Linköping studies in science and technology. Theses.
No. 1448

Inverse Shortest Path Routing Problems in the Design of IP Networks

Mikael Call

INSTITUTE OF TECHNOLOGY
LINKÖPINGS UNIVERSITET

Department of Mathematics
Linköping University, SE–581 83 Linköping, Sweden

Linköping 2010

Linköping studies in science and technology. Theses.
No. 1448

Inverse Shortest Path Routing Problems in the Design of IP Networks –

Mikael Call

`mikael.call@liu.se`
`www.mai.liu.se`
Division of Mathematical Statistics
Department of Mathematics
Linköping University
SE–581 83 Linköping
Sweden

ISBN 978-91-7393-329-2 ISSN 0280-7971

Printed by LiU-Tryck, Linköping, Sweden 2010

In memory of my father.

Abstract

This thesis is concerned with problems related to shortest path routing (SPR) in Internet protocol (IP) networks. In IP routing, all data traffic is routed in accordance with an SPR protocol, e.g. OSPF. That is, the routing paths are shortest paths w.r.t. some artificial metric. This implies that the majority of the Internet traffic is directed by SPR. Since the Internet is steadily growing, efficient utilization of its resources is of major importance. In the operational planning phase the objective is to utilize the available resources as efficiently as possible without changing the actual design. That is, only by re-configuration of the routing. This is referred to as traffic engineering (TE). In this thesis, TE in IP networks and related problems are approached by integer linear programming.

Most TE problems are closely related to multicommodity routing problems and they are regularly solved by integer programming techniques. However, TE in IP networks has not been studied as much, and is in fact a lot harder than ordinary TE problems without IP routing since the complicating shortest path aspect has to be taken into account. In a TE problem in an IP network the routing is performed in accordance with an SPR protocol that depends on a metric, the so called set of administrative weights. The major difference between ordinary TE problems and TE in IP networks is that all routing paths must be simultaneously realizable as shortest paths w.r.t. this metric. This restriction implies that the set of feasible routing patterns is significantly reduced and that the only means available to adjust and control the routing is indirectly, via the administrative weights.

A constraint generation method for solving TE problems in IP networks is outlined in this thesis. Given an "original" TE problem, the idea is to iteratively generate and augment valid inequalities that handle the SPR aspect of IP networks. These valid inequalities are derived by analyzing the inverse SPR problem. The inverse SPR problem is to decide if a set of tentative routing patterns is simultaneously realizable as shortest paths w.r.t. some metric. When this is not the case, an SPR conflict exists which must be prohibited by a valid inequality that is then augmented to the original TE problem. To derive strong valid inequalities that prohibit SPR conflicts, a thorough analysis of the inverse SPR problem is first performed. In the end, this allows us to draw conclusions for the design problem, which was the initial primary concern.

Populärvetenskaplig sammanfattning

Denna avhandling handlar om problem relaterade till kortaste-väg-ruttning i IP-nätverk. I ett IP-nätverk dirigeras all datatrafik i enlighet med ett kortaste-väg-protokoll (t.ex. OPSF) d.v.s. alla vägar måste vara kortaste vägar m.a.p. någon (artificiell) metrik. Detta leder till att majoriteten av all trafik på Internet styrs via kortaste vägar. Eftersom Internet är stadigt växande är det viktigt att utnyttja dess resurser effektivt. I verksamhetsplaneringsfasen är målet att utnyttja de befintliga resurserna så bra som möjligt utan att ändra den faktiska, underliggande designen. Det innebär att det enda tillgängliga medlet är att konfigurera om ruttningen. Detta är kallas trafikplanering. I denna avhandling behandlas problem relaterade till trafikplanering för IP-nätverk med hjälp av linjär heltalsprogrammering.

De flesta trafikplaneringsproblem är nära besläktade med multicommodity-problem och löses regelbundet med heltalsprogrammeringstekniker. Trafikplanering i IP-nätverk har dock inte studerats lika mycket och är faktiskt mycket svårare att lösa än vanliga trafikplaneringsproblem utan IP-ruttning. Detta eftersom den komplicerade aspekten med kortaste vägar måste beaktas. I ett trafikplaneringsproblem i ett IP-nätverk måste ruttningen ske i enlighet med ett kortaste-väg-protokoll som är beroende av en uppsättning så kallade administrativa vikter. Den stora skillnaden mellan vanliga trafikplaneringsproblem och trafikplanering i IP-nätverk är att alla vägar måste vara realiserbara som kortaste vägar m.a.p. dessa vikter samtidigt. Denna begränsning innebär att antalet realiserbara ruttningsplaner minskar kraftigt. Dessutom är de administrativa vikterna den enda möjligheten att (indirekt) påverka trafikdirigeringen.

En metod baserad på bivillkorsgenerering beskrivs i denna avhandling för att lösa trafikplaneringsproblem i IP-nätverk. Utgående från ett "vanligt" trafikplaneringsproblem är tanken att iterativt generera giltiga olikheter som hanterar ruttnings-aspekten i ett IP-nätverk. Dessa olikheter härleds genom att analysera det omvända ruttningsproblemet. Detta omvända ruttningsproblem består i att avgöra om en preliminär ruttningsplan kan realiseras som kortaste vägar m.a.p. någon metrik. När detta inte är fallet finns en konflikt som måste förbjudas. Detta sker genom att en giltig olikhet läggs till det ursprungliga trafikplaneringsproblemet. För att få fram starka olikheter som förbjuder konflikter analyseras det omvända ruttningsproblemet. I slutändan möjliggör detta att slutsatser för designproblemet kan dras.

Acknowledgments

I am very grateful to professor Kaj Holmberg for all the support and encouragement he has given me. I am very happy for the opportunity to conduct research studies in combinatorial optimization, in particular network optimization and shortest path routing problems. I enjoy being able to freely investigate research ideas of my own. Kajs feedback on incomplete drafts on many subjects have significantly improved the quality of the final presentations.

My PhD colleagues, Åsa Holm and Nils-Hassan Quttineh, deserve my thanks for their reading of parts of this thesis. Their comments have improved my presentation of the material. My other Opt and PhD colleagues contribute to making MAI a nice working place.

Finally, I thank my family and friends for all the love and support you have given me, especially my wife Anna.

Linköping, August 19, 2010
Mikael Call

Contents

1 Introduction and Overview **1**

1.1 Background 1

1.2 Outline 3

1.3 Contributions 4

2 Traffic Engineering in IP Networks **7**

2.1 Technical Background 7

2.2 Traffic Engineering 9

2.2.1 A Conceptual Mathematical Model 10

2.2.2 A Heuristic Approach: Search in the Weