \( \square \)

We summarize the properties of our general procedure to translate cuts in the original space to cuts in the extended space.

Its main features are that it is (1) very general, and (2) does not rely on knowing anything about the structure of the problem, and (3) comes with an easily solvable separation problem. The main drawback of the approach is actually due to its generality, since this means that it cannot take problem specific structure into account. Also, the procedure requires an inequality in the original space. It may be hard to find inequalities that induce good inequalities in the extended space; we may even have to consider inequalities that are non-violated in the original space, see Example 11.1. Due to the last negative remark, we believe that the procedure is likely most useful as a way of strengthening some already violated inequalities in the original space.

### 11.5 Some Practical Dantzig–Wolfe Considerations

In this final section, some practical aspects that usually arise in the context of solving column generation problems are discussed. Some very important issues/remedies are:
initialization, warm starting, over-generation of columns, tailing-off, stabilization, using interior dual points, generation and selection of cuts, etc., see e.g. [166]. We mention some particularities of our pricing problem that can be used to handle some of these issues.

**Warm starting.** By the nature of the problem it is easy to produce feasible solutions. Indeed, it suffices to search in the weight space. In practice, it seems like this approach also produces high quality solutions fast, see the references in Section 2.5.1 on page 28. Hence, heuristics can be used to obtain a set of initial solutions that can be decomposed into arborescences which can be used to initialize column generation. Note that it is also possible to generate more starting columns from such columns in the same manner as in the over-generation procedure below.

**Over-generation.** In cut-generation, i.e. the "dual" of column-generation, it is usually a poor strategy to only produce one cut. Instead, several (preferably well-coordinated, see e.g. [2, 11, 19]) cuts are generated in rounds. In column generation, the analogue of generating many columns is referred to as over-generation. The coordination aspect can be translated as generating near-orthogonal columns.

In our setting, it is easy to generate several near-optimal columns from an optimal column. Indeed, just insert a new arc and remove the appropriate arc in the arborescence. The effect of the modification can easily be deduced in constant time from the distances to the root in the arborescence w.r.t. \( \bar{\mu} \), the fixed costs and the demands. Hence, it is easy to stay near-optimal when desired.

Let us also consider the consequences of using stronger inequalities based on arborescence variables. Unfortunately, this comes at a possibly high price; it destroys the structure of the pricing problem. For simplicity, assume that the strong inequality is a rank inequality obtained from a minimal cover \( C := \{ C^l \}_{l \in L} \) as in (11.53), i.e.

\[
\sum_{l \in L(C)} \sum_{p \in A^l[C^l]} \lambda^l_p \leq |L(C)| - 1,
\]

with dual variable \( \phi_C \leq 0 \). To appropriately take this constraint into account when an arborescence variable is priced in, the expression for the reduced cost must be altered. Indeed, if the new arborescence variable, \( p \) say, contains all arcs in \( C^l \), i.e. \( p \in A^l[C^l] \), then the reduced cost should be increased by \( -\phi_C \geq 0 \).

The complication arising from these strong inequalities can be modelled in the pricing problem as follows. Use a binary variable, \( z_C \), say, that indicates (for each inequality) if all arcs in \( C^l \) are included in the arborescence or not. This affects the objective by the addition of the term \( -\phi_C z_C \). To make sure that \( z_C \) gets the correct value, the following constraint can be used,

\[
\sum_{a \in C^l} y_a^l \leq |C^l| - 1 + z_C.
\]

We are not aware of a satisfactory way of dealing with these new constraints. We mention two possible heuristic approaches: (1) do not take lifting into account in the
pricing problem, but correct the coefficients in the master problem, and (2) take lifting into account to some extent in the pricing problem by solving it approximately via Lagrangian relaxation (possibly only for a few iterations). Note that both approaches give lower bounds on the smallest reduced cost for the correct pricing problem which is a desirable property in the Dantzig–Wolfe setting.
A common assumption when shortest path routing (SPR) is used in IP networks and shortest paths are not required to be unique, is that traffic is divided "evenly" on all shortest paths. More precisely, if there at some node are several emanating arcs that all belong to some shortest path to the same destination, then the outgoing traffic from that node is divided evenly on all emanating arcs in some shortest path to that destination. This is referred to as equal cost multi-path (ECMP) splitting. Observe that ECMP does not necessarily imply that all paths carry the same amount of traffic, only that the non-zero flow on emanating arcs are equal if the destinations are equal, see Example 2.2.

In this chapter we are interested in how to model and handle the ECMP aspect. To this end, it suffices to consider relaxations of optimization problems in IP networks where the SPR aspect is to some extent ignored, but splitting is taken into account. We deal with the ECMP issue in three ways.

First, we consider how to model ECMP splitting at a single node. In particular, we derive a polynomial size extended formulation that projects down to the convex hull of the single node ECMP splitting polytope.

Second, we consider a relaxation similar to the one in Chapter 11 where the SPR aspect is neglected but the ECMP splitting is taken into account. This implies that the traffic is routed along ingraphs instead of intrees. The relaxation is approached by a Dantzig–Wolfe discretization reformulation. We show how to efficiently solve the pricing problem also when some variables in the original formulation are fixed.

Finally, we study the acyclic ingraph problem of optimizing over $\mathcal{I}$ from Chapter 10. In particular, several classes of facets are derived for $\text{conv } \mathcal{I}$. Since $\mathcal{Y} \subseteq \mathcal{I}$, these facets carry over as valid inequalities for $\mathcal{Y}$.

Outline The modelling of splitting at a single node is considered in Section 12.1. We introduce the ECMP relaxation in Section 12.2 and show how to solve the pricing problem in Section 12.3. Finally, in Section 12.4, the acyclic ingraph problem and the associated polytope is analyzed.
12.1 Modelling Splitting at a Single Node

As usual, \( G = (N, A) \) is a strongly connected digraph and \( K \) a set of OD-pairs. Suppose that each OD-pair, \( k \in K \), has a flow demand, \( h^k > 0 \), to be sent from the origin, \( o^k \), to the destination, \( d^k \). For each, \( k \in K \), the fraction of \( h^k \) sent on arc \( a \in A \) is modelled via the flow variable \( x^k_a \). We also use \( Z_a^b := \{a, b\} \cap \mathbb{Z} = \{a, \ldots, b\} \) to denote the integers from \( a \in \mathbb{Z} \) to \( b \in \mathbb{Z} \) and \( n := |\delta^+(i)| \) for the number of arcs emanating from node \( i \).

Since we focus on splitting at a single node, we fix the node \( i \in N \) and the OD-pair \( k \in K \) with destination \( l = d^k \) throughout this section. We assume that the outflow from node \( i \) is 1, this holds for instance in the important case where \( i = o^k \), i.e.

\[
\sum_{a \in \delta^+(i)} x^k_a = 1. \tag{12.1}
\]

First, we will consider the case where the splitting decision depends on the OD-pair. Later, we restrict splitting to depend only on the destination of the OD-pair. Let \( y^k_a \) be a binary variable used to indicate whether \( x^k_a \) is strictly positive or not. (In the SPR context this corresponds to the arc \( a \) being on a shortest path from \( o^k \) to \( d^k \).) The flow conservation and the indicator relation between \( x^k_a \) and \( y^k_a \) yields

\[
\sum_{a \in \delta^+(i)} x^k_a - \sum_{a \in \delta^-(i)} x^k_a = b^k_i, \tag{12.2a}
\]

\[
0 \leq x^k_a \leq y^k_a, \quad a \in \delta^+(i), \tag{12.2b}
\]

\[
y^k_a \in \mathbb{B}, \quad a \in \delta^+(i), \tag{12.2c}
\]

where \( b^k_i \) is again the node balance, see e.g. (9.5) on page 131.

The additional splitting requirement imposed by the ECMP principle is naturally described as follows. Let \( v^k_i \) be the value of the common outflow on arcs emanating from \( i \) that carries strictly positive flow. Since flow must be divided evenly and the outflow is 1, the common outflow value, \( v^k_i \), is obtained from the flow indicator variables, i.e.

\[
v^k_i = \frac{1}{\sum_{a \in \delta^+(i)} y^k_a}. \tag{12.3}
\]

The arc flow value, \( x^k_a \), is determined as

\[
x^k_a = y^k_a v^k_i = \begin{cases} v^k_i, & \text{if } y^k_a = 1, \\ 0, & \text{if } y^k_a = 0, \end{cases} \quad a \in \delta^+(i). \tag{12.4}
\]

The non-linear relations in (12.3) and (12.4) are not suitable in a MILP model. The following big-\( M \) linearization is commonly used, see e.g. [50, 186, 191],

\[
0 \leq v^k_i - x^k_a \leq M(1 - y^k_a), \quad a \in \delta^+(i). \tag{12.5}
\]

Observe that \( y^k_a = 1 \) in (12.5) forces \( v^k_i = x^k_a \) and \( y^k_a = 0 \) in (12.2b) forces \( x^k_a = 0 \).
Remark 12.1. Note that (12.3) is based on the assumption that the outflow is 1, i.e. (12.1), whereas (12.4) and (12.5) hold in general.

An alternative to (12.5) is to project out the outflow variable \( v^k_a \), to obtain

\[
x^k_{a'} - x^k_{a''} \leq M(1 - y^k_{a''}), \quad a', a'' \in \delta^+(i), i \in N, l \in N.
\]

(12.6)

In both cases, \( M \) can be set to 1 since \( x^k_{a'} \) models the fraction of the demand sent on arc \( a \).

**Proposition 12.1**

Define the polytopes \( P_{xy}^{LP} \) and \( P_{xyv}^{LP} \) as

\[
P_{xy}^{LP} := \{(x, y) \mid (12.2a), (12.2b) \text{ and } (12.6) \text{ are satisfied} \}, \quad \text{and}
\]

\[
P_{xyv}^{LP} := \{(x, y, v) \mid (12.2a), (12.2b) \text{ and } (12.5) \text{ are satisfied} \},
\]

respectively, then \( P_{xy}^{LP} = \text{proj}_{xy}(P_{xyv}^{LP}) \), where

\[
\text{proj}_{xy}(P) = \{(x, y) \mid \text{there exists } w \text{ such that } (x, y, w) \in P \},
\]

i.e. \( \text{proj}_{xy}(\cdot) \) is the projection onto the \((x, y)\)-variables.

**Proof:** Take \((x, y) \in P_{xy}^{LP}\) and set \( v^k_i = \max_{a \in \delta^+(i)} x^k_a \). Then, \( 0 \leq v^k_i - x^k_a \) for all \( a \in \delta^+(i) \).

Further, \( v^k_i - x^k_a \leq M(1 - y^k_a) \) holds since for all \( a \in \delta^+(i) \):

\[
v^k_i - x^k_a - M(1 - y^k_a) \leq \max_{a' \in \delta^+(i)} \left( v^k_a - x^k_a - M(1 - y^k_a) \right) =
\]

\[
= \max_{a', a'' \in \delta^+(i)} \left( x^k_{a'} - x^k_{a''} - M(1 - y^k_{a''}) \right) \leq 0.
\]

(12.10)

(12.11)

Hence, \( P_{xy}^{LP} \subseteq \text{proj}_{xy}(P_{xyv}^{LP}) \).

For the other direction, take \((x, y, v) \in P_{xyv}^{LP} \), i.e. \((x, y) \in \text{proj}_{xy}(P_{xyv}^{LP}) \). Then,

\[
x^k_{a'} - x^k_{a''} \leq v^k_i - x^k_{a''} \leq M(1 - y^k_{a''}).
\]

(12.12)

Hence, \((x, y) \in P_{xy}^{LP}\). Therefore, \( \text{proj}_{xy}(P_{xyv}^{LP}) \subseteq P_{xy}^{LP} \) and \( P_{xy}^{LP} = \text{proj}_{xy}(P_{xyv}^{LP}) \).

Our primary interest lies in the SPR case, or stronger relaxations of the SPR case. Therefore, we will only consider the restriction of ECMP where all OD-pairs with the same destination use the same splitting. (This holds in the SPR context.)

To this end, a binary indicator variable \( y^k_a \) is set to 1 if \( x^k_a \) is strictly positive for some \( k \) with \( d^k = l \), or equivalently,

\[
y^k_a := \max_{k \in \mathcal{K}^l} y^k_a.
\]

(12.13)

It is straightforward to verify that the theory above is valid also when \( y^k_a \) is replaced by \( y^k_a \). In the following, we focus on the points satisfying
\[ \sum_{a \in \delta^{+}(i)} x_{a}^{k} - \sum_{a \in \delta^{-}(i)} x_{a}^{k} = b_{l}^{k}, \]  
\[ 0 \leq x_{a}^{k} \leq y_{a}^{l}, \quad a \in \delta^{+}(i), \]  
\[ 0 \leq v_{i}^{k} - x_{a}^{k} \leq 1 - y_{a}^{l}, \quad a \in \delta^{+}(i), \]  
\[ y_{a} \in \mathbb{R}, \quad a \in \delta^{+}(i). \]

Denote the convex hull of the solutions to (12.14) by \( P^{C} \), i.e.

\[ P^{C} := \text{conv} \left\{ (x, y, v) \in \mathbb{R}^{\delta^{+}(i)} \times \mathbb{R}^{\delta^{+}(i)} \times \mathbb{R} \mid (x, y, v) \text{ solves } (12.14) \right\}. \]  

Formulation (12.14) is weak, observe for instance that for any arc subset, \( J \subseteq \delta^{+}(i) \), points satisfying

\[ x_{a}^{k} = y_{a}^{l} = v_{i}^{k} = \frac{1}{|J|}, \quad a \in J, \]  
\[ x_{a}^{k} = y_{a}^{l} = 0, \quad a \not\in J, \]

can be in a feasible solution to the LP-relaxation of (12.14). Observe that setting \( v_{i}^{k} = 1 \) gives a point in \( P^{C} \). Next, we strengthen model (12.14).

### 12.1.1 Stronger Single Node Formulations

Our preliminary polyhedral studies showed that the facial structure of (12.14) is rather complex. In particular, many (classes of) facets have dense support and large coefficients. Some classes of facets, along with polynomial time separation algorithms, are presented in \[186\] for the version of model (12.14) based on \( y_{a}^{l} \) instead of \( y_{a}^{l} \).

We will not focus on the polyhedral structure of the convex hull, \( P^{C} \), directly. Instead, we propose some extended formulations by introducing splitting variables to obtain stronger LP-relaxations. Via a particular choice of variables we are able to obtain an extended formulation that is polynomial in size and projects down to \( P^{C} \). This formulation is closely connected to our polynomial time algorithm to optimize over \( P^{C} \).

An advantage of introducing splitting variables is that the additional information can be utilized in other subsystems of the original problem. We believe that this can be very valuable; in particular, it is possible to use information about traffic splitting to obtain stronger inequalities when prohibiting SPR conflicts. Also, we expected that branching early on such variables is very fruitful since they represent major decisions.

The first extended formulation has a single binary splitting variable \( s_{i}^{l} \) that is set to 1 if traffic is split at node \( i \) to destination \( l \) and 0 otherwise. To incorporate this new variable into model (12.14) we augment it by,
12.1 Modelling Splitting at a Single Node

\[ \sum_{a \in \delta^+ (i)} y_{a}^l \geq 1 + s_i^l, \quad (12.17a) \]
\[ \sum_{a \in \delta^+ (i)} y_{a}^l \leq 1 + (\bar{n} - 1) s_i^l, \quad (12.17b) \]
\[ v_k^i + s_i^l \geq 1, \quad (12.17c) \]
\[ v_k^i \leq 1 - \frac{1}{2} s_i^l, \quad (12.17d) \]
\[ s_i^l \in \mathbb{B}. \quad (12.17e) \]

The first two constraints connect \( s \) and \( y \), i.e. if two or more arcs are used, \( s_i^l \) must be 1 and if \( s_i^l = 0 \) only a single arc can be used. The last two constraints are not necessary, but strengthen the model by connecting \( s \) and \( v \).

Observe that augmenting model (12.14) by the constraints (12.17) increases the model size very little, but yields a strictly stronger LP-relaxation. For example, it is easy to see that the point in (12.16) is prohibited (after projecting out \( s_i^l \)). In our second extended formulation several splitting variables are used. For each \( m \in \mathbb{Z}_n \), the binary variable \( q_{im}^l \), is set to 1 if traffic from node \( i \) to destination \( l \) is split over \( m \) arcs and 0 otherwise. To ensure that these variables take the correct values, model (12.14) is augmented by the following constraints,

\[ \sum_{m=1}^{n} q_{im}^l = 1, \quad (12.18a) \]
\[ \sum_{a \in \delta^+ (i)} y_{a}^l = \sum_{m=1}^{n} m q_{im}^l, \quad (12.18b) \]
\[ \sum_{m=1}^{n} \frac{1}{m} q_{im}^l = v_k^i, \quad (12.18c) \]
\[ q_{im}^l \in \mathbb{B}, \quad m = 1, \ldots, n. \quad (12.18d) \]

The first constraints forces exactly one \( q \)-variable to be selected, i.e. it specifies how many arcs, \( m \), that should be used. Then, the next constraint makes sure that \( m \) arcs are used and the last constraint defines \( v_k^i \) from \( m \).

Note that this formulation is stronger than using a single splitting variable, e.g. already for two arcs the point

\[ x_{a}^k = v_k^i = 0.5, \quad a \in \delta^+ (i), \]
\[ y_{a}^l = 0.75, \quad a \in \delta^+ (i), \quad (12.19) \]

is in (the projection of the LP-relaxation of) the polytope induced by (12.14) and (12.17) while it is not in (the projection of the LP-relaxation of) the polytope induced by (12.14) and (12.18). This yields the following proposition.
Proposition 12.2
The relations between the LP-relaxations of the ECMP polytopes,
\[ P_{xy} := \{ (x, y, v) \mid (12.14) \text{ is satisfied} \}, \]
\[ P_{xyv} := \{ (x, y, v, s) \mid (12.14) \text{ and (12.17) are satisfied} \}, \]
\[ P_{xysr} := \{ (x, y, v, r) \mid (12.14) \text{ and (12.18) are satisfied} \}, \]
is
\[ \text{proj}_{xy}(P^C) \subset \text{proj}_{xy}(P^{LP}_{xy}) \subset \text{proj}_{xy}(P^{LP}_{xyv}) \subset \text{proj}_{xy}(P^{LP}_{xyvq}) = P^{LP}_{xy}, \] (12.21)
where \( \text{proj}_{xy}(\cdot) \) denotes the projection onto the \((x, y)\)-variables.

Proof: The inclusions are trivial and strictness follows from the notes above about the points in (12.16) and (12.19).

Using the facet enumeration software cdd+ based on [114] we found the facets of the polytopes considered above for small values of \( \bar{n} \). We conclude that besides giving stronger formulations, the splitting variables also reduce the number of facets required to describe the convex hulls, see Table 12.1. However, the facets for the extended formulations remain complex and have dense supports and large coefficients.

<table>
<thead>
<tr>
<th>( \bar{n} )</th>
<th>#Points</th>
<th>( P_{xy} )</th>
<th>( P_{xyv} )</th>
<th>( P_{xyvs} )</th>
<th>( P_{xyvq} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>13</td>
<td>8</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>85</td>
<td>43</td>
<td>25</td>
<td>28</td>
</tr>
<tr>
<td>5</td>
<td>31</td>
<td>576</td>
<td>329</td>
<td>171</td>
<td>109</td>
</tr>
<tr>
<td>6</td>
<td>63</td>
<td>5689</td>
<td>5105</td>
<td>2928</td>
<td>1190</td>
</tr>
<tr>
<td>7</td>
<td>127</td>
<td>85702</td>
<td>133293</td>
<td>92849</td>
<td>25575</td>
</tr>
</tbody>
</table>

Observe from Table 12.1 that the number of extremal points in the splitting polytopes is quite small; it is "only" \( 2^{\bar{n}} - 1 \). In practice, graphs correspond to real-world networks that are close to planar and have small average and maximal degrees, hence this number is often reasonably small.

This observation suggests that if it is important to have a good description of \( P^C \), then it may even be a viable option to use an inner representation directly, i.e. to describe the convex hull as a convex combination of its extreme points by enumerating them. Later, we show how to optimize over \( P^C \) efficiently. This gives an algorithm that can be used to price in extreme points implicitly (all natural branching decisions can easily be handled).

We describe the enumeration approach briefly below. Denote the polytope under consideration by \( P \). For \( j = 1, \ldots, 2^n - 1 \), let \( p^{(j)} \) be the extreme point of \( P \) induced by the \( y \) corresponding to the binary representation of \( j \). See Table 12.2 for a list of the extreme points for \( \bar{n} = 3 \). Using variables \( \lambda_j \) for \( j = 1, \ldots, 2^n - 1 \), a point \( p \in P \) is expressed as,
\[ P \ni p = \left( \begin{array}{c} y \\ x \\ v_j^k \\ s_j^l \\ q_j^l \end{array} \right) = \sum_{j=1}^{2^n-1} \lambda_j p^{(j)}, \quad \text{where } 0 \leq \lambda_j \leq 1, \ j = 1, \ldots, 2^n - 1. \quad (12.22) \]

It should be clear from the context which elements of the point \( p \) that are used, i.e. by the domain of the polytopes \( \text{conv } P_{xy}, \text{conv } P_{xyv}, \text{conv } P_{xyvs} \) or \( \text{conv } P_{xyvq} \).

A computational study in [186] for a multicommodity flow problem with ECMP splitting showed that single node splitting inequalities based on a polytope similar to \( \text{conv } P_{xyv} \) are efficient. In particular, such inequalities reduce the number of nodes in the enumeration tree and the computational time. On the other hand, inequalities based on two node systems reduce the former, but not as much the latter. A drawback with the experiments is that the instances are based on random graphs not similar to real-world networks.

Based on this, it seems important to derive a good description of \( P^C \) and to be able to optimize over \( P^C \) efficiently. We solve these two problems next.

### 12.1.2 The Single Node Splitting Polytope

Given a combinatorial optimization problem that admits a polynomial time algorithm it is quite common that we can also find a a good description of the associated polyhedron. Sometimes, the polyhedron can even be described by a polynomial number of constraints, possibly in an extended space. We show below that this holds for the problem of optimizing over the single node splitting polytope, i.e. solving

\[
\begin{align*}
\text{minimize} & \quad c'_x x + c'_y y \\
\text{subject to} & \quad (x, y) \in \text{proj}_{xy}(P^C).
\end{align*}
\]

Based on the following crucial observation it is straightforward to derive an algorithm for solving problem (12.23). If the number of emanating arcs that we choose to use is
Given arc costs \( c_x, c_y \in \mathbb{Q}^{\delta^+(i)} \), find the optimal value \( w^* \) of problem (12.23).

1. For \( p = 1 \) to \( \tilde{n} \) do the following.
   (a) Calculate the vector \( \vec{c}^p := \frac{1}{p}c_x + c_y \).
   (b) Find the \( p \) smallest values of \( \vec{c}^p \) and denote their sum by \( w^*(p) \).

2. Set \( w^* = \min_{p \in \mathbb{Z}_n^+} w^*(p) \).

The subproblem of finding the \( p \) smallest values of the vector \( \vec{c}^p \) in Algorithm 12.1.1 can be solved by sorting. Then, the total complexity of the algorithm is \( \mathcal{O}(\tilde{n}^2 \log \tilde{n}) \).

For arbitrary \( p \in \mathbb{Z}_n^+ \), it is possible to improve the algorithm for the subproblem. W.l.o.g. assume \( p \leq \tilde{n}/2 \). By symmetry, we can find the largest \( \tilde{n} - p \) values instead if \( p > \tilde{n}/2 \). To improve the running time, we must not sort the values. Instead, we find the median value and divide the values into two sets, say \( S_S(p) \) and \( S_L(p) \), that contain all values smaller and larger than the median, respectively. Since \( p \leq \tilde{n}/2 \), all values in \( S_L(p) \) can be discarded and the \( p \) smallest values are in \( S_S(p) \). These are found by recursion.

The median can be found in linear time, see e.g. [53]. Therefore, the recurrence relation for the running time is

\[
T(\tilde{n}) \leq T(\tilde{n}/2) + \mathcal{O}(\tilde{n}) .
\]

Hence, the subproblem can be solved in linear time and the overall time complexity of the algorithm is \( \mathcal{O}(\tilde{n}^2) \).

We have two remarks on this algorithm.

**Remark 12.2.** If \( c_x, c_y \geq 0 \), then there is an optimal solution with \( p = 1 \) which implies that problem (12.23) can be solved in linear, i.e. \( \mathcal{O}(\tilde{n}) \), time. This holds also when some \( y \)-variables are fixed, see Algorithm 12.3.2.

**Remark 12.3.** We believe that an algorithm based on binary search in \( p \) can be used to solve the problem with high probability. This is based on the empirical observation that \( w^*(p) \) is often quasi-convex. The running time of such an binary search algorithm is \( \mathcal{O}(\tilde{n} \log \tilde{n}) \).

Algorithm 12.1.1 is based on dividing problem (12.23) into \( \tilde{n} \) subproblems and then for each subproblem selecting the \( p \) smallest values. This leads quite naturally to the following choice of variable in an integer programming formulation for problem (12.23); for each arc \( a \in \delta^+(i) \) and each \( p \in \mathbb{Z}_n^+ \), define

\[
r_a^p := \begin{cases} 
1, & \text{if } p \text{ arcs are chosen and } a \text{ is one of them}, \\
0, & \text{otherwise}.
\end{cases} \tag{12.25}
\]
Denote the element in $c_x$ and $c_y$, respectively, associated with arc $a$ by $c_{xa}$ and $c_{ya}$.

Based on the binary variables, $r$, the formulation below is derived,

\[
\text{minimize } \sum_{a \in \delta^+(i)} \left( c_{xa} x_a^k + c_{ya} y_a^f \right) \\
\text{subject to } \sum_{a \in \delta^+(i)} x_a^k = 1, \quad (12.26a) \\
\sum_{p=1}^n \frac{1}{p^2} r_a^p = x_a^k, \quad a \in \delta^+(i), \quad (12.26b) \\
\sum_{p=1}^n r_a^p = y_a^f, \quad a \in \delta^+(i), \quad (12.26c) \\
\sum_{a \in \delta^+(i)} \frac{1}{p^2} r_a^p = v_i^k, \quad (12.26d) \\
r_a^n = r_{a'}^n, \quad a, a' \in \delta^+(i), \quad (12.26e) \\
\sum_{a' \in \delta^+(i): a' \neq a} r_a^p \geq (p-1)r_a^p, \quad a \in \delta^+(i), \quad p \in Z_2^{n-1}, \quad (12.26f) \\
r_a^p \geq 0, \quad a \in \delta^+(i), \quad p \in Z_1^n, \quad (12.26g) \\
x \in R^{\delta^+(i)}, \ y \in B^{\delta^+(i)}, \ v \in R, \ r \in B^{\delta^+(i)} \times Z_1^n. \quad (12.26h)
\]

The rationale behind the model is as follows. Constraints (12.26b)-(12.26d) define $x, y$ and $v$ from $r$. Constraint (12.26f) makes sure that if $r_a^n = 1$, i.e. if we decide to choose $p$ arcs including $a$, then we also choose $p-1$ of the other arcs. Similarly, constraint (12.26e) guarantees that if some $r_a^n = 1$, then all $n$ arcs are chosen.

Denote the convex hull of the feasible solutions to (12.26) by $P_{xyv}$. We will show that the LP-relaxation of (12.26) has the integrality property.

To simplify the analysis, we eliminate $x, y$ and $v$, and then replace $r_a^n$ for $a \in \delta^+(i)$ by a common variable $\bar{r}_i$. The cost of $\bar{r}_i$ is denoted by $\bar{c}_i$, it is the objective value obtained by using all arcs, i.e.

\[
\bar{c}_i := \sum_{a \in \delta^+(i)} \left( \frac{1}{p} c_{xa} + c_{ya} \right). \quad (12.27)
\]

We also use the vector $\bar{c}^p := \frac{1}{p} c_x + c_y$ from Algorithm 12.1.1. The element in $\bar{c}^p$ associated with $a \in \delta^+(i)$ is denoted by $\bar{c}_a^p$.

After some algebra, this yields a formulation only in $r$-variables,
minimize \[ \sum_{a \in \delta^+(i)} \sum_{p=1}^{\bar{n}-1} \bar{c}_a^p r_a^p + \bar{c}_i \bar{r}_i \]

subject to
\[ \sum_{a \in \delta^+(i)} \sum_{p=1}^{\bar{n}-1} \frac{1}{p} r_a^p + \bar{r}_i = 1, \quad (12.28a) \]
\[ \sum_{a' \in \delta^+(i): a' \neq a} r_a^p \geq (p-1)r_a^p, \quad a \in \delta^+(i), \quad p \in \mathbb{Z}^{\bar{n}-1}_2, \quad (12.28b) \]
\[ r_a^p \geq 0, \quad a \in \delta^+(i), \quad p \in \mathbb{Z}^{\bar{n}-1}_1, \quad (12.28c) \]
\[ \bar{r}_i \geq 0, \quad (12.28d) \]
\[ \bar{r}_i \in \mathbb{R}, \quad r \in \mathbb{B}^{\delta^+(i) \times \mathbb{Z}^{\bar{n}-1}_2}. \quad (12.28e) \]

The convex hull of the feasible solutions to (12.28) is denoted by \( P^R \). We want to show that the LP-relaxation of (12.28) has the integrality property. To this end, we consider its dual. Let \( w \) be the dual variable of the equality constraint. For each \( a \in \delta^+(i) \) and \( p \in \mathbb{Z}^{\bar{n}-1}_1 \), the dual variable is denoted by \( \pi_a^p \). We obtain the dual problem,

maximize \[ w \]

subject to
\[ \frac{1}{p} w + \sum_{a' \in \delta^+(i): a' \neq a} \pi_a^{p'} - (p-1)\pi_a^p \leq \bar{c}_a^p, \quad a \in \delta^+(i), \quad p \in \mathbb{Z}^{\bar{n}-1}_2, \quad (12.29a) \]
\[ w \leq \bar{c}_a^p, \quad a \in \delta^+(i), \quad p = 1, \quad (12.29b) \]
\[ w \leq \bar{c}_i, \quad (12.29c) \]
\[ \pi_a^p \geq 0, \quad a \in \delta^+(i), \quad p \in \mathbb{Z}^{\bar{n}-1}_2, \quad (12.29d) \]
\[ \pi \in \mathbb{R}^{\delta^+(i) \times \mathbb{Z}^{\bar{n}-1}_2}, \quad w \in \mathbb{R}. \quad (12.29e) \]

**Theorem 12.1**

*The extreme points of the LP-relaxation of (12.28) are integral.*

**Proof:** Take \((c_x, c_y) \in \mathbb{B}^{\delta^+(i) \times \delta^+(i)}\) arbitrary and form the induced objective vector \( \bar{c} \) from \( \bar{c}_a^p \) and \( \bar{c}_i \). Denote the optimal value of (12.28) by \( w^* \) attained by \( r^* \in \mathbb{B}^{\delta^+(i) \times \mathbb{Z}^{\bar{n}}_2} \). If we can show that the optimal value of (12.29) is \( w^* \), then \( r^* \) is an extreme point of the LP-relaxation of (12.28), which is then also integral, since the objective is arbitrary.

To this end, we construct a feasible dual solution with \( w = w^* \). Observe that since \( w \) is fixed, model (12.29) decomposes into \( \bar{n} \) feasibility subproblems, one for each \( p \in \mathbb{Z}_{\bar{n}}^1 \). By the optimality of \( w^* \) in (12.28), the cases \( p \in \{1, \bar{n}\} \) are satisfied. Hence, it suffices to find a feasible \( \pi^p \) for each \( p \in \mathbb{Z}_{\bar{n}}^2 \).
Take \( p \in \mathbb{Z}_2^{n-1} \) and re-write the constraints for \( p \) in (12.29) into standard matrix form,

\[
\frac{w^*}{p} \mathbf{1} + (\mathbf{1} - p \mathbf{I}) \pi + s = \bar{c},
\]

(12.30)

where \( \mathbf{1} \) and \( I \) denote the all ones vector and identity matrix, respectively. W.l.o.g. assume that the rows in (12.30) are sorted so that the topmost row has the smallest value of \( \bar{c} \).

Define \( \bar{R} \) to be the set of arcs induced by the \( p \) smallest values of \( \bar{c} \). The sub-vectors of \( \pi, s \) and \( \bar{c} \) corresponding to the rows induced by \( \bar{R} \) are denoted by \( \pi_{\bar{R}}, s_{\bar{R}} \) and \( \bar{c}_{\bar{R}} \).

Similarly, the rows induced by the complement of \( \bar{R} \) are \( \pi, s \) and \( \bar{c}_{\bar{R}} \). Finally,

\[
\bar{c}_{a'} = \max_{a \in \bar{R}} \bar{c}_a, \quad \text{and} \quad w^*_{\bar{p}} = \sum_{a \in \bar{R}} \bar{c}_a,
\]

(12.31)

i.e. \( \bar{c}_{a'} \) is the \( \bar{p} \)th smallest value of \( \bar{c} \) and \( w^*_{\bar{p}} \) is the sum of the \( p \) smallest values of \( \bar{c} \).

We claim that a non-negative feasible solution to (12.30) is given by

\[
\pi = \begin{pmatrix} \pi_{\bar{R}} \\ \pi_{\bar{R}} \end{pmatrix} = \frac{1}{p} \left( s - \bar{c} + \bar{c}_{a'} \mathbf{1} \right),
\]

(12.32)

and

\[
s = \begin{pmatrix} s_{\bar{R}} \\ s_{\bar{R}} \end{pmatrix}, \quad \text{where} \quad \begin{pmatrix} 1' s_{\bar{R}} \\ s_{\bar{R}} \end{pmatrix} = w^*_{\bar{p}} - w^*.
\]

(12.33)

Note that (12.32) gives \( \pi_{\bar{R}} = 0 \) and that all variables are non-negative since \( \bar{c}_a \leq \bar{c}_{a'} \) when \( a \in \bar{R} \) and \( \bar{c}_{a} \geq \bar{c}_{a'} \) when \( a \in \bar{R} \). It remains to verify that (12.30) holds, i.e.

\[
\begin{pmatrix} \mathbf{1}' - p \mathbf{I} \\ \mathbf{1}' - p \mathbf{I} \end{pmatrix} \begin{pmatrix} \pi_{\bar{R}} \\ \pi_{\bar{R}} \end{pmatrix} = \begin{pmatrix} \mathbf{1}' - p \mathbf{I} \end{pmatrix} \begin{pmatrix} s_{\bar{R}} - \bar{c}_{\bar{R}} + \bar{c}_{a'} \mathbf{1} \end{pmatrix} = \bar{c} - \frac{w^*}{p} \mathbf{1} - s.
\]

(12.34)

To see this, we use

\[
\begin{align*}
\mathbf{1}' \pi_{\bar{R}} &= \frac{1}{p} \left( s_{\bar{R}} - \bar{c}_{\bar{R}} + \bar{c}_{a'} \mathbf{1} \right) \quad = \frac{1}{p} \left( \mathbf{1}' s_{\bar{R}} - \mathbf{1}' \bar{c}_{\bar{R}} + \bar{c}_{a'} \mathbf{1} \right) \\
&= \bar{c}_{a'} + \frac{1}{p} \left( \mathbf{1}' s_{\bar{R}} - w^*_{\bar{p}} \right) = \bar{c}_{a'} - \frac{w^*}{p},
\end{align*}
\]

(12.35)

to obtain,

\[
\begin{pmatrix} \mathbf{1}' - p \mathbf{I} \\ \mathbf{1}' - p \mathbf{I} \end{pmatrix} \begin{pmatrix} \pi_{\bar{R}} \\ \pi_{\bar{R}} \end{pmatrix} = \begin{pmatrix} \mathbf{1}' - p \mathbf{I} \end{pmatrix} \begin{pmatrix} s_{\bar{R}} - \bar{c}_{\bar{R}} + \bar{c}_{a'} \mathbf{1} \end{pmatrix} = \bar{c} - \frac{w^*}{p} \mathbf{1} - s.
\]

(12.36)

We have a dual solution with objective \( w^* \) and the LP-relaxation of (12.28) is integral.
Remark 12.4. In the proof we actually show that the system of inequalities in (12.28) has the total dual integrality property (just assume that \( \tilde{c} \) is integral).

Corollary 12.1
The projection of the LP-relaxation of (12.26) describes the single node splitting polytope, i.e. \( P^C = \text{proj}_{xyv}(P^R_{xyv}) \).

In conclusion, we have essentially four options to use the results in this section to obtain stronger formulations of the splitting at single nodes: (1) do not strengthen the formulation, (2) settle with a relatively weak extended formulation based on \( s \) or \( q \), (3) use the strong formulation based on \( r \), or (4) use the polynomial time solvability of the problem to separate points from \( P^C \).

12.2 An ECMP Relaxation of an IP Network Problem

We derive an ECMP analogue of the problems and formulations considered in the last chapter. Hence, there are several similarities with the presentation in the previous chapter. The main difference is the pricing problem arising in the Dantzig–Wolfe reformulation. The solution of the pricing problem is deferred to Section 12.3.

As in the USPR case in Chapter 11, we assume that every destination, \( l \in L \), satisfies

\[
K_l = N \setminus \{l\},
\]

which in the ECMP case implies that the induced acyclic ingraphs are spanning. Our starting point is a relaxation of the minimalistic core problem in Section 9.1. The SPR aspect is neglected and ECMP splitting is taken into account. We derive a Dantzig–Wolfe reformulation. This yields a standard multicommodity flow problem with two modifications: (1) ECMP splitting, and (2) the objective is to minimize the congestion measured as the relatively maximally utilized link. The formulation of this problem becomes,

\[
\begin{align*}
\text{minimize} \quad & \zeta \\
\text{subject to} \quad & \sum_{k \in K} h_k x_a^k \leq u_a \zeta, \quad a \in A, \\
& \sum_{a \in \delta^+(i)} x_a^k - \sum_{a \in \delta^-(i)} x_a^k = b_i^k, \quad i \in N, \ k \in K, \\
& x_a^k \leq y_a^l, \quad a \in A, \ k \in K^l, \ l \in L, \\
& x_a^k \leq v_i^k, \quad a \in \delta^+(i), \ i \in N, \ k \in K^l, \ l \in L, \\
& v_i^k x_a^k \leq 1 - y_a^l, \quad a \in \delta^+(i), \ i \in N, \ k \in K^l, \ l \in L, \\
& x_a^k \geq 0, \quad a \in A, \ k \in K^l, \ l \in L,
\end{align*}
\]

\( v \in \mathbb{R}^{N \times K}, \ x \in \mathbb{R}^{A \times K}, \ y \in \mathbb{R}^{A \times L}, \ z \in \mathbb{R} \).
12.2 An ECMP Relaxation of an IP Network Problem

The LP-relaxation of (12.38) is weak in the sense that the optimal LP value is no better than relaxing the splitting constraints, i.e. constraints (12.38d)-(12.38e) are LP redundant. This holds as long as there are no restrictions or incitements for some \( y \) variables to be strictly larger than the corresponding \( x \) variables.

**Proposition 12.3**

Let \( z^*_L \) and \( z^*_R \) be the optimal values of the LP-relaxations of (12.38) with and without the splitting constraints (12.38d)-(12.38e), respectively. Then, \( z^*_L = z^*_R \).

**Proof:** Let \((x^*, y^*)\) be an optimal solution to (12.38) without ECMP splitting constraints. We construct a solution \((\tilde{x}; \tilde{y}; \tilde{v})\) to the LP-relaxation of (12.38) from \((x^*; y^*)\) as follows. Set \( \tilde{x} = x^* \), \( \tilde{y} = x^* \) and \( \tilde{v} = 1 \); this solution is feasible. Since the objective value depends only on the \( x \) variables and \( \tilde{x} = x^* \), the result follows.

**Remark 12.5.** The solution constructed in the proof of Proposition 12.3 is feasible even if the single node splitting constraints are replaced by constraints describing the convex hull of the single node splitting polytope. This is related to the fact that we are actually considering a discretization instead of a convexification, see e.g. [166] or [148, Chapter 13] for a discussion of the difference of convexification and discretization and its consequences.

Next, we present a Dantzig–Wolfe reformulation of (12.38) based on discretization where we enumerate feasible ECMP flow patterns, i.e. \( x \)-solutions.

### 12.2.1 A Discretization Approach via ECMP Flow Patterns

Given a fixed destination, \( l \in L \), the set of feasible ECMP flow solutions are described by the integer linear inequality system,

\[
\begin{align*}
\sum_{k \in K^l} h^k x^k_a &= \tilde{x}^l_a, & a \in A, \\
\sum_{a \in \delta^+ (i)} x^k_a - \sum_{a \in \delta^- (i)} x^k_a &= b^i_k, & i \in N, k \in K^l, \\
0 &\leq x^k_a \leq y^l_a, & a \in A, k \in K^l, \\
0 &\leq v^k_a - x^k_a \leq 1 - y^l_a, & a \in \delta^+ (i), i \in N, k \in K^l, \\
v &\in \mathbb{R}^{N \times K^l}, & x \in \mathbb{R}^A \times \mathbb{K}^l, & \tilde{x} \in \mathbb{R}^A, & y \in \mathbb{B}^A.
\end{align*}
\]

where \( \tilde{x}^l_a \) is an auxiliary variable for the total flow on arc \( a \).

Let \( \mathcal{P}^l \) be the set of points representing the aggregated flow and routing patterns induced by a feasible solution to (12.39), i.e.

\[
\mathcal{P}^l := \left\{ (\tilde{x}, y) \in \mathbb{R}^A \times \mathbb{B}^A \mid (\tilde{x}, \{x^k\}_{k \in K^l}, y, v) \text{ solves (12.39)} \right\}.
\]

Denote the projection onto \( \tilde{x} \) and \( y \) components of \( \mathcal{P}^l \) by \( \mathcal{P}^l_{\tilde{x}} \) and \( \mathcal{P}^l_{y} \), respectively, i.e.
\[ P^l_x := \{ \bar{x} \in \mathbb{R}^A \mid (\bar{x}, y) \in P^l \text{ for some } y \in \mathbb{B}^A \}, \]
\[ P^l_y := \{ y \in \mathbb{B}^A \mid (\bar{x}, y) \in P^l \text{ for some } \bar{x} \in \mathbb{R}^A \}. \]  

(12.41)

**Remark 12.6.** Note that \( P^l_y \) corresponds to the set of ingraphs earlier denoted by \( I^l \).

Let \( \bar{x}_1^l, \ldots, \bar{x}_p^l, \ldots, \bar{x}^l_{r^l} \), be an enumeration of the points in \( P^l_x \). The component of a point \( \bar{x}_p^l \in P^l_x \) corresponding to arc \( a \in A \) is denoted by \( \bar{x}_a^l \). To obtain a Dantzig–Wolfe reformulation of (12.38) we use the variable \( \lambda^l_p \) to be set to 1 if point \( \bar{x}_p^l \in P^l_x \) is selected.

This yields the Dantzig–Wolfe master problem,

\[
\text{minimize} \quad \zeta
\]
\[ \text{subject to} \]
\[ \sum_{l \in L} \sum_{p=1}^{r^l} \bar{x}_a^l \lambda^l_p \leq u_a \zeta, \quad a \in A, \]  
\[ \sum_{p=1}^{r^l} \lambda^l_p = 1, \quad l \in L, \]  
\[ \lambda^l_p \in \mathbb{B}, \quad p = 1, \ldots, r^l, \quad l \in L, \]  

(12.42a-12.42c)

To solve the LP-relaxation of (12.42) by column generation, we use dual variables \( \mu_a \leq 0 \) for the capacity constraint (12.42a) and \( \eta^l \) for constraint (12.42b). For a fixed \( l \in L \), this yields the reduced cost of the column associated with ingraph \( p \in P^l_x \) as

\[ \bar{\zeta}^l_p = -\eta^l - \sum_{a \in A} \bar{\bar{x}}_{ap} \mu_a, \]  

(12.43)

which gives the pricing problem,

\[
\text{minimize} \quad -\eta^l = \sum_{a \in A} \sum_{k \in K^l} \mu_a h^k \bar{x}_a^k
\]
\[ \text{subject to} \]
\[ \sum_{a \in \delta^+(i)} \bar{x}_a^k - \sum_{a \in \delta^-(i)} \bar{x}_a^k = b_i^k, \quad i \in N, \quad k \in K^l, \]  
\[ 0 \leq \bar{x}_a^k \leq y_a, \quad a \in A, \quad k \in K^l, \]  
\[ 0 \leq v_k^i - x_k^i \leq 1 - y_a, \quad a \in \delta^+(i), \quad i \in N, \quad k \in K^l, \]  
\[ v \in \mathbb{R}^{N \times K^l}, \quad x \in \mathbb{R}^{A \times K^l}, \quad y \in \mathbb{B}^A. \]  

(12.44a-12.44d)

A (potentially) improving column is obtained from an optimal solution to (12.44) via

\[ \bar{x}_a^l = \sum_{k \in K^l} h^k \bar{x}_a^k, \quad a \in A. \]  

(12.45)
The next result follows from Proposition 12.3.

**Corollary 12.2**

If \( \mu_a \leq 0 \) for all \( a \in A \), then there is an optimal solution to (12.44) where \( y \) is an anti-arborescence.

This implies that (12.44) can be solved as a shortest path problem since there is no cost on the \( y \)-variables. Also note that all optimal solutions to (12.44), i.e., even solutions in \( \mathcal{I} \setminus \mathcal{T} \), can be obtained from the reduced cost zero arcs from an optimal solution to the associated shortest path problem. Indeed, an optimal ingraph must contain only such arcs to be optimal.

Model (12.44) relies on the discretization approach. Further, the convex hull of \( \mathcal{P}_2 \) coincides with the convex hull of arborescence routings considered in the Chapter 11. Since this polytope has the integrality property, this implies that we gain nothing in terms of formulation strength using the Dantzig–Wolfe reformulation approach.

**Proposition 12.4**

Denote the optimal values of the LP-relaxation of (12.42) and (12.38) by \( z^1_{LP} \) and \( z^2_{LP} \). Further, consider the tree-routing relaxations in Chapter 11 and denote the optimal values of the LP-relaxation of (11.6) and (11.17) by \( z^3_{LP} \) and \( z^4_{LP} \). Then,

\[
z^1_{LP} = z^2_{LP} = z^3_{LP} = z^4_{LP}.
\]

(12.46)

Even though this is a quite negative result, it is not as bad as it seems. From a computational perspective, observe that the Dantzig–Wolfe master, (12.42), has very few constraints compared to the original formulation (12.38). This, together with the fact that the pricing problem can be solved very fast (as a shortest path problem), implies that the Dantzig–Wolfe reformulation can be expected to be solved faster than the original formulation. Our computational experiments showed that LP relaxation can indeed be solved much faster via the Dantzig–Wolfe reformulation.

Note that the last two results and the associated comments only apply to the root node. Also observe that it is important that costs are non-negative. When branching forces some variables to 1, the conditions changes. In particular, the pricing problem can no longer be solved as a shortest path problem. Therefore, (12.44) has to be solved. We believe that there is an optimal solution to the LP-relaxation of (12.44), whose value equals the optimal value of (12.44) also with fixations of \( y \)-variables. However, even so, it is still very costly to solve the LP-relaxation of (12.44) compared to a shortest path problem. In Section 12.3, we present \( O(nm) \) and \( O(nm \log n) \) algorithms for the pricing problem with fixations.

### 12.2.2 Solving the Dantzig–Wolfe Reformulation

Model (12.42) can be solved by Branch-and-Price (B&P). We need to specify a branching scheme; it suffices to branch on the original variables, i.e., \( y \). This branching scheme has the desirable properties mentioned in Section 11.3.1, see page 189. First, the properties B1 and B2 are obvious. Then, the enumeration tree is reasonably well balanced (B3). Indeed, given two arcs \( a, a' \in \delta^+(i) \), the decision of setting \( y_a = 1 \) no longer forces
\[ y_{a'} = 0 \] as in the USPR case. Finally, we show in Section 12.3 that the pricing problem can be solved efficiently also when some \( y \)-variables are fixed.

We mention that it is possible to branch on an outdegree constraint in a manner similar to SOS-branching. For a node \( i \in N \), partition the emanating arcs as \( \delta^+(i) = S_1 \cup S_2 \) and consider the ternary branching scheme,

\[
\begin{align*}
\sum_{a \in S_1} y_a & \geq 1, \quad \text{and} \quad \sum_{a \in S_2} y_a = 0, \\
\sum_{a \in S_1} y_a & = 0, \quad \text{and} \quad \sum_{a \in S_2} y_a \geq 1, \\
\sum_{a \in S_1} y_a & \geq 1, \quad \text{and} \quad \sum_{a \in S_2} y_a \geq 1.
\end{align*}
\]  

(12.47) (12.48) (12.49)

The potential advantage of this scheme is that it better balances the search tree.

A related branching scheme is to branch on the splitting variables described in Section 12.1. For instance, fixing a splitting variable corresponds to branching on a disjunction

\[
\sum_{a \in \delta^+(i)} y_a \leq p, \quad \text{or} \quad \sum_{a \in \delta^+(i)} y_a \geq p + 1.
\]  

(12.50)

This is a major decision. Hence, we would like to branch on these constraints early on. It is straightforward to branch on the other types of splitting variables in a similar manner.

The good news about the above non-standard branching schemes is that it is possible to take them into account in the pricing problem, see Algorithm 12.3.5.

### 12.2.3 Related Dantzig–Wolfe Reformulations

There are several problems related to (12.38) that can be approached by the Dantzig–Wolfe reformulation technique in a similar manner to the above. We mention some of them here, in particular their pricing problems. To keep the discussion short, we omit all details including their models. These are typically similar to the models considered in this and the previous chapter and/or straightforward to derive.

A natural extension of the ECMP relaxation (12.38) is to take the SPR aspect into account. As in the previous chapter, this results in a Branch-and-Cut-and-Price approach since the SPR conflict constraints must be augmented. Along the lines of the derivation in Section 11.2, this (only) affects the pricing problem (12.44). Indeed, the SPR conflict constraints only involve \( y \)-variables, see Chapter 10. This implies that the optimization is still over the feasible set induced by (12.44), but there will be an additional sum in the objective function involving the \( y \)-variables.

Instead of discretizing the part of the original model, i.e. (12.38), associated with \( x \)-variables, i.e. the ECMP flow patterns, we can also consider convexifying the part associated with \( y \)-variables. By inspection, or by projecting out the \( x \)-variables, it is seen that the feasible solutions formed by \( y \)-variables correspond to acyclic ingraphs. Hence, applying a Dantzig–Wolfe reformulation in this manner results in a pricing problem over the polytope \( \text{conv } \mathcal{I} \). Further, if we would in such an approach also take SPR conflict
12.3 Solving ECMP Splitting Problems

It was observed above that the ECMP splitting pricing problem can be solved as a shortest path problem in the root node. When some variables are fixed at 1, due to branching, this no longer holds. To avoid the costly solution of the pricing problem as formulated in (12.44), we propose a new model for the ECMP splitting problem to destination \( l \in L \) with arc costs \( c \) and fixations.

Let \( A, A^1 \subseteq A \) be the sets of \( y \)-variables fixed by branching to 0 and 1, respectively. For each \( i \in N \setminus \{l\} \), define \( A^1_i := A^1 \cap \delta^+(i) \) and \( N^1 := \{i \in N \mid A^1_i \neq \emptyset\} \).

The ECMP splitting problem with fixations can be seen as selecting a feasible subset of emanating arcs from each node. For a node \( i \in N \setminus \{l\} \), the feasible subsets of emanating arcs are all \( S_i \) where \( A^1_i \subseteq S_i \subseteq \delta^+(i) \). Thus, the collection of all feasible subsets is

\[
S := \bigcup_{i \in N \setminus \{l\}} \bigcup_{A^1_i \subseteq S \subseteq \delta^+(i)} \{S\}. \tag{12.51}
\]

We use a kind of (hyper-arc) flow variable for each feasible subset of arcs. For each \( S \in \mathcal{S} \), the interpretation of \( x_S \) is that it measures the total flow sent on all arcs in the arc subset \( S \). By the ECMP principle, the flow has to be equal for all arcs emanating from a node, i.e. for each arc in \( S \). Hence, the flow contribution from \( x_S \) to an arc \( a \in S \) is \( \frac{1}{|S|} x_S \). From this variable definition, the model below follows quite naturally.

\[
\begin{align*}
\text{minimize} & \sum_{S \in \mathcal{S}} \frac{1}{|S|} \sum_{a \in S} c_a x_S \\
\text{subject to} & \sum_{A^1_i \subseteq S \subseteq \delta^+(i)} x_S - \sum_{S \cap \delta^-(i) \neq \emptyset} \frac{1}{|S|} x_S = b_i, \quad i \in N, \tag{12.52a} \\
& x_S \in \mathbb{R}_+, \quad S \in \mathcal{S}, \tag{12.52b}
\end{align*}
\]

where the node balances are defined as
\begin{equation}
\begin{array}{ll}
\quad 218 & 12 \text{ ECMP Modelling and Relaxations} \\
\end{array}
\end{equation}

Constraint (12.52a) is a generalized hyper-flow conservation constraint; the outflow is in a sense scaled and we use directed hyper-arcs. The first sum measures the total outflow from node \( i \) as usual. The second sum measures the total flow into node \( i \). Indeed, if the flow on a hyper-arc \( S \in S \) enters \( i \), i.e. \( S \cap \delta^- (i) \neq \emptyset \), then the flow contribution into \( i \) from this hyper-arc is \( x_S / |S| \). Note that this implies that the total outflow from node \( i \) is equal to the total inflow plus 1, i.e. \( b_i \).

We will not solve model (12.52) directly. Instead we consider its dual. Let \( \pi_i \) be the dual variable for the flow conservation constraints. This yields,

\begin{equation}
\text{maximize } \sum_{i \in N} (\pi_i - \pi_l) \\
\text{subject to } \pi_i \leq \frac{1}{|S|} \sum_{a : = (i,j) \in S} (c_a + \pi_j), \quad A^1_i \subseteq S \subseteq \delta^+ (i), \quad i \in N.
\end{equation}

In analogy with an ordinary shortest path problem, \( \pi_i \) is a distance variable and it is feasible to set \( \pi_l = 0 \). Then, the exact interpretation of \( \pi_i \) is that it measures the weighted average cost (over all paths used) per unit of flow sent from node \( i \) to node \( l \). As in ordinary shortest path problems, feasible dual solutions yield lower bounds on these distances and optimal solutions give exact distances.

**Proposition 12.5**

*In an optimal solution to (12.54) it holds that*

\begin{equation}
\pi_i = \min_{A^1_i \subseteq S \subseteq \delta^+ (i)} \frac{1}{|S|} \sum_{a : = (i,j) \in S} (c_a + \pi_j),
\end{equation}

Observe the resemblance between (12.55) and Bellman’s equations for shortest path problems. This suggests that the ECMP splitting problem can be solved by dynamic programming. Based on this observation, it is clear that there are (at least) two natural approaches to solving the ECMP splitting problem: (1) directly solving the LP model (12.54), or (2) by dynamic programming.

Regardless of the choice of method, there are some preprocessing techniques that can be used. Consider a node, \( i \in N \setminus N^1 \), i.e. \( A^1_i = \emptyset \). In this case, the optimal value (12.55) is attained by a (not necessarily unique) singleton set. When we solve the LP, this implies that for node \( i \), it suffices to consider constraints associated with each set \( S = \{ a \} \) for each \( a \in \delta^+ (i) \). Hence, (12.54) simplifies to

\begin{equation}
\begin{array}{ll}
\quad 218 & 12 \text{ ECMP Modelling and Relaxations} \\
\end{array}
\end{equation}
maximize \[ \sum_{i \in N} (\pi_i - \pi_l) \]
subject to
\[ \pi_i \leq \frac{1}{|S|} \sum_{a := (i,j) \in S} (c_a + \pi_j), \quad A_1^i \subseteq S \subseteq \delta^+(i), \quad i \in N^1, \quad (12.56a) \]
\[ \pi_i \leq c_a + \pi_j, \quad a := (i,j) \in \delta^+(i), \quad i \in N \setminus N^1. \quad (12.56b) \]

This observation gives the following proposition, which is very useful in practice.

**Proposition 12.6**

Let \((\bar{x}, \bar{\pi})\) be primal-dual pair corresponding to an optimal solution to the ordinary shortest path problem with arc costs \(c \in \mathbb{R}^A_+\). Let \(i\) be a node where there exists some shortest \((i, l)\)-path, \(P_{il}\) say, that does not use any node in \(N^1\). Then, for each node \(u \in P_{il}\), the optimal solution to (12.54) satisfies \(\pi_u = \bar{\pi}_u\).

In practice, this result strongly suggests that one should first solve an ordinary shortest path problem. Then, fix the arborescence to \(l\) obtained by removing all nodes in \(N^1\).

**Solving the ECMP Splitting Problem as an LP**

By the equivalence of optimization and separation via the ellipsoid method, see e.g. [127, 128, 180], we only need to consider the separation problem to solve (12.54). It suffices to do so for a single node, \(i\) say. Let \(\bar{x}\) be a point to be separated, i.e. we seek a set \(S_i\) where \(A_1^i \subseteq S_i \subseteq \delta^+(i)\) and

\[ \bar{x}_i > \frac{1}{|S_i|} \sum_{a := (i,j) \in S_i} (c_a + \bar{\pi}_j). \quad (12.57) \]

If such a set \(S_i^*\) exists, it is found by solving

\[ S_i = \arg\min_{A_1^i \subseteq S \subseteq \delta^+(i)} \frac{1}{|S|} \sum_{a := (i,j) \in S} (c_a + \bar{\pi}_j). \quad (12.58) \]

The optimization problem in (12.58) is solved as follows. First observe that it suffices to consider nodes in \(N^1\). Indeed, for other nodes, the number of inequalities to check is polynomial, see (12.56b). When \(|A_1^i| \geq 1\), the crucial observation to be made is as follows. **Whatever the optimal size \(|S_i^*|\) is in (12.58), the \(|S_i^* \setminus A_1^i|\) smallest values of \(c_a + \bar{\pi}_j\) are chosen.** This translates to Algorithm 12.3.1 below.

**Algorithm 12.3.1.** Given an arc set \(A_1^i \subseteq \delta^+(i)\) and costs \(c_a + \bar{\pi}_j\) for \(a := (i,j) \in \delta^+(i)\), find a solution \(S_i^*\) to problem (12.58).

1. Sort the terms \(c_a + \bar{\pi}_j\) for \(a \in \delta^+(i) \setminus A_1^i\). Denote the sorted quantities by

\[ \alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_{r_i}, \quad \text{where} \quad r_i := |\delta^+(i) \setminus A_1^i|. \quad (12.59) \]
2. Let 0 := 0 and \( \bar{\alpha} := \sum_{a \in A^1_i} (c_a + \pi_j) \). Then, for 0 \( k \leq r_i \), define
\[
\beta_k := \frac{1}{|A^1_i| + k} (\bar{\alpha} + \alpha_k).
\] (12.60)

3. Find \( k^* \) and \( \beta_{k^*} \), i.e.
\[
\beta_{k^*} := \min_{0 \leq k \leq r_i} \beta_k.
\] (12.61)

4. Form an optimal solution \( S^*_i \) as the union of the required arcs \( A^1_i \) and the arcs associated with the \( k^* \) smallest terms in Step 1, i.e. in (12.59).

Observe that \( \beta_k \) evaluates the sum in (12.58) for the set containing the required terms and the \( k \) smallest terms. Hence, the optimal value in (12.58) is given by (12.61) and the minimum is attained by the set, \( S^*_i \), constructed in Step 4 of Algorithm 12.3.1.

**Proposition 12.7**

The separation problem for (12.54) can be solved in \( O(n^2 \log n) \) time.

**Proof:** When the separation problem is solved for a single node in Algorithm 12.3.1, the most expensive operation is to sort the terms in Step 1. This requires \( O(n \log n) \) time. A more accurate analysis gives the running time \( O(m + |N^1| \Delta \log \Delta) \) where \( \Delta \) is the maximum degree in the graph. \( \square \)

It is in theory possible to avoid the costly sorting operation in Algorithm 12.3.1. To this end, we observe that the values \( \beta_k \) first decrease, and then increase, when \( k \) grows.

**Proposition 12.8**

The function \( \beta : Z^+_0 \rightarrow \mathbb{R} \) defined by (12.60) is unimodal (also known as quasi-convex).

Based on this observation, we can apply a divide-and-conquer approach to solve the single node separation problem (12.58) in linear time. As in the (theoretically) efficient version of Algorithm 12.1.1 for the general single node problem (12.23) on page 208, we use that the median value can be found in linear time. This implies that \( \beta_k \) can be determined for an arbitrary value of \( k \) without sorting.

**Algorithm 12.3.2.** Given an arc set \( A^1_i \subseteq \delta^+(i) \) and costs \( c_a + \pi_j \) for \( a := (i, j) \in \delta^+(i) \), find a solution \( S^*_i \) to problem (12.58).

1. Set \( E := \delta^+(i) \setminus A^1_i \) and \( S_i := A^1_i \).
2. If \( |E| = 0 \), set \( S^*_i := S_i \). Stop.
3. Set \( k = [(|E| - 1)/2] \). Let \( C(E) := \{c_a + \pi_j : a := (i, j) \in E\} \).
4. Find the \( k \) and \( k + 1 \) smallest elements in \( C(E) \) by splitting \( C(E) \) into two sets with values smaller and larger than the median, respectively. Denote the associated sets of arcs by \( C_k \) and \( C_{k+1} \).
5. Determine $\beta_k$ and $\beta_{k+1}$ via

$$
\beta_k = \frac{1}{|S_i \cup C_k|} \sum_{a \in S_i \cup C_k} (e_a + \pi_j), \\
\beta_{k+1} = \frac{1}{|S_i \cup C_{k+1}|} \sum_{a \in S_i \cup C_{k+1}} (e_a + \pi_j).
$$

6. If $\beta_k \leq \beta_{k+1}$, set $E := E \setminus C_k$ and $S_i := S_i \cup C_k$. Goto Step 2.
7. Set $E := C_k$. Goto Step 2.

All steps in Algorithm 12.3.2 can be solved in linear time. Hence, the recurrence relation for the running time is as in (12.24) and Algorithm 12.3.2 solves problem (12.58) in $O(n)$ time.

**Proposition 12.9**
The separation problem for (12.54) can be solved in $O(m)$ time.

**Proof:** We solve the separation problem for node $i \in N$ in time $O(\delta(i)) = O(m)$. Hence, the total running time is $\sum_{i \in N} O(\delta(i)) = O(m)$. \qed

**Remark 12.7.** In practice, Algorithm 12.3.1 is superior to Algorithm 12.3.2 since the overhead in the latter algorithm is very large and node degrees in typical instances are quite small.

**Theorem 12.2**
Model (12.54), and hence, the ECMP splitting problem can be solved in polynomial time.

**Solving the ECMP Splitting Problem by Dynamic Programming**

From Proposition 12.5 we have that

$$
\pi_i = \min_{A^1_i \subseteq S \subseteq S^+(i)} \frac{1}{|S|} \sum_{a \in S \setminus \{ i \}} (e_a + \pi_j).
$$

In dynamic programming terms, this can be interpreted as a functional equation. To solve (12.64), we use (the less common) approach of successive approximation instead of directly via recursion, see e.g. [212] for a treatment of the successive approximation approach in general and [211] applied to the shortest path problem. This results in a method almost identical to the Bellman-Ford algorithm for shortest path problems with negative arc costs and no negative cost cycles.

**Algorithm 12.3.3.** Given $c \in \mathbb{R}_+^I$ and $A^1_i$ for $i \in N^1$, solve (12.64) for all $i \in N$.

1. Initialize $\pi$. Set $\pi_l = 0$ and $\pi_i = \infty$ for $i \in N \setminus \{ l \}$.
2. For $k = 1$ to $n$ do the following.
(a) For $i \in N \setminus \{l\}$, update $\pi_i$ by solving

\[
\pi_i := \min_{A^i \subseteq S \subseteq \hat{\delta}^+(i)} \frac{1}{|S|} \sum_{a := (i,j) \in S} (c_a + \pi_j).
\]  

(12.65)

by using Algorithm 12.3.1 (or Algorithm 12.3.2).

For the correctness proof, we need a definition. An $(s;t)$-connector is a subgraph that contains an $(s;t)$-path, see e.g. [205, Chapter 13]. We are interested in the $(s;t)$-connectors where no arcs enter $s$, no arcs leave $t$ and all other nodes have both entering and leaving arcs.

**Definition 12.1**
A closed $(s;t)$-connector, $C$, is an $(s;t)$-connector that is acyclic, connected and satisfies $\delta^-(s) = \delta^+(t) = \emptyset$ and $\delta^-(i) \neq \emptyset$, $\delta^+(i) \neq \emptyset$ for $i \in N(C) \setminus \{s,t\}$.

A closed $(s,t)$-connector defines an ECMP splitting. The average cost of sending a unit from $s$ to $t$ via this splitting will be referred to as its ECMP length and the number of arcs in a longest path from $s$ to $t$ as its size. In the trivial case of an $(s,t)$-path, the ECMP length coincides with the path cost and the size with the number of arcs in the path.

**Proposition 12.10**
Algorithm (12.3.3) correctly solves the functional equation (12.64).

**Proof:** The proof follows essentially verbatim from the proof of correctness of the Bellman-Ford algorithm via induction over $k$, see e.g. [4, p. 142]. The main observation is that at iteration $k$ (i.e. the stage in a formal recursive dynamic programming description) it holds that:

- If $\pi_i < \infty$, then there is a closed $(i,l)$-connector where the ECMP length is $\pi_i$.
- If there is a closed $(i,l)$-connector of size at most $k$, then $\pi_i$ is at most the ECMP length of the shortest (w.r.t. ECMP length) $(i,l)$-connector of size at most $k$.

**Proposition 12.11**
The running time of Algorithm 12.3.3 is $O(mn \log \Delta)$ if Algorithm 12.3.1 is used in Step 2a and $O(mn)$ if Algorithm 12.3.2 is used.

**Proof:** There are $n$ iterations. For Algorithm 12.3.1 the remark in the proof of Proposition 12.7 gives $\sum_{i \in N} O(\delta^+(i) \log \delta^+(i)) = O(m \log \Delta)$. For Algorithm 12.3.2 the result is trivial.

**Corollary 12.3**
The ECMP splitting problem can be solved in polynomial time.

**Remark 12.8.** Observe that dynamic programming approach for the ECMP splitting problem is in theory as efficient as the Bellman-Ford approach for the closely related shortest path problem. (The additional cost is a factor $\log \Delta$ when Algorithm 12.3.1 is used).
12.3 Solving ECMP Splitting Problems

It is well known, see e.g. [4], that the practical running time of the Bellman-Ford algorithm can be improved, e.g. the dequeue implementation seems to work very well. However, this comes at the price of a pseudo-polynomial running time in theory, see e.g. [4, 205]. We also mention that there are $O(mn)$ implementations that perform comparably well to the dequeue implementation, see e.g. [121].

Based on the above discussion, we think that it is very reasonable to expect that a dequeue implementation of Algorithm (12.3.3) solves the ECMP splitting problem more efficiently in practice, even though its theoretical running time is pseudo-polynomial.

Finally, we consider another successive approximation algorithm were we use lower bounds on the distance instead of upper bounds as in Algorithm (12.3.3).

**Algorithm 12.3.4.** Given $c \in \mathbb{R}_+^A$ and $A_i^1$ for $i \in N^1 \subset N$, solve (12.64) for $i \in N$.

**Notation:** $M$ is the set of settled nodes and $S_i$ the set of arcs currently used from node $i$.

1. Initialize.

   (a) Mark $l$ as settled and all other nodes as unsettled, i.e. set $M = \{l\}$.
   (b) Set $\pi_l = 0$ and determine $\pi_i$ by finding a shortest path anti-arborescence to $l$ with arc costs $c$.
   (c) Set the successor set, $S_i$, of node $i$ to $\{a\}$, where $a \in \delta^+(i)$ is arc emanating from $i$ in the shortest path anti-arborescence obtained in the previous step.
   (d) For each node $i$, if the path from $i$ to $l$ does not use a node in $N^1$, then mark $i$ as settled, i.e. set $M := M \cup \{i\}$.

2. Repeatedly update the average costs until all nodes are settled, i.e. while $N \neq M$.

   (a) Choose an unsettled node, $i \in N \setminus M$.
   (b) Update $\pi_i$ and $S_i$ by using (12.64).
   (c) If all nodes in $S_i$ are settled, i.e. $S_i \subseteq M$, then mark $i$ as settled, i.e. set $M := M \cup \{i\}$.

We argue that if the unsettled node, $i \in N \setminus M$, to be updated in Step 2b is chosen properly, then Algorithm 12.3.4 correctly solves the functional equation (12.64). By properly, we mean that as long as no node $i \in M \setminus N$ gets a new distance label $\pi$ or becomes settled in Step 2, a new node gets examined. This holds for instance (trivially) for FIFO queues. Indeed, after examining all unsettled nodes (which occurs since we choose nodes properly) Algorithm 12.3.4 makes progress in the sense that some node either gets improved or settled. Hence, Algorithm 12.3.4 terminates with a solution to (12.64).

**Remark 12.9.** The value of $\pi_i$ in Algorithm 12.3.4 is always a lower bound on the correct value of $\pi_i$.

**Proposition 12.12**

*Algorithm 12.3.4 correctly solves the functional equation (12.64).*

The discussion of the practical and theoretical running time of Algorithm 12.3.3 applies also to Algorithm 12.3.4. A potential advantage of the latter is its utilization of...
settled nodes where we know that we have found the correct value of \( \pi_i \). (Observe the close connection to Proposition 12.5.)

We conclude the treatment of the ECMP splitting problem by discussing two issues related to the bigger picture, i.e. solving the original ECMP relaxation. First, we consider over-generation of columns. Then we show how to modify the algorithms for solving the pricing problem for branching rules other than ordinary variable dichotomy branching.

Similar to using over-generation in the USPR as discussed in Section 11.5, it is possible to generate several near-optimal columns from an optimal column. Indeed, given a graph that solves the pricing problem, a new graph is obtained by inserting, or (if feasible w.r.t. outdegree constraints and branching decisions) removing, an arc. Based on the demand \( \mu \) and the distance labels, it is possible to anticipate the effect of the modification which can be used to stay near-optimal when desired.

We turn to the complication caused by allowing more advanced branching rules. A possible description of a very general branching rule that subsumes the ones discussed in Section 12.2.2 is as follows. Given a partitioning of the arcs in \( \delta^+(i) \), i.e.

\[
\delta^+(i) = \bigcup_{k=1}^{K} S^k, \quad \text{and} \quad S^k \cap S^{k'} = \emptyset, \quad \text{for} \quad k \neq k' \in \{1, \ldots, K\},
\]

a feasible set of emanating arcs, \( S_i \), must satisfy associated upper and lower bounds, i.e.

\[
l^k \leq |S_i \cap S^k| \leq u^k, \quad k = 1, \ldots, K.
\]

**Remark 12.10.** Consider the standard branching rule. Let \( A^1_i := \{a_1, \ldots, a_K \} \subseteq \delta^+(i) \) be the variables that have been fixed at 1. In the general branching rule framework above, this is modelled by partitioning \( \delta^+(i) \) via

\[
S^0 := \delta^+(i) \setminus A^1_i, \quad S^k := \{a_k\}, \quad k = 1, \ldots, K,
\]

and using the bounds

\[
l^0 = 0, \quad u^k = |\delta^+(i) \setminus A^1_i|, \quad \text{and} \quad l^k = u^k = 1, \quad k = 1, \ldots, K.
\]

To handle the general branching rule, we adopt Algorithm 12.3.1 to solve the more general separation problem of finding an \( S \subseteq \delta^+(i) \).

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{|S|} \sum_{a = (i,j) \in S} (c_a + \pi_j) \\
\text{subject to} & \quad l^k \leq |S \cap S^k| \leq u^k, \quad k = 1, \ldots, K.
\end{align*}
\]

In analogy with the derivation of Algorithm 12.3.1 we observe that it suffices to sort the terms \( (c_a + \pi_j) \). Since there are lower bounds on the number of terms to be chosen from each partition, we first select the required number of the smallest terms in each partition. Then, we consider the remaining terms in ascending order; if a term decreases
the objective value and does not violate some upper bound, the associated arc is selected to be in the solution. Terminate in as soon as a non-improving term is found. This yields the following algorithm.

Algorithm 12.3.5. Find an optimal solution, $S^*_i$, to problem (12.70)

Notation: $\hat{u}^k$ denotes how many additional elements are allowed in $S^k$.

1. Set $S_i := \emptyset$ and $E := \delta^+(i)$.

2. For $k = 1, \ldots, K$, do the following.

   (a) Sort the terms $(c_a + \pi_j)$ for $a \in S^k$. Let $C$ be the arcs associated with the $l^k$ smallest terms.

   (b) Select the arcs in $C$, i.e. set $S_i := S_i \cup C$ and $E := E \setminus C$.

   (c) Set $\hat{u}^k := \hat{u}^k - l^k$.

3. Determine the current objective value (modulo scaling), $\hat{\alpha} := \sum_{a \in S_i} (c_a + \pi_j)$.

4. In increasing order of $(c_a + \pi_j)$ for $a \in E$, do the following.

   (a) If the term $c := (c_a + \pi_j)$ is not improving, i.e. if

   \[ \frac{\hat{\alpha}}{|S_i|} < \frac{\hat{\alpha} + c}{|S_i| + 1}, \]  

   then goto Step 5.

   (b) If the term $c$ is allowed, i.e. if $a \in S^k$ and $\hat{u}^k > 0$, then select it and update, i.e. set $S_i := S_i \cup \{a\}$, $\hat{u}^k := \hat{u}^k - 1$ and $\hat{\alpha} := \hat{\alpha} + c$.

5. Set $S^*_i := S_i$.

**Proposition 12.13**

Algorithm 12.3.5 solves problem (12.70) in $O(\hat{n} \log \hat{n})$ time.

**Proposition 12.14**

When the general branching rule is used, the separation problem for (12.54) can be solved in $O(n^2 \log n)$ time.

It is possible to improve the theoretical running time of Algorithm 12.3.5 to $O(\hat{n})$ by the same technique as above. Hence, the separation problem can be solved in $O(n^2)$ time. We omit the details of this straightforward modification.

**Theorem 12.3**

The ECMP splitting problem can be solved in polynomial time also when the general branching rule is used.
12.4 Solving the Acyclic Ingraph Problem

The acyclic ingraph problem is to find a minimum cost acyclic sub-graph such that there is a path from each node to the root node. Using notation from above it is the problem to,

\[
\begin{align*}
\text{minimize} & \quad c'y \\
\text{subject to} & \quad y \in \mathcal{I}^l.
\end{align*}
\]  
(12.72a)

We are interested in this problem mainly for the following reasons: (1) it arises as a pricing problem in some problems with ECMP splitting (with and without SPR), and (2) since \( \mathcal{Y} \subseteq \mathcal{I} \), the valid inequalities of \( \text{conv} \mathcal{I} \) are potentially useful also for \( \text{conv} \mathcal{Y} \). Further, (3) we think it is an interesting combinatorial optimization problem on its own; it is closely connected to the arborescence problem (see e.g. [97, 123] and Chapter 52 in [205]), the linear ordering problem (see e.g. [124, 125, 159, 160, 173]) and the acyclic subgraph problem (see e.g. [119, 126]). More precisely, the arborescence polytope is the minimal rank face of \( \text{conv} \mathcal{I} \), the linear ordering polytope is the maximal rank face of \( \text{conv} \mathcal{I} \) and the acyclic subgraph problem is the relaxation obtained by dropping all outdegree constraints.

Based on the relation to the arborescence, linear ordering and acyclic subgraph problems, it comes as no surprise that the acyclic ingraph problem is in general NP-hard, but solvable in polynomial time for non-negative objective functions.

**Theorem 12.4**

Let \( c \in \mathbb{Q}^A \) be a cost vector. Then, problem (12.72) is solvable in polynomial time if \( c \in \mathbb{Q}_A^+ \) and NP-hard otherwise.

**Proof:** If \( c \in \mathbb{Q}_A^+ \), there is an optimal solution that is a spanning anti-arborescence. There are several polynomial time algorithms for solving this problem, e.g. [54, 78, 97, 104, 217].

When \( c \) is allowed to be negative it is easy to polynomially reduce a linear ordering instance on a graph with \( n \) nodes to an instance of the acyclic ingraph problem on a graph with \( n + 1 \) nodes as follows. First recall that the linear ordering problem is to find a permutation \( p = p_1, \ldots, p_n \) of the nodes that minimizes

\[
\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c(p_i, p_j).
\]  
(12.73)

Consider a linear ordering instance on a graph \( G \) with cost vector \( c \). Augment \( G \) with a root node, 0, and an arc from each node in \( G \) to 0. Set the cost vector \( c \) to 0 on all new arcs to 0 and solve (12.72) on the augmented graph, with costs \( c - M \), where \( M \) is a vector where all components are equal to the largest element in \( c \), plus 1. This implies that all costs are negative and that an optimal solution have the maximum \( n(n+1)/2 \) arcs (i.e. it lies in the maximum rank face, which is the linear ordering polytope). An optimal linear ordering is obtained by deleting the augmented root node, 0.

We will first consider the polytope \( \text{conv} \mathcal{I}^l \) and derive some classes of facets. Then, we present the core of a heuristic for solving the acyclic ingraph problem.
12.4 Solving the Acyclic Ingraph Problem

12.4.1 The Acyclic Ingraph Polytope

An integer linear description for $I_l$ was given in Section 10.3 in model $(10.12)$ on page 155. We repeat this formulation here. It will be assumed that the graph is essentially complete, except for arcs leaving the root, i.e. $A = (N \setminus \{l\}) \times N$. Let $D^l$ be the collection of all directed cycles in $G$ not containing node $l$. Then, ingraphs corresponds 1-to-1 to binary incidence vectors $y \in \mathbb{B}^A$ that solves the following system.

\[
\sum_{a \in \delta^{+}(i)} y_a^l \geq 1, \quad i \in S^l, \quad (12.74a)
\]
\[
\sum_{a \in C} y_a^l \leq |C| - 1, \quad C \in D^l, \quad (12.74b)
\]
\[
y_a^l \in \mathbb{B}, \quad a \in A. \quad (12.74c)
\]

We analyze $\text{conv } I^l$. First, consider the dimension and the trivial facets.

**Proposition 12.15**

When $A = (N \setminus \{l\}) \times N$ we have the following.

(a) If $n \geq 3$, then $\text{conv } I^l$ is full-dimensional, i.e.

\[
\dim \text{conv } I^l = |A| = (n - 1)^2. \quad (12.75)
\]

(b) If $n \geq 4$, then the variable lower bound induces a facet,

\[
y_a \geq 0, \quad (12.76)
\]

of $\text{conv } I^l$ for each arc $a \in A$.

(c) If $n \geq 3$, then the variable upper bound induces a facet,

\[
y_a \leq 1, \quad (12.77)
\]

of $\text{conv } I^l$ if and only if $a \in \delta^-(l)$. (When $a \in A \setminus \delta^-(l)$, the dimension of the upper bound face is $\dim \text{conv } I^l - 2$, i.e. it is a ridge.)

To prove the results in the above proposition it suffices to find an appropriate number of affinely independent points. For all parts, we can choose an appropriate subset of the following (structurally very simple) set of points. Take $p^{(0)} := \chi(\delta^-(l))$, i.e. $p^{(0)}$ is the incidence vector associated with the instar to the root node, $l$. Then, create $p^{(a)}$ from $p^{(0)}$ for $a \in A$ by augmenting the arc $a$. Also, create $\bar{p}^{(a)}$ from $p^{(0)}$ for $a \in A$ by augmenting the arc $a$ and removing the arc in $p^{(0)}$ with the same tail as $a$ (the tail of $a := (i,j)$ is $i$). Observe that $p^{(0)}, \bar{p}^{(a)} \in T^l$ and $p^{(a)} \in T^l$.

Using the same points, it is also easy to prove the next result.
Proposition 12.16
When \( A = (N \setminus \{l\}) \times N \), the outdegree inequality,
\[
\sum_{a \in \delta^+(i)} y_a \geq 1,
\] (12.78)
induces a facet of \( \text{conv } I^l \) for all \( i \in N \setminus \{l\} \).

The remaining inequalities in (12.74) prohibit cycles. They also induce facets.

Proposition 12.17
When \( A = (N \setminus \{l\}) \times N \), the cycle prohibiting inequality,
\[
\sum_{a = (i;j)} y_a \leq |C| - 1,
\] (12.79)
induces a facet of \( \text{conv } I^l \) for each cycle \( C \in D_l \).

Proof: Let \( F \) be the face induced by (12.79). Assume that \( F \) is not a facet, but contained in a facet \( F' \) induced by the inequality
\[
\sum_{a \in A} a_a y_a \leq \beta.
\] (12.80)
We show that (12.80) is (12.79) up to scaling.

Arbitrary choose an arc \( \tilde{a} := (i,j) \in C \). Construct \( p^{(ij)} \) by setting \( y_{\tilde{a}} = 1 \) for \( a \in C \setminus \{\tilde{a}\} \) and also for \( a \in \delta^{-}(i) \cup \delta^{+}(j) \cup \delta^{-}(l) \), i.e. the ingraph associated with \( p^{(ij)} \) contains all arcs in the cycle except \( \tilde{a} \) and in addition all arcs entering \( i \) or \( l \) and all arcs leaving \( j \). See Figure 12.1.

Observe that the point \( p^{(ij)} \) remains feasible even if an arc entering \( i \) or an arc leaving \( j \) is removed. If an arc not incident with the cycle is reversed in \( p^{(ij)} \), it also remains feasible. Indeed, no cycle is created and every node still has at least one emanating arc.

Consider an arc \( a \in (\delta^{-}(i) \cup \delta^{+}(j)) \setminus C \). Create the point \( p \) from \( p^{(ij)} \) by removing the arc \( a \). Then, \( p \in F' \). Hence, \( \alpha_a = 0 \), since \( p \) and \( p^{(ij)} \) differ only in this component.

Since \( \tilde{a} \) was chosen arbitrary, \( \alpha_a = 0 \) for all non-cycle arcs, \( a \not\in C \), with an endpoint in \( N(C) \). For an arc \( a \) without an endpoint in \( N(C) \) we take some \( p^{(ij)} \) from above.

Construc \( p \) from \( p^{(ij)} \) by changing the value of \( y_{\tilde{a}} \) to \( 1 - y_a \). Then, \( p \in F \) and \( \alpha_a = 0 \).

From this we have that the only non-zero \( \alpha_a \) in (12.80) have \( a \in C \). Since \( p^{(ij)} \in F \) for all \( a = (i,j) \in C \), it follows that \( \alpha_a = \alpha_{a'} \) for all \( a,a' \in C \). Thus, (12.79) is (12.80) up to scaling, so \( F \) must be a facet.

Figure 12.1: The support graph of the point \( p^{(ij)} \) in the proof of Proposition 12.17 associated with the arc \( \tilde{a} := (i,j) \in C \). Bold arcs are in the cycle.
Next, we derive a large class of facets for $\text{conv} \mathcal{I}$. Denote by $F(R; S)$ the collection of rooted forests that have its roots in the node set $R$ and spans the node set $S$. See Figure 12.2 for an example of a rooted forest. Recall that the set of arcs from node set $S$ to node set $T$ is commonly denoted by $\gamma(S, T) := \{(i, j) \in A : i \in S, j \in T\}$.

**Proposition 12.18**

Take $Q, R \subset N$ such that $Q \cap R = \emptyset$ and $l \in Q$. Then, for each $F \in \mathcal{F}(R, N \setminus Q)$, the inequality,

$$\sum_{a \in F} y_a - \sum_{a \in \gamma(R, Q)} y_a \leq |F| - 1, \quad (12.81)$$

is valid for $\text{conv} \mathcal{I}$.

**Proof:** Validity of this inequality is straightforward; if all arcs in the forest, $F$, are chosen, then there must also be an arc $(i, j) \in A$ from some forest root node, $i \in R$, to some non-forest node $j \in Q = N \setminus N(F)$. \hfill \Box

It is in fact the case that (12.81) very often induces a facet. To describe the only exception we need some definitions. Let $\mathcal{F} \subset \mathcal{F}$, and define $S_0(F)$ to be the set of nodes in $F$ that dangles from a root, i.e. they are directly connected to some root node and has no entering arcs,

$$S_0(F) := \{i \in N(F) \mid \delta^+(i) \cap \delta^-(R) \neq \emptyset \text{ and } \delta^-(i) = \emptyset\}. \quad (12.82)$$

Define $\mathcal{F}$ to be the family of rooted forests with a single root and some dangling node, i.e.

$$\mathcal{F} := \bigcup_{r_1 \in N \setminus \{l\}} \{F \in \mathcal{F}(\{r_1\}, N \setminus \{l\}) \mid S_0(F) \neq \emptyset\}. \quad (12.83)$$

The rooted forest in Figure 12.2 has two dangling nodes marked with a $d$. Since it has three roots, it does not belong to $\mathcal{F}$. According to the following proposition, this implies that the associated inequality induces a facet.

**Proposition 12.19**

Assume $A = (N \setminus \{l\}) \times N$. Take $Q, R \subset N$ such that $Q \cap R = \emptyset$ and $l \in Q$. Then, for each $F \in \mathcal{F}(R, N \setminus Q)$, the inequality,
\[ \sum_{a \in F} y_a - \sum_{a \in \gamma(R, Q)} y_a \leq |F| - 1, \]  

(12.84)

induces a facet of \( \text{conv} I^I \) if and only if \( F \notin \mathcal{F} \).

**Proof:** To see necessity, take \( F \in \mathcal{F} \), and let \( i \in S_0(F) \) and \((i, r_1) \in F\). The face induced by the inequality associated with \( F \) implies that either the arc \( a := (i, r_1) \) or its reversal \( a := (r_1, i) \) must be chosen, i.e. \( y_a + y_\overline{a} = 1 \). Indeed, if \((i, r_1) \in F\) is not chosen, the only possibility of connecting \( r_1 \) to \( i \) is via \( a \), since using another arc to \( F \) creates a cycle and an arc to \( Q \) implies that we leave the face. Hence, the inequality cannot induce a facet.

To prove sufficiency, let \( \tilde{F} \) be the face induced by (12.84) and assume that \( \tilde{F} \) is not a facet, but contained in a facet \( F' \supset \tilde{F} \) induced by the inequality

\[ \sum_{a \in A} \alpha_a y_a \leq \beta. \]  

(12.85)

Arbitrary take an ordering, \( \sigma(R) := (\sigma^R_1, \ldots, \sigma^R_{|R|}) \), of the nodes in \( R \) and an ordering, \( \sigma(Q) := (l = \sigma^Q_1, \ldots, \sigma^Q_{|Q|}) \), of the nodes in \( Q \). Define \( p^0 := p(\sigma(R), \sigma(Q)) \in \tilde{F} \) as follows. Set \( y_a = 1 \) for \( a := (i, j) \) if \( a \in F \cup \gamma(S, \{l\}) \cup \{(r_1, l)\} \) or \( i = \sigma^R_k, j = \sigma^Q_k \in R \) and \( k > k' \) or \( i = \sigma^Q_k, j = \sigma^Q_{k'} \in Q \) and \( k > k' \), i.e. \( p^0 \) induces the graph with all arcs in the forest, all arcs from \( S \) or \( r_1 \) to \( l \) and, in addition, for each node pair, \((i, j)\), in \( R \), there is an arc if \( j \) precedes \( i \) in the ordering \( \sigma(R) \), and similarly for \( Q \) and \( \sigma(Q) \). The support of \( p^0 \) is depicted in Figure 12.3. (This figure is likely vital to follow the proof.)

Take an arc \( a := (i, j) \in \gamma(Q, Q) \). Construct \( p \) from \( p^0 \) by removing \( a \), i.e. set \( y_a = 0 \). Then, \( p \in \tilde{F} \) and \( \alpha_a = 0 \), since \( p \) and \( p^0 \) differ only in this component. Since the ordering is arbitrary, we get \( \alpha_a = 0 \) for all \( a \in \gamma(Q, Q) \). Analogously, \( \alpha_a = 0 \) for all \( a \in \gamma(R, R) \).

Take an arc \( a \in \gamma(Q, R \cup S) \cup \gamma(S, R \cup Q) \setminus F \) and construct \( p \) from \( p^0 \) by adding the arc \( a \) if it does not exist and removing it if it does, i.e. set \( y_a \) to \( 1 - y_a \). Adding an arc potentially creates a cycle, but since the orderings \( \sigma(R) \) and \( \sigma(Q) \) are arbitrary, this is avoided by selecting other orderings. Then, \( p \in F \) and \( \alpha_a = 0 \).

The remaining potential non-zero components of \( \alpha \) in (12.85) and not in (12.84) are associated with an arc \( a \in \gamma(R \cup S, S) \setminus F \). We show that \( \alpha_a = 0 \) for these arcs as well by defining an auxiliary point \( p^{(j)} \) from \( p^0 \) for each \( j \in S \). For \( j \in S \), let \( j' \in N(F) \) be the unique successor of \( j \) in \( F \). Consider an arc \( a := (i, j) \in \gamma(R \cup S, S) \setminus F \). There are two cases.

First, if \( j' \neq r_1 \), define \( p^{(j)} \) from \( p^0 \) by removing arc \((j, j') \) and replacing the arc \((r_1, l) \) by \((r_1, j) \). Observe that \( p^{(j)} \in F \) since \((j, j') \in F \) and \((r_1, l) \in \gamma(R, Q) \). Furthermore, since \((j, l) \) was already in \( p^0 \) and \((r_1, j) \) was added, there is a path from \( r_1 \) to \( l \). If \( a \) and \( p^0 \) do not induce a cycle, create \( p \) from \( p^0 \) by adding \( a \). This yields \( p \in \tilde{F} \) and \( \alpha_a = 0 \) as above. If \( a \) and \( p^0 \) induce a cycle, instead create \( p \) from \( p^{(j)} \) by adding \( a \) to obtain \( p \in F \) and \( \alpha_a = 0 \) as before.

In the other case, \( j' = r_1 \) and \( R = \{r_1\} \) (if \( |R| > 1 \) just choose another ordering to obtain the case \( j' \neq r_1 \)). Since \( F \notin \mathcal{F} \), we have \( \delta^-(j) = 0 \) and there is an arc \((i', j) \in F \). Create \( p^{(j)} \) from \( p^0 \) by removing arc \((j, j') \) and replacing the arc \((r_1, l) \) by
12.4 Solving the Acyclic Ingraph Problem

Figure 12.3: An illustration of the construction of the point $p^0$ used in the proof of Proposition 12.19. In Figure 12.3a, the general construction is outlined. All nodes spanned by the forest have an arc to $l$. Some arcs connect directly to the set of root nodes, $R$, see also Figure 12.3b. In Figure 12.3c, the arcs in $R$ or $Q$ induced by the ordering $\sigma = \{1, 2, 3, 4, 5\}$ are shown.

(12.84) and (12.85) are the same. To see that $\alpha_a = \alpha_a'$ for $a, a' \in \gamma(R, Q)$, observe that the point $p^0 \in \bar{F}$ obtained by an appropriate selection of permutations implies that $\alpha' := (\alpha'_F, \alpha'_\gamma)$ satisfies the following system of equations,

$$1' \alpha_F + I \alpha_\gamma = 1 \beta,$$

where $1$ is the all-ones vector and $I$ is the identity matrix. Similarly, from the points $p^{(j)} \in \bar{F}$ for $j \in N(F) \setminus R$,

$$\alpha_F + (1' - I) \alpha_F = 1 \beta,$$

we get $\alpha_a = \alpha_a'$ for $a, a' \in \bar{F}$.

Finally, combining (12.86) and (12.87) gives

$$I \alpha_F + I \alpha_\gamma = 0,$$

and $\alpha_a = -\alpha_a'$ for $a \in F$ and $a' \in \gamma(R, Q)$.

We have shown that (12.84) is (12.85) up to scaling, so $\bar{F}$ must be a facet.  \qed
Take \( F \in \mathcal{F} \) and let \( r_1 \in R \) be the unique root in \( F \). We define \( S_0^r(F) \) to be the arcs from \( r_1 \) to some node dangling from \( r_1 \), i.e.

\[
S_0^r(F) := \{(r_1, i) \in A \mid i \in S_0(F) \text{ and } (i, r_1) \in F\}, \quad F \in \mathcal{F}.
\]

From the necessity part of the proof of Proposition 12.19 it follows that the dimension of the face induced by \( F \) is reduced by at least \(|S_0^r(F)| = |S_0(F)|\). From the sufficiency part it follows that it is only reduced by \(|S_0^r(F)|\) as long as \( S_0(F) \neq N \setminus \{r_1, l\} \), i.e. unless all nodes but \( r_1 \) and \( l \) are dangling from \( r_1 \). This shows that \( F \in \mathcal{F} \) induces a facet of the face of \( \text{conv} \mathcal{I}^l \) where the arcs in \( S_0^r(F) \) are restricted to be 0.

**Proposition 12.20**

Assume \( A = (N \setminus \{l\}) \times N \). Take \( F \in \mathcal{F} \) such that \( S_0(F) \neq N \setminus \{r_1, l\} \) and consider the face

\[
\bar{F}_0 := \{y \in \mathbb{R}^A \mid y_a = 0 \text{ for } a \in S_0^r(F)\}.
\]

Then, the inequality,

\[
\sum_{a \in F} y_a - \sum_{a \in \gamma(R,Q)} y_a \leq |F| - 1,
\]

induces a facet of \( \text{conv} \mathcal{I}^l \cap \bar{F}_0 \).

Based on this result, a facet of \( \text{conv} \mathcal{I}^l \) is obtained by lifting in the variables restricted to 0 in \( \bar{F}_0 \) associated with arcs in \( S_0^r(F) \).

**Proposition 12.21**

Assume \( A = (N \setminus \{l\}) \times N \). Take \( F \in \mathcal{F} \) such that \( S_0(F) \neq N \setminus \{r_1, l\} \). Then, the inequality,

\[
\sum_{a \in F} y_a - \sum_{a \in \gamma(R,Q)} y_a + \sum_{a \in S_0^r(F)} y_a \leq |F| - 1,
\]

induces a facet of \( \text{conv} \mathcal{I}^l \).

**Proof:** Apply sequential lifting of the variables in \( \bar{F}_0 \) from the base inequality, i.e. for \( S \subset S_0^r(F) \) and \( \bar{a} \in S_0^r(F) \setminus S \), solve the problem

\[
\text{maximize } \sum_{a \in F} y_a - \sum_{a \in \gamma(R,Q)} y_a + \sum_{a \in S} y_a
\]

subject to

\[
y_a = 1, \quad a \in S_0^r(F) \setminus S \cup \{\bar{a}\},
\]

\[
y_a = 0, \quad a \in S_0^r(F) \setminus S \cup \{\bar{a}\},
\]

\[
y \in \text{conv} \mathcal{I}^l,
\]
to determine the maximal lifting coefficient, $\alpha_a$ for $y_a$ as $\alpha_a = F - 1 - \zeta$ where $\zeta$ is the optimal value of (12.93), see e.g. [180].

Trivially, $\zeta \leq F - 1$. Denote the reversal of arc $\bar{a}$ by $\bar{a} \in F$. Since $y_a = 1$ and $y_a + y_{\bar{a}} \leq 1$, it follows that $\zeta \leq F - 1 - 1$. However, a solution that attains this value is easily constructed by taking all arcs in $F$ except $\bar{a}$ and appropriately augment it to an ingraph. Hence, $\zeta = F - 2$ and $\alpha_a = 1$.

Since $S \subseteq S_0^r(F)$ was arbitrary, (12.92) is a facet obtained by sequential maximal lifting from (12.91). (It is the only facet obtainable like this.)

We conclude the analysis of $\text{conv } \mathcal{T}^l$ by considering some separation problems. In the remainder of this section, let $\tilde{y} \in \mathbb{R}^A \cap [0, 1]^A$ be a point to be separated.

First we show that it is easy to separate the cycle inequalities. Indeed, we can solve

$$\begin{align*}
\text{minimize} & \quad \sum_{a \in C} (1 - y_a) \\
\text{subject to} & \quad C \in \mathcal{D}^l.
\end{align*}$$

If the optimal value in model (12.94) is less than 1, then there is a violated cycle inequality. Solving (12.94) is straightforward. It suffices to solve an all-pairs shortest path problem which can be done in $O(n(m + n \log n))$, see Section 2.1.

To separate inequalities associated with some $F \subseteq \mathcal{F}(R, N \setminus Q)$ for some $R$ and $Q$ we re-write (12.81) as

$$\sum_{a \in F} (1 - \tilde{y}_a) \geq 1 - \sum_{a \in \gamma(R, Q)} \tilde{y}_a. \quad (12.95)$$

Observe that a rooted forest becomes an anti-arborescence if it is augmented by arcs from the forest root nodes $r \in R$ to the ingraph root node $l$. By setting the cost on such arcs to 0, we see that the left hand side in (12.95) can be minimized by solving the problem

$$\begin{align*}
\text{minimize} & \quad \sum_{a \in F} (1 - \tilde{y}_a) \\
\text{subject to} & \quad F \in \mathcal{F}(R, N \setminus Q).
\end{align*}$$

This problem can be solved as a minimum cost anti-arborescence problem in an auxiliary graph where all nodes in $Q$ except $l$ are removed and the arcs from a node in $R$ are freely connected to the root $l \in Q$, i.e. with cost 0. Hence, for fixed $R$ and $Q$, the inequality (12.95) is violated for some $F \subseteq \mathcal{F}(R, N \setminus Q)$ if and only if the optimal value in (12.96) is less than the (fixed) right hand side in (12.95). As mentioned before, several polynomial time algorithms exist for solving this problem, see for instance the references in the proof of Theorem 12.4.

**Proposition 12.22**

*The separation problem for the forest inequalities (12.81) for $F \subseteq \mathcal{F}(R, N \setminus Q)$ can be solved in polynomial time when $Q$ and $R$ are fixed.*
The construction of the auxiliary graph for the case where $R$ and $Q$ are fixed will be used also to derive a formulation for the separation problem when $R$ and $Q$ are not fixed.

We base our model on the ordinary arborescence formulation, see e.g. (11.19), and include variables that indicate which set each node belongs to. For $i \in N$, let $r_i$ and $q_i$ be binary variables to be set to 1 if $i \in R$ and $i \in Q$, respectively. Also, for $a := (i, j) \in A$, let $z_a$ indicate if $r_i = q_j = 1$. We claim that there exist $Q \subseteq N$ and $R \subseteq N$ and an $F \in F(R, N \setminus Q)$ associated with a violated inequality if and only if the optimal value of the following problem is less than 1.

\[
\begin{align*}
\text{minimize} & \quad \sum_{a \in A \setminus \delta^- (l)} (1 - y_a) y_a + \sum_{a \in A} y_a z_a \\
\text{subject to} & \quad \sum_{a \in \delta^+(i)} x^k_a - \sum_{a \in \delta^-(i)} x^k_a = \begin{cases} 
1 - q_i, & \text{if } i = k, \\
q_i - 1, & \text{if } i = l, \\
0, & \text{otherwise},
\end{cases} \quad i \in N, \ k \in N, \quad (12.97a) \\
& \quad x^k_a \leq y_a^l, \quad a \in A, \ k \in N, \quad (12.97b) \\
& \quad r_i + q_i \leq 1, \quad i \in N, \quad (12.97c) \\
& \quad z_a \geq r_i + q_j - 1, \quad a := (i, j) \in A, \quad (12.97d) \\
& \quad y_a \leq 1 - q_i, \quad a \in \delta^- (i), \ i \in N, \quad (12.97e) \\
& \quad y_a = r_i, \quad a := (i, l) \in \delta^- (l), \quad (12.97f) \\
& \quad q_i = 1, \quad (12.97g)
\end{align*}
\]

We show that for binary values of $r$ and $q$, model (12.97) becomes the arborescence problem in (12.96) for the case where $R$ and $Q$ are fixed. The motivation is as follows. Constraint (12.97c) ensures that each node is in at most one of $R$ and $Q$. Due to (12.97a), there is a path from a node not in $Q$ to $l$. Hence, (12.97a)-(12.97c) implies that the $y$-variables induce an arborescence to $l$ that spans all nodes not in $Q$. Then, constraint (12.97e) make sure that no arcs out from a node in $Q$ are used and constraint (12.97f) forces each root node in $R$ to connect to $l$. Due to (12.97d), the value of $z_a$ for $a := (i, j)$ in an optimal solution is $r_i q_j$, when $y_a > 0$. Finally, the first part of the objective measures the cost induced by forest arcs and the second part measures the cost induced by the cut between nodes in $R$ and nodes in $Q$.

From the formulation of the forest inequality in (12.95), it follows that the objective must not exceed 1 if all inequalities associated with an $F \in F(R, N \setminus Q)$ are satisfied.

**Proposition 12.23**

There is a violated forest inequality (12.81) for an $F \in F(R, N \setminus Q)$ if and only if the optimal value of (12.97) is less than 1.

The right hand side in (12.95) must not be greater than or equal to 1. Since it decreases with the size of $R$ and $Q$, it seems likely that these sets should be of small cardinality. Hence, in practice, it might suffice to solve the separation problem for either $R$ or $Q$ fixed.
for a small number of candidate sets, e.g. all $Q = \{i, l\}$ or $R = \{i\}$ for some $i \in N$. Fortunately, the separation problem (12.97) can be solved efficiently in this important special case, i.e. when $Q$ or $R$ is fixed.

When $Q$ is fixed, (12.97) reduces to a problem that can be solved as an ordinary arborescence problem on an auxiliary graph as follows. Remove all nodes in $Q$. Introduce an auxiliary node $\hat{r}$ as the root, i.e. it represents the set $Q$. If $r_i = 1$ and $q_j = 1$ in the the original problem, then the associated arc $a := (i, j) \in \gamma(R, Q)$ induces cost $\tilde{y}_a$ via $\tilde{c}_a$. The total cost induced by setting $r_i = 1$ is obtained by summing over $j \in Q$. We put the total cost on an auxiliary arc from $i$ to $\hat{r}$. Hence, the resulting arc costs $\tilde{c}$ are given by,

$$
\tilde{c}_a := \begin{cases} 
1 - \tilde{y}_a, & \text{if } a = (i, j) \notin \delta^-(\hat{r}), \\
\tilde{y}_a, & \text{if } a = (i, \hat{r}) \in \delta^-(\hat{r}).
\end{cases}
$$

(12.98)

When $R$ is fixed, the situation is slightly more complicated. Given a node $j \in N$, there are two cases to consider. First, if $q_j = 0$, i.e. $j \notin Q$ in (12.97), then all constraints involving $q_j$ are satisfied. The complication arises when $q_j = 1$, i.e. $j \in Q$. Then, $z_a = r_i$ for $a := (i, j) \in \delta^-(j)$. Also, no arc is allowed to enter the node $j$.

To model the above, we set up a problem where not all arborescences are allowed as follows. Let the existence of a direct arc from a non-root node $i$ to $l$ mean that node $i$ is assigned to $Q$. This implies that $z_a = 1$ for all $a \in \gamma(R, \{i\})$ and than no arc may enter node $i$. The interpretation of all other arcs remain intact. Hence the costs $\bar{c}$ assigned to arcs are defined as,

$$
\bar{c}_a := \begin{cases} 
1 - \tilde{y}_a, & \text{if } a \notin \delta^-(l), \\
\tilde{y}_a, & \text{if } a \in \delta^-(l).
\end{cases}
$$

(12.99)

To guarantee that no node $i$ assigned to $Q$ has en entering arc, we prohibit flow for commodities not associated with node $i$, i.e. for each $a := (i, l) \in \delta^-(l)$ and each $i \neq k$ we set $x_a = 0$. This yields,

\[ \begin{align*}
\text{minimize} & \quad \sum_{a \in A} \bar{c}_a y_a \\
\text{subject to} & \quad \sum_{a \in \delta^+\{i\}} x_a^k - \sum_{a \in \delta^-\{i\}} x_a^k = b_k^i, \quad i \in N, \; k \in N \smallsetminus \{l\}, \\
& \quad x_a^k \leq y_a^l, \quad a \in A, \; k \in N, \\
& \quad x_a^k = 0, \quad i \neq k, \; a := (i, l) \in \delta^-(l), \\
& \quad x \in \mathbb{R}^{A \times N}, \; y \in \mathbb{B}^A.
\end{align*} \]

(12.100)

where $b_k^i$ are the usual node balances, i.e.

$$
\begin{cases} 
1, & \text{if } i = k, \\
-1, & \text{if } i = l, \\
0, & \text{otherwise}.
\end{cases}
$$

(12.101)
Proposition 12.24  
The LP-relaxation of model (12.100) defines an integral polytope.

Proof: The original arborescence formulation, i.e. model (12.100) without constraints (12.100c), is integral. Hence, restricting some variables to 0 in (12.100) yields a face of an integral polytope, which is again an integral polytope. \[ \square \]

Proposition 12.25  
The separation problem for the forest inequalities (12.81) for \( F \in \mathcal{F}(R, N \setminus Q) \) can be solved in polynomial time when \( Q \) or \( R \) is fixed.

This concludes our analysis of \( \text{conv} \, I^I \).

12.4.2 Heuristics for the Acyclic Ingraph Problem

Acyclic ingraphs induce partial orders. We present a class of heuristics that is based on searching in the space of total orders, or permutations. The basic idea is that it is easy to find an optimal ingraph that is consistent with a given total ordering.

Let \( \sigma = (\sigma_1, \ldots, \sigma_n = l) \), be a permutation. We say that an arc \( a := (i, j) \) is consistent with \( \sigma \) if \( i < j \) in \( \sigma \). An ingraph, \( I^* \in \mathcal{I}^I \), is consistent with \( \sigma \) if all its arcs are consistent with \( \sigma \), i.e. \( a := (i, j) \in I^* \) only if \( i < j \) in \( \sigma \). It is straightforward to find an optimal ingraph \( I^* \) consistent with \( \sigma \). Indeed, for each node \( i \in N \setminus \{l\} \) it is optimal to include \( a \in \delta^+(i) \) in \( I^* \) if \( c_a < 0 \). If \( c_a \geq 0 \) for all \( a \in \delta^+(i) \), it suffices to include an arc with smallest cost in \( I^* \).

Based on this observation it is possible to develop a large class of heuristics by searching in the space of permutations. Many ideas can be found in the scheduling and routing literature. Some natural neighborhoods for permutations include insertion, exchanges and inversion and \( k \)-opt. The latter was introduced in [164] for the classical travelling salesman problem where it has been shown to be successful, in particular for \( k \in \{2, 3\} \). The famous Lin-Kernighan heuristic [165] is an improvement where \( k \) changes.

As mean for intensification, we consider a large-neighborhood-move as follows, see e.g. [3] for a survey of the large neighborhood search meta-heuristic. An ingraph \( I \in \mathcal{I}^I \) induces a partial order. A permutation \( \sigma \) that is compatible with this partial order is called a linear extension. Any linear extension again gives an optimal consistent ingraph. It is NP-hard to find the best linear extension. (Indeed, take the partial order induced by \( I = \delta^-(l) \), then the best linear extension solves the original problem.)

A heuristic algorithm for finding a linear extension for an ingraph \( I \) can be based on the following observation. Suppose that \( i \) and \( j \) are incomparable in the partial order, i.e. there is no path between \( i \) and \( j \) in the ingraph \( I \). Then, the reduction in the objective induced by having \( i \) before \( j \) in the total order is \( \min\{c_a, 0\} \), where \( a := (i, j) \). For a leaf node \( i \), with indegree 0, the total reduction is determined by summing \( \min\{c_a, 0\} \) over arcs \( a \in \delta^+(i) \) to nodes that are incomparable with \( i \). Our algorithm constructs a linear extension recursively from \( I \) by considering all leaf nodes and removing the best one.

We propose to use meta-heuristics that have been successfully applied to other problems with the underlying permutation structure in combination with some neighborhood mentioned above and our intensification procedure.
Separation of SPR Inequalities and Computational Aspects

A CRUCIAL ISSUE in cutting plane based methods is to find violated inequalities, i.e. to solve the separation problem. In this chapter we derive some heuristic and exact separation algorithms for valid inequalities for $\text{conv } \mathcal{Y}$ and $\text{conv } \mathcal{Y}$, i.e. the set of feasible routing patterns introduced in Chapter 10. Our models and algorithms are based on the characterization of shortest path routing (SPR) conflicts in Part II. In our opinion, SPR conflicts are more comprehensible in the unique shortest path routing (USPR) case, therefore this will be our focus. Unless explicitly stated, algorithms applies only to the USPR case.

Let $\mathcal{Q}$ be the collection of SPR inequalities in the USPR case. The separation problem for the class of SPR inequalities induced by $\mathcal{Q}$ can be formulated as follows. Given a tentative fractional routing pattern $\bar{y} \in \mathbb{Q}^{A \times L}$, find a family of cycles $\mathcal{C} := \{C^i\} \in \mathcal{Q}$, such that the corresponding SPR inequality,

$$
\sum_{l \in L} \sum_{a \in B(C^i)} \bar{y}_{la} \leq \sum_{l \in L} |C^i| - 1,
$$

is violated. Such a violated inequality exists if and only if the separation problem,

$$
\begin{align*}
\text{minimize} \quad & \sum_{l \in L} \sum_{a \in C^i} (1 - \bar{y}_{la}) \\
\text{subject to} \quad & \mathcal{C} \in \mathcal{Q},
\end{align*}
$$

has an objective value less than 1, where the minimum is taken over all possible families of cycles in $\mathcal{Q}$.

**Outline** We first consider the heuristic separation of general routing conflict inequalities in Section 13.1 by adapting some models from Part II. In Section 13.2, we derive a model...
for the exact separation of general routing conflict inequalities. Then, in Section 13.3, we develop several efficient separation algorithms for separating inequalities based on valid cycles and subpath inconsistency. We conclude in Section 13.4 by outlining a computational scheme for SPR problems, where we also discuss some computational aspects.

13.1 Heuristic Separation of Routing Conflicts

The models in Part II can be used to determine if the binary part of a tentative routing pattern is realizable, or more appropriately, partially realizable. This is an excellent starting point for deriving a heuristic separation algorithm; Indeed, it suffices to round $\tilde{y}$ to obtain a (partially) binary routing pattern. See e.g. [50].

Let $\epsilon_0 < \epsilon_1$ be binary rounding thresholds and treat $y_{la}$ as 1, or an SP-arc, if $y_{la} \geq \epsilon_1$ and as 0, or a non-SP-arc, if $y_{la} \leq \epsilon_0$, and neglect it otherwise. Suggested values are $\epsilon_0 = 0.1$ and $\epsilon_1 = 0.7$. Based on $\epsilon_0, \epsilon_1$ and $\tilde{y}$, the induced family of SP-graphs

$$A_L := A_L(\epsilon_0, \epsilon_1, \tilde{y})$$

is defined from its SP-graphs via the relation,

$$A_l(\tilde{y}) := \left\{ a \in A \mid y_{la} \geq \epsilon_1 \right\}$$

$$A_l(\tilde{y}) := \left\{ a \in A \mid y_{la} \leq \epsilon_0 \right\}.$$

From $A_l(\hat{y})$ and $\tilde{A}_l(\hat{y})$, the D-arcs $D_l(\hat{y})$ are defined in the natural way, i.e.

$$D_l(\hat{y}) := \left\{ (i, l) \in N \times N : i \neq l, \delta^+(i) \cap A_l(\hat{y}) = \emptyset \right\}.$$

Since there will eventually be a path from every node to every destination, it is natural to set the value of $\hat{y}$ to 1 for D-arcs, i.e.

$$\hat{y}_{la} = 1, \quad a \in D_l(\hat{y}), \quad l \in L.$$

As before, $\tilde{D}$ denotes the set of all D-arcs and $\tilde{A}$ the union of ordinary arcs and D-arcs. We also use $AD_l$ to denote all SP-arcs (including D-arcs) to l, i.e.

$$AD_l(\hat{y}) := A_l(\hat{y}) \cup D_l(\hat{y}).$$

A natural heuristic separation algorithm is to determine if the tentative routing pattern induced by $\hat{y}$ is feasible by solving one of the LPs from Part II. We suggest to solve a slightly modified version of model (5.13) on page 63 (or (10.19) on page 156); To also take the fractional values of $\hat{y}$ into account to some extent, we replace the right hand sides by $1 - \hat{y}$.

```
find w
subject to
w_a + \pi_i - \pi_j = 1 - \hat{y}_{la}, \quad a := (i, j) \in AD_l(\hat{y}), \quad l \in L, \quad (13.8a)
w_a + \pi_i - \pi_j \geq 1 - \hat{y}_{la}, \quad a := (i, j) \in \tilde{A} \setminus AD_l(\hat{y}), \quad l \in L, \quad (13.8b)
w_a \geq 1, \quad a \in \tilde{A}. \quad (13.8c)
```
Observe that the reduced cost constraints for non-SP-arcs and unrestricted arcs in (13.8) have been combined in constraint (13.8b). Also note that models (13.8) and (5.13) or (10.19) coincide when $\bar{y}$ is binary. To use (13.8) as a model for the separation problem we consider its Farkas system and make it an optimization problem.

\[
\begin{align*}
\text{minimize} & \quad \sum_{l \in L} \sum_{a \in \tilde{A}} \bar{y}_a \theta^l_a \\
\text{subject to} & \quad \sum_{a \in \delta^+(i)} \theta^l_a - \sum_{a \in \delta^-(i)} \theta^l_a = 0, \quad i \in N, \ l \in L, \quad (13.9a) \\
& \quad \sum_{l \in L} \theta^l_a \leq 0, \quad a \in \tilde{A}, \quad (13.9b) \\
& \quad \theta^l_a \geq 0, \quad a \in \tilde{A} \setminus AD^l(\bar{y}), \ l \in L. \quad (13.9c)
\end{align*}
\]

The feasible region in model (13.9) is a convex polyhedral cone and the objective is either 0 or unbounded from below. In the latter case, it follows in particular that the sum of unrestricted variables associated with SP-arcs must be unbounded from below, i.e.

\[
\sum_{l \in L} \sum_{a \in AD^l(\bar{y})} \theta^l_a \rightarrow -\infty. \quad (13.10)
\]

The feasible region in (13.9) can be restricted to bound this sum. This can be accomplished in several ways. Some restrictions preserve a precise correspondence between extreme rays in the cone and extreme points in the polyhedron induced by the restriction. Choosing the restriction is referred to as normalization and has recently proved to be of utmost importance in several cut generation procedures, see e.g. [20, 102, 103, 189].

We bound the sum in (13.10) from below by -1. In combination with a minor adjustment of the objective, we will argue that this normalization more likely provides solutions that correspond to conflicts of small cardinality and large violation. We consider the modified problem,

\[
\begin{align*}
\text{minimize} & \quad \sum_{l \in L} \sum_{a \in A} \bar{y}_a \theta^l_a - \epsilon \sum_{l \in L} \sum_{a \in AD^l(\bar{y})} \theta^l_a \\
\text{subject to} & \quad \sum_{a \in \delta^+(i)} \theta^l_a - \sum_{a \in \delta^-(i)} \theta^l_a = 0, \quad i \in N, \ l \in L, \quad (13.11a) \\
& \quad \sum_{l \in L} \theta^l_a \leq 0, \quad a \in \tilde{A}, \quad (13.11b) \\
& \quad \theta^l_a \geq 0, \quad a \in \tilde{A} \setminus AD^l(\bar{y}), \ l \in L, \quad (13.11c) \\
& \quad \sum_{l \in L} \sum_{a \in AD^l(\bar{y})} \theta^l_{ij} \geq -1. \quad (13.11d)
\end{align*}
\]
The general rationale of our choice is based on [103]. The support of vertices of the polyhedron resulting after a proper normalization corresponds to minimal infeasible subsystems [117]. Further, the objective and normalization can be changed [82] in the sense that the objective is bounded by a constraint and the normalization constraint is used in the objective. Hence, to obtain minimally infeasible subsystems, the normalization where the 1-norm is bounded by 1 is used. To heuristically reduce the cardinality of the infeasible subsystem, the objective is augmented by a second sum that favors solutions with few negative \( \theta \), i.e. where the negative support has small cardinality. In the Benders’ context, it was observed in [103] that there are some normalization coefficients that should be set to 0 since they correspond to constraints in the Benders’ subproblem that are satisfied anyway (in our context, these are the reduced cost non-negativity constraints, which correspond to \( \theta \)-variables that must be non-negative). The computations in [103] showed that such a normalization is superior.

We also illustrate the connection between maximizing the violation in (13.2) and the heuristic separation problem (13.11). Consider a unitary solution \( \theta \), i.e. all non-negative absolute values of \( \theta \) are equal. When \( \epsilon \) is set to 0, the objective value in (13.11) is

\[
\eta := \sum_{l \in L} \sum_{a \in A} \bar{y}_a \theta^l_a = \sum_{l \in L} \sum_{a \in B(C^l)} \theta^l_a \bar{y}_a + \sum_{l \in L} \sum_{a \in F(C^l)} \theta^l_a \bar{y}_a. \tag{13.12}
\]

Recall that \( \theta^l_a < 0 \) when \( a \in B(C^l) \) and \( \theta^l_a > 0 \) when \( a \in F(C^l) \). Since \( \theta \) is a unitary solution, \( \eta \) is proportional to

\[
- \sum_{l \in L} \sum_{a \in B(C^l)} \bar{y}_a + \sum_{l \in L} \sum_{a \in F(C^l)} \bar{y}_a \leq - \sum_{l \in L} \sum_{a \in B(C^l)} \bar{y}_a, \tag{13.13}
\]

where \( C := \{ C^l \} \subseteq Q \) is the cycle family corresponding to \( \theta \). Therefore, minimizing in (13.11) can be expected to do good in maximizing the left hand side in (13.1), i.e.

\[
\sum_{l \in L} \sum_{a \in B(C^l)} \bar{y}_a, \tag{13.14}
\]

over cycle families. Also, since \( \theta \) is optimal, it is likely that the slack in (13.13) is small. Indeed, we minimize and the slack contributes to the objective by a positive amount. Hence, \( \eta \) seems to correlate well to the violation of the inequality associated with its corresponding cycle family.

The purpose of the sum in the objective in (13.11) involving \( \epsilon \) is to heuristically reduce the cardinality of the negative support of \( \theta \). Since the negative support is contained in \( AD^l(\bar{y}) \) for \( l \in L \), the total contribution to the objective is positive when we multiply by \(-\epsilon \). Hence, the secondary objective of minimizing the support is to some extent taken into account.

### 13.2 Exact Separation of Routing Conflicts

To solve the separation problem (13.2), i.e. finding a most violated inequality of the form (13.1), we modify model (13.11) for the heuristic separation problem.
There are three issues with the connection between the violation and the objective in (13.11). First, our argument about proportionality relied on unitary solutions. Second, the size of the support of the solution, i.e. the cycle family, was only heuristically taken into account. Finally, only negative θ-variables contribute to the violation.

We resolve these issues in the model below. Take $0 < \epsilon \leq 1$ sufficiently small. (This ε is not related to ε in Section 13.1) Let $\zeta$ be a binary variable that indicates if there is a violated inequality. When $\zeta = 1$, we should model a cycle family $C := \{C_i\} \in \mathcal{Q}$ that induces a violated inequality. We use binary $y$-variables to indicate if an arc is an SP-arc in $C$. For each $(a, l) \in \hat{A} \times L$, set $y^l_a = 1$ if and only if $(a, l) \in B(C^l)$. As in model (13.11), the θ-variables model arc flow. The flow has to be circulating and the total capacity must not exceed 0. The flow $\theta^l_a$ on arc $a \in A$ for destination $l \in L$ may be negative if and only if the corresponding arc is an SP-arc, i.e. when $y^l_a = 1$.

\[
\text{maximize} \quad \sum_{l \in L} \sum_{a \in A} (\hat{y}^l_a - 1)y^l_a + \zeta
\]

subject to

\[
\sum_{l \in L} \sum_{a \in A} y^l_a \geq \zeta, \quad (13.15a)
\]

[Frac-SEP]

\[
\sum_{a \in \delta^+(i)} \theta^l_a - \sum_{a \in \delta^-(i)} \theta^l_a = 0, \quad i \in N, \ l \in L, \quad (13.15b)
\]

\[
\sum_{l \in L} \theta^l_a \leq 0, \quad a \in \hat{A}, \quad (13.15c)
\]

\[
y^l_a \leq \theta^l_a, \quad a \in \hat{A}, \ l \in L, \quad (13.15d)
\]

\[
\theta^l_a + (1 + \epsilon)y^l_a \leq 1, \quad a \in \hat{A}, \ l \in L, \quad (13.15e)
\]

\[
\theta \in \mathbb{R}^{\hat{A} \times L}, \ y \in \mathbb{B}^{\hat{A} \times L}, \ z \in \mathbb{B}. \quad (13.15f)
\]

The explanation and motivation of correctness is as follows. If $\zeta = 0$, then the objective and all variables are 0 and there is no cycle family $C$ that induces a violated inequality of the form (13.1). Assume $\zeta = 1$. The objective measures the violation of the cycle family $C$ induced by the $y$-variables; Since we now maximize, the contribution to the objective in (13.2) is $(1 - \hat{y}^l_a)$ when $(a, l) \in B(C^l)$, i.e. if $y^l_a = 1$. Further, $\zeta$ is used to handle the $-1$ in the right hand side in (13.1).

It remains to show that the connection between θ-variables and $y$-variables is valid, i.e. that $\theta^l_a < 0$ if and only if $y^l_a = 1$. This is handled in constraints (13.15d)-(13.15e) by defining a box around $\theta$ based on $y$ and $\epsilon$. When $0 < \epsilon \leq 1$ and $y^l_a = 1$, the feasible values for $\theta^l_a$ is $-1 \leq \theta^l_a \leq -\epsilon$. Hence, when $\epsilon$ is sufficiently small, the ratio between different θ-variables can be arbitrary, and hence all general solutions are allowed in (13.15). In the special case where $\epsilon = 1$, the connection between $y^l_a$ and $\theta^l_a$ is much stronger, then $\theta^l_a = -y^l_a$. This restriction only allows unitary solutions.

We have shown the following theorem and corollary.

**Theorem 13.1**

For sufficiently small $\epsilon > 0$, model (13.15) has an optimal solution with strictly positive objective value if and only if $y$ violates some SPR inequality of the form (13.1).
Corollary 13.1
For \( \epsilon = 1 \), model (13.15) has an optimal solution with strictly positive objective value if and only if \( \bar{y} \) violates some SPR inequality of the form (13.1) associated with a unitary cycle family.

We consider some issues related to solving model (13.15). Let \( \Delta \) be the minimum number of arcs that we require in a cycle family. For instance, if we are only interested in solutions more complicated than valid cycles, it is feasible to set \( \Delta = 6 \) since non-dominated valid inequalities associated with a unitary solution have at least 6 ordinary arcs. (This is a non-trivial consequence of Example 6.3 and the characterization of domination in Section 10.3.2.) It is reasonable to only separate routing conflicts more complicated than valid cycles using (13.15) since there are more efficient combinatorial separation algorithm for valid cycle inequalities, see Section 13.3.

Given \( \Delta \), the right hand side in (13.15a) can be strengthened, i.e.

\[
\sum_{l \in L} \sum_{a \in A} y^l_a \geq \Delta \zeta. \tag{13.16}
\]

Model (13.15) is weak in the sense that it is feasible to set \( y^l_a = 1/(1 + \epsilon) \) for any \((a, l) \in A \times L\) in the LP-relaxation. A solution of this form is often optimal, it suffices that there are \((1 + \epsilon)\Delta\) (ordinary) arc destination pairs where \( y^l_a = 1 \).

Proposition 13.1
Let \( \bar{n} := (1 + \epsilon)\Delta \). Assume that \(|\{(a, l) \in A \times L \mid \bar{y}^l_a = 1\}| \geq \lceil \bar{n} \rceil \). Then, the optimal value to the LP-relaxation of (13.15) is 1.

Proof: There is a feasible solution where \( \theta^l_a = 0 \) for all \((a, l) \in A \times L\), \( \zeta = 1 \), and \( \lceil \bar{n} \rceil \) \( y \)-variables are set to \( 1/(1 + \epsilon) \). By the assumption, these can be chosen to have objective coefficient 0.

Setting \( \epsilon = 1 \) implies that all valid inequalities corresponding to unitary solutions can be separated. We focus on this case and w.l.o.g assume that \( \zeta = 1 \). Proposition 13.1 implies that \( 2\Delta \) of the \( y \)-variables are set to \( 1/2 \). Observe that there is a lot of symmetry in the LP-relaxation of (13.15) since any choice of \( 2\Delta \) of the \( y \)-variables suffices which is disastrous if there are many (ordinary) arc destination pairs where \( y^l_a = 1 \).

We consider some approaches to strengthen (13.15) via preprocessing. The optimal value of (13.15) is in the range 0 to 1. Hence, the knapsack constraint

\[
\sum_{l \in L} \sum_{a \in A} (\bar{y}^l_a - 1) y^l_a > -1 \iff \sum_{l \in L} \sum_{a \in A} (1 - \bar{y}^l_a) y^l_a < 1, \tag{13.17}
\]

is valid. This constraint has many large coefficients in comparison to its right hand side which can be seen by calculating the average value of all coefficients,

\[
\frac{1}{|L||A|} \sum_{l \in L} \sum_{a \in A} (1 - \bar{y}^l_a) = \frac{|A| - |N| + 1}{|A|} = 1 - \frac{|N| - 1}{|A|}. \tag{13.18}
\]

This value is typically between 0.5 and 1, and often closer to the latter. This implies that the conflict graph induced by the knapsack constraint is very dense which makes for
13.3 Separation of Valid Cycle Inequalities

many preprocessing possibilities, see Chapter 10 in [1], in particular Section 10.2, 10.6 and 10.7.

Remark 13.1. Strong inequalities can be derived from (13.17). However, they typically do not cut off the optimal LP solution at the root in the enumeration tree since solutions are of the form in Proposition 13.1.

We believe that the potential advantage of the density of the knapsack constraint should be used in probing. Recall that probing means to fix a variable and analyze its implications. It is a general, very powerful but costly technique used in preprocessing.

Consider the consequences of setting \( y_{la} = 1 \) in (13.15). This is quite restrictive. It implies \( \theta_a = -1 \), which in turn forces the existence of a \( \theta \)-cycle. The negative \( \theta \)-variables in the cycle in turn imply that some other \( y \)-variables must take positive values. Further, the capacity constraint must be satisfied, which forces some other \( \theta \)-cycles. They in turn imply that even more \( y \)-variables must take positive values. In all, setting \( y_{la} = 1 \) can have large impact and since it forces several \( y \) to be set to 1. This potentially implies that a solution is found or that the impact on the objective value in (13.15) allows fixing \( y_{la} = 0 \). The major drawback with this approach is that a large LP problem has to be solved, but it still seems much better than having to deal with the symmetry issue.

We consider how the analysis above can be speed up by only taking into account the implications for one \( \theta \)-cycle. If \( y_{la} = 1 \), then \( \theta_a = -1 \) and there must be a \( \theta \)-cycle including \( a := (s,t) \). To obtain a lower bound on the impact of such a cycle, we find a shortest path from \( t \) to \( s \) where arcs can be used forwards with cost \( 1 - y_{la} \) and backwards with cost \( 1 - y_{la} \) for some \( l' \in L \setminus \{l\} \) and backwards with cost \( 1 - y_{la} \), where \( a'' \) is the reversal of arc \( a' \). This approach is computationally efficient, but it will not be able to fix \( y_{la} \) at 0 if almost all arcs have some commodity where \( y_{la} = 1 \) since such arcs have cost 0 in the shortest path problem from \( t \) to \( s \).

This concludes our analysis of the separation of valid inequalities for general routing conflicts. As stated earlier, the most important conflicts in practice correspond to valid cycles. We give efficient algorithms to separate their associated valid inequalities next.

13.3 Separation of Valid Cycle Inequalities

In Section 7.1 we motivated that the practical importance of valid cycle conflicts, and hence valid cycle inequalities that prevent them, is indisputable. Also, valid inequalities based on valid cycles subsumes all combinatorial cuts presented in the literature, e.g. in [46, 63, 73, 85, 213, 218]. We also demonstrated that the class of routing conflicts associated with valid cycles is very rich and comprehensive. This allows for the derivation of combinatorial algorithms that are much more efficient than solving the general separation problems in the previous sections.

The least complicated routing conflicts correspond to directed cycles and subpath inconsistencies. They form subclasses of the valid cycle conflicts. We first consider the separation problem for these subclasses.
13.3.1 Directed Cycle and Subpath Inconsistency Separation

For each \( l \in L \), let \( D^l \) be the collection of directed cycles not containing node \( l \). For each \( s, t \in N \) we denote the set of \((s, t)\)-paths by \( P_{st} \).

A directed cycle inequality can be written as

\[
\sum_{a \in C} (1 - \bar{y}_a^l) \geq 1, \quad C \in D^l. \tag{13.19}
\]

From this representation it is easily seen that a most violated inequality is obtained by finding a shortest cycle in \( G \) w.r.t. arc costs \( 1 - \bar{y} \). This problem is efficiently solved since all arc costs are non-negative.

Remark 13.2. The separation of directed cycle applies also to the ECMP case. If it is assumed that \( S_l = N \setminus \{l\} \) in the USPR case, then it is possible to instead separate lifted directed cycle inequalities. Indeed, re-writing them using the outdegree equalities, these constraints become cut inequalities that can be separates using the standard subtour elimination separation algorithm, see e.g. [13].

To describe subpath inconsistency conflicts we take two destinations, \( l', l'' \in L \). Then, a conflict can be seen as two arc-disjoint paths with the same origin and the same destination. For a node pair \((s, t)\) \( \in N \times N \), and two paths \( P', P'' \in P_{st} \) where \( P' \neq P'' \) this gives the subpath inconsistency inequality

\[
\sum_{a \in P'} (1 - y_{s}^{l'}) + \sum_{a \in P''} (1 - y_{s}^{l''}) \geq 1. \tag{13.20}
\]

To find a most violated subpath inconsistency inequality, it suffices to find the shortest \((s, t)\)-path \( P' \) w.r.t. arc costs \( 1 - \bar{y}^{l'} \) and the shortest \((s, t)\)-path \( P'' \) w.r.t. arc costs \( 1 - \bar{y}^{l''} \). See also [50, 218].

We choose to treat directed cycle and subpath inconsistency inequalities together since they can both be represented by two paths. This implies that we can essentially separate the inequalities simultaneously. For each \( l \in L \) and each \((s, t)\) \( \in N \times N \), we define \( d_{st}^l \) as the shortest path distance from \( s \) to \( t \) w.r.t. arc costs \( 1 - \bar{y}^l \), i.e.

\[
d_{st}^l := \min_{P \in P_{st}} \sum_{a \in P} (1 - y_a^l), \tag{13.21}
\]

All distances can easily be found by e.g. using Floyd–Warshall’s algorithm or Johnson’s algorithm, see Section 2.1. Calculating \( d_{st}^l \) is the bottleneck operation in the separation algorithms described below, i.e. for directed cycles and subpath inconsistency conflicts.

Based on \( d_{st}^l \), a most violated cycle inequality is found by evaluating \( d_{st}^l + d_{ts}^l \) for all \( l \in L \) and \((s, t)\) \( \in N \times N \). This yields \( O(|L|n^2) \) combinations. Similarly, a most violated subpath inconsistency inequality is found by evaluating \( d_{st}^{l'} + d_{ts}^{l''} \) for all \( l', l'' \in L \) and \((s, t)\) \( \in N \times N \). This yields \( O(|L|^2n^2) \) combinations.

If the distances \( d_{st}^l \) are available, it is possible to speed up the algorithm for finding a most violated subpath inconsistency inequality. Let \( d_{st}^{l'} \leq d_{st}^{l''} \) be the two smallest values of \( d_{st}^l \) over \( l \in L \). Then, for each \( l \in L \), a most violated subpath inconsistency inequality is found by evaluating \( d_{st}^l + d_{ts}^{l'} \), where
\[ d_{st}^* := \begin{cases} d_{st}^{d_1}, & \text{if } d_{st}^d \neq d_{st}^d, \\ d_{st}^{d_2}, & \text{otherwise.} \end{cases} \]  

(13.22)

This algorithm improves over the trivial algorithm above since it only yields \( O(\sqrt{J/N^2}) \) combinations.

### Separation of Inequalities with D-arcs

Recall from Section 10.3.2 that all non-dominated inequalities with a D-arc in the USPR case are associated with an \((i;l)\) path \( P \). Let \( a \) be the first arc in \( P \). There are two cases.

First, if \( P \) is combined with a D-arc \( \sim a := (i;l) \), it yields the inequality

\[ y_l^a + \sum_{a \in B^r(C)} (1 - y^p_a) \geq 1, \]  

(13.23)

In the second case, \( \tilde{P} \) is combined with another \((i;l)\)-path \( \tilde{P}^i_l = \{a, \tilde{a}\} = \{(i,j),(j,l)\} \), where \( \tilde{a} \neq a \). This yields the inequality

\[ \sum_{a \in B^r(C)} (1 - y^p_a) - y^l_a \geq 0. \]  

(13.24)

This inequality can be lifted as in Example 10.4 to

\[ \sum_{a \in B^r(C)} (1 - y^p_a) - \sum_{a \notin \tilde{a}^\gamma(i)} y^l_a \geq 0. \]  

(13.25)

To separate path-inequalities of form (13.23), it suffices to use the computed values of \( d_{st}^d \) in (13.21). A most violated inequality is found by minimizing \( \tilde{y}_a - y^p_a + d^r_{jl} \), e.g. by enumeration over \( a := (i,j) \in A \) and \( l, l' \in L \).

Lifted path-inequalities of form (13.25) are separated similarly. A most violated inequality is found by minimizing \( d^r_{jl} - \sum_{a \notin \tilde{a}} y^p_a \), e.g. by enumeration over \( a := (i,j) \in A \) and \( l, l' \in L \).

Remark 13.3. From Remark 10.4 it follows that the separation algorithm for (13.23) applies also to the ECMP case.

Remark 13.4. For the lifted path-inequalities, the best \( l' \in L \setminus \{l\} \) can be found by adapting the ”trick” in (13.22) to improve the enumeration complexity to \( O(|L|n) \).

### 13.3.2 Valid Cycle Separation

We give a combinatorial algorithm to separate a most violated valid cycle inequality. Take \( l', l'' \in L \). Recall that a valid cycle is represented as \( C = F \cup B \) and is associated with the inequality

\[ \sum_{a \in F} (1 - y^p_a) + \sum_{a \in B} (1 - y^p_a) \geq 1. \]  

(13.26)
As for the algorithm in Section 7.1.4 for finding a valid cycle for two given SP-arc sets, the crucial observation is that valid cycles correspond to directed cycles in the auxiliary graph $G$, see Lemma 7.1 and Theorem 7.1. By choosing arc costs appropriately, we make shortest cycles correspond to maximally violated valid cycle inequalities. We construct $G$ so that arcs associated with $l'$ and $l''$ are used forwards and backwards, respectively, and have costs $1 - y_{l'}$ and $1 - y_{l''}$.

Algorithm 13.3.1. Given $G = (N, A)$ and $l', l'' \in L$ and $y_{l'}', y_{l''}'' \in Q^A$. Find a most violated cycle inequality (13.26).

1. Construct $G$. If $a \notin A$, let $y_{a'} = y_{a''} = 0$.
   (a) For each $a' := (i, j) \in A$, set $a'' := (j, i)$ and add arc $a'$ to $G$ with cost
   \[ c_a := \min\{1 - y_{a'}, 1 - y_{a''}\}. \quad (13.27) \]
   (b) Remove arcs with cost at least 1.

2. Find a shortest cycle in $G$.
   (a) For each $s \in N$ do the following.
      i. Find the shortest path distance $\pi_i$ from $s$ to $i \in N$ w.r.t. $c$. Denote the shortest path tree by $T_s$.
      ii. Examine all arcs $a := (i, s) \in \delta^+(s)$ entering $s$. If
         \[ \pi_i + c_a < 1 \quad \text{and} \quad a'' := (s, i) \notin T_s, \quad (13.28) \]
         then a cycle in $G$ corresponding to a violated valid cycle inequality has been found.
         (The valid cycles is formed by the shortest path from $s$ to $i$ and the arc $a := (i, s)$. To find the F/B labelings, use (13.27).)

3. Return the valid cycle with the largest violation.
   (Or, return all cycles with a pre-specified violation.)

To make sure that the cycle induced by $a := (i, s)$ is simple in Algorithm 13.3.1, we required that the reverse arc $a' := (s, i)$ is not in the shortest path tree. This is to avoid generating a "cycle" that consists of two parallel arcs.

Remark 13.5. In the shortest path calculations, all nodes that get a distance label of at least 1 can be ignored since this is an upper bound on the length of interesting cycles.

A most violated valid cycle inequality is found by considering all destination pairs.

Proposition 13.2
The separation problem for valid cycles can be solved in $\mathcal{O}(\|L\|^2 n (m + \log n))$ time.

Remark 13.6. An alternative to finding a most violated inequality is to find (sufficiently) violated inequalities with support of small cardinality. It is straightforward to adapt Algorithm 13.3.1 to this end by using a bucket approach. Let $K$ be an upper bound on the size of an acceptable valid cycle. Then, use a distance label $\pi_i^k$ for each $k = 0, 1, \ldots, K$, to denote the length of a shortest path from $s$ to $i$ with at most $k$ arcs. This worsen the time and space complexity by a factor $K \leq n$. 
13.3.3 Valid Cycle Separation in the ECMP Case

We now consider how to find a most violated valid cycle inequality in the ECMP case for \( l', l'' \in L \). For simplicity, we omit D-arcs. From Chapter 10 it follows that the valid cycle inequalities associated with \( C = F \cup B \) are of the form,

\[
y_a'^l + \sum_{a \in F} \left( 1 - y_a'^l \right) + \sum_{a \in B} \left( 1 - y_a'^l \right) \geq 1, \quad \tilde{a} \in B, \ l = l',
\]

\[
y_a'^l + \sum_{a \in F} \left( 1 - y_a'^l \right) + \sum_{a \in B} \left( 1 - y_a'^l \right) \geq 1, \quad \tilde{a} \in F, \ l = l''.
\]

(13.29)

Recall the decomposition of a valid cycle into alternating path segments in Section 7.1, see Equation (7.5) and (7.6) and the template in Figure 7.1. Re-writing (13.29) using this decomposition yields

\[
y_a'^l + \sum_{p=1}^{K} \sum_{a \in P} \left( 1 - y_a'^l \right) + \sum_{p=1}^{K} \sum_{a \in P} \left( 1 - y_a'^l \right) \geq 1,
\]

(13.30)

for appropriate combinations of \( l \in \{l', l''\} \) and \( \tilde{a} \in C \).

To derive a separation algorithm we make a fundamental observation. The left hand side in (13.30) is the sum of shortest path distances and a complicating extra term associated with some arc in some shortest path. Since \( C \) is decomposed into alternating path segments, a most violated valid cycle inequality corresponds to a cycle in an auxiliary graph created in a similar manner as in the USPR case. We show this construction when we have defined the notation for the path segments that form \( C \).

As above, let \( d_{st}^l \) be the shortest path distance from \( s \) to \( t \) w.r.t. arc costs \( 1 - y_a'^l \), see (13.21). We write

\[
p_{st}^l = d_{st}^l \quad \text{and} \quad p_{st}'' = d_{st}''.
\]

(13.31)

To handle the complicating arc \( \tilde{a} \) in (13.30), we define

\[
p_{st}^l = \min_{P \in P_{st}^l} \left\{ \sum_{a \in P} \left( 1 - y_a'^l \right) + \min_{a \in P} y_a'' \right\},
\]

\[
p_{st}'' = \min_{P \in P_{st}'} \left\{ \sum_{a \in P} \left( 1 - y_a'' \right) + \min_{a \in P} y_a' \right\}.
\]

(13.32)

In a sense, \( p_{st}^l \) and \( p_{st}'' \) are shortest path distances where the complicated arc \( \tilde{a} \) must be compensated for. Observe that the path associated with \( p_{st}'' \) is from \( t \) to \( s \).

In analogy with the USPR case, we create the auxiliary graph \( \tilde{G} \) such that valid cycles correspond to directed cycles in \( \tilde{G} \). As before, arcs associated with \( l' \) and \( l'' \) are used forwards and backwards, respectively. The difference is that arcs in \( \tilde{G} \) in the ECMP case correspond to path segments. Also, when calculating the cost of a cycle, one arc, i.e. path segment, must pay for the complicating arc \( \tilde{a} \). Hence, for the cycle cost to equal the violation of the valid cycle inequality, we choose arc costs from \( p_{st}^l, p_{st}'' \) and \( p_{st}'' \).
Algorithm 13.3.2. Given $G = (N, A)$ and $l', l'' \in L$ and $\bar{y}', \bar{y}'' \in Q^A$. Find a most violated valid cycle inequality (13.29).

1. Construct $\tilde{G}$. If $a \notin A$, let $y'_a = y''_a = 0$.

   (a) For each $(s, t) \in N \times N$, determine the quantities $p'_st, p''_st, p'_st$ in (13.31) and (13.32). Add two parallel arcs $a := (s, t)$ and $\tilde{a} := (s, t)$ to $\tilde{G}$ with costs

   \[ c_a := \min \{ p'_st, p''_st \}, \quad \tilde{c}_a := \min \{ p'_st, p''_st \}. \]  

   (We refer to the arc $\tilde{a}$ as a transitive arc.)

   (b) Remove arcs with cost at least 1.

2. Find a shortest cycle in $\tilde{G}$ that contains at least one transitive arc.

3. Return the valid cycle corresponding to a shortest cycle.

We show how to implement Step 1a and 2 below.

Remark 13.7. It is possible that a shortest cycle in Algorithm 13.3.2 induce two parallel paths. In the ECMP case, this is not an issue due to the contribution to the objective caused by the complicating arc $\tilde{a}$.

Remark 13.8. If a cycle in $\tilde{G}$ is shortest it corresponds to some most violated inequality. It is not uncommon that the cycle returned needs some polishing. Consider for instance two subpaths that are not internally node disjoint, then the associated cycle family is reducible, and can be divided into two smaller cycle families. Even though the violation is not affected, the inequalities associated with the smaller cycle families are superior to the inequality associated with the original cycle family.

Theorem 13.2
Let $C = F \cup B$, $\tilde{a} \in C$ and $l \in \{l', l''\}$ be the valid cycle, the associated arc and the destination that minimize

\[ y'_a + \sum_{a \in F} \left( 1 - y'_a \right) + \sum_{a \in B} \left( 1 - y''_a \right) \]  

over all valid cycle inequalities without a $D$-arc. Then, $C$ corresponds to a directed cycle of minimal length in $\tilde{G}$ that uses exactly one transitive arc.

To compute the quantities $p'_st, p''_st, p'_st$ in Step 1a in Algorithm 13.3.2, we propose to use a dynamic programming approach. Finding a shortest cycle in Step 2 can either be accomplished by using dynamic programming or via a modification of Dijkstra’s algorithm where two distance labels are used to record the distance with and without a transitive arc. We give the dynamic programming solutions to these problems.

We can use Floyd–Warshall’s algorithm to compute $p'_st$ and $p''_st$. This is a dynamic programming algorithm. To find the costs $\tilde{p}'_st$ and $\tilde{p}''_st$ for transitive arcs, we modify the recurrence relation used in Floyd–Warshall’s algorithm in the natural way. First consider $p'_st$. Let $\tilde{p}'_st$ be the length of a shortest transitive path from $s$ to $t$ using only nodes in $\{s, t, 1, \ldots, k\}$. The recurrence relation for $\tilde{p}'_st$ becomes as follows.
If \( k = 0 \), set
\[
p^{k}_{st} = 1 - y'_{st} + y''_{st}.
\] (13.35)

If \( k > 0 \), set
\[
p^{k}_{st} = \min \{ p^{k-1}_{st}, p^{k-1}_{sk} + p^{k-1}_{kt} \}.
\] (13.36)

The motivation for the latter recurrence is straightforward. If a transitive path becomes shorter by taking a shortcut via node \( k \), then the new transitive path consists of an ordinary shortest path and a shortest transitive path. The arc costs \( p''_{st} \) are found similarly taking directions into account, cf. (13.32).

To find a shortest transitive cycle we define \( d^{k}_{st} \) to be the length of a shortest path from \( s \) to \( t \) in \( G \) and \( d^{k}_{st} \) the length of the shortest transitive path from \( s \) to \( t \) in \( G \) using only nodes \( 1, \ldots, k \) as intermediate nodes. This yields the following recurrence relations.

If \( k = 0 \), let
\[
d^{k}_{st} = \min \{ p'_{st}, p''_{st} \} \quad \text{and} \quad d^{k}_{st} = \min \{ p'_{st}, p''_{st} \}.
\] (13.37)

If \( k > 0 \), let
\[
d^{k}_{st} = \min \{ d^{k-1}_{st}, d^{k-1}_{sk} + d^{k-1}_{kt} \}
\] (13.38)

and
\[
d^{k}_{st} = \min \{ d^{k-1}_{st}, d^{k-1}_{sk} + d^{k-1}_{kt}, d^{k-1}_{st} + d^{k-1}_{kt} \}.
\] (13.39)

The motivation for the latter recurrence is as for \( p^{k}_{st} \); A transitive path that benefits from using node \( k \) will consist of a transitive path and an ordinary shortest path.

A most violated valid cycle inequality corresponds to a shortest transitive cycle which is associated with a minimal diagonal element \( d^{n}_{ss} < 1 \). To find the actual cycle and arc, the solution can be traced in a similar manner as in Floyd–Warshall’s algorithm.

All recurrence relations above can be solved in \( O(n^3) \) time by dynamic programming. As in the USPR case, a most violated valid cycle inequality is found by considering all destination pairs.

**Proposition 13.3**
The separation problem for valid cycles can be solved in \( O(|L|^2 n^3) \) time in the ECMP case.

### 13.4 Computational Aspects of SPR Problems

A possible computational scheme for solving traffic engineering problems in IP networks is outlined in Figure 13.1. First we discuss how the pieces in Part III of this thesis relates to each other and how they are intended to be used in computations. Then, the scheme is described.

Recapitulating, we have primarily considered two approaches for solving core traffic engineering problems in IP networks. First, solving it as an ordinary mixed integer linear
Further, we have shown how to take several common aspects of optimization problems in IP networks into account, i.e. how to modify the core problem to handle more realistic real-world applications. Assembling a properly chosen subset of these components yields a concrete traffic engineering problem formulation, we refer to this as setting up the model, and the resulting model as the complete model. This is the first step in the computational scheme in Figure 13.1.

To handle the SPR aspect of traffic engineering problems in IP networks, we characterized valid inequalities based on routing conflicts in Chapter 10, and showed how to use them in the core master problem in Chapter 9 and 11. The final, and crucial, issue of how to separate such inequalities was resolved in this chapter.

Finally, several problems related to traffic engineering problems in IP networks have been considered, e.g. ECMP splitting at a single node and the acyclic ingraph problem in Chapter 12. The associated models for such problems appears as subsystems of inequalities in the complete model. Hence, facets and valid inequalities for the associated models translate to valid inequalities for the complete model. Recall that many related problems came with solution algorithms or separation algorithms, hence we can separate these valid inequalities also in the setting of the complete model (if necessary, via the equivalence of optimization and separation).

### 13.4.1 Computation Scheme Components

We briefly describe the components of the computational scheme in Figure 13.1. A complete description of a full-scale implementation of some real-world traffic engineering problems in IP networks in the USPR case can be found in Chapter 9 in [46].

**Setup model** This step was handled in the discussion above.

---

**Figure 13.1:** An illustration of a possible computational scheme for solving traffic engineering problems in IP networks.
Primal heuristics The use of primal heuristics to find good feasible solutions is of major importance in optimization. For traffic engineering problems in IP networks this is even more true. Indeed, these problems can be immensely hard and just finding a feasible solution is challenging for state-of-the-art MILP solvers. Further, there exists a variety of efficient metaheuristics for these problems. See e.g. the references in Section 2.5.1.

Preprocessing Another topic of major importance optimization is preprocessing, see e.g. [1, 203] for general treatments of this subject. For some traffic engineering problems in IP networks it is possible to remove well above 50% of all variables by simple and efficient preprocessing rules [150].

Branch / Node selection We discuss the selection of the variable to branch on. The general rule of thumb is to branch on variables that represent major decisions first. In traffic engineering problems, an example is to first branch on variables associated with big demands. Another example is if an extended formulation is used in the ECMP case, then deciding whether to split or not represent a major decision.

Pricing Variable pricing was discussed in Chapter 11 and 12. We also mentioned how to easily use over-generation in Chapter 11.

Cutting The LP formulation at intermediate nodes in an enumeration tree can be strengthened by adding cuts, e.g. general purpose cuts such as \( \{0, \frac{1}{2}\}\)-cuts and Gomory mixed integer cuts. This has proven to be of utmost importance for solving MILPs. Preferably, several cuts should be added in rounds. The issue of selecting which cuts is referred to as cut selection, and is very important. For instance, in the traffic engineering context, it makes sense to select cuts that involve variables that correspond to destinations associated with big demands. Also recall that valid cycles from different strongly connected components yield coordinated cuts.

Valid cycle separation We showed how to separate valid cycles in Section 13.3 for a given pair of destinations. Heuristically, the biggest impact is obtained by selecting destination pairs associated with big demands. Since valid cycles can be separated efficiently, this step can be used in relatively many nodes.

General separation The problem of separating a general routing conflict was solved in Section 13.1 and 13.2 for the USPR case. It is quite costly to solve the LP for separating a general routing conflict heuristically, and it is very costly to solve the separation problem exactly. Therefore, this step should be used in relatively few nodes.

Greedy primal heuristics Several components in the solution scheme in Figure 13.1 produces as a byproduct a set of link weights, e.g. explicitly if an inverse shortest path problem is solved, and implicitly via the dual variables for the link capacity constraints, see e.g. [110]. These link weights can then be improved e.g. using a metaheuristic and a greedily chosen neighborhood that facilitates improvement.

This concludes our discussion of computational aspects, and the thesis. Some directions for future research are given in the next chapter.
The primary concern of this thesis has been the theoretical aspect of shortest path routing in traffic engineering problems in IP networks. These problems lead to very difficult optimization problems. Our approach is based on analyzing an inverse shortest path routing problem that is a key element in exact methods for such problems. The analysis give rise to an improved theoretical understanding of the shortest path routing mechanism that translates to improved approaches to the modelling and solution of problems related to traffic engineering in IP networks. A summary of all chapters can be found in Section 1.1, and a more detailed account for our contributions is given in Section 1.2.

Since we have focused on theory, the practical aspect has suffered — this is the main weakness of the thesis. The natural continuation of our work is to compare full-scale implementations of the solution methods that we have considered. Our Dantzig–Wolfe reformulation in Chapter 12 for the traffic engineering problem where splitting is allowed is particularly promising since it avoids the explicit inclusion of splitting constraints.

We strongly believe that one of the most important directions for future research is to derive valid inequalities for based on the combination of shortest path routing and capacities, e.g. cutset inequalities. Such inequalities have shown to be very efficient for many network design problems.

On the theoretical side, a continued investigation of shortest path routing polytopes is an interesting direction for future research. In particular, characterizing facets and developing separation algorithms.
Bibliography


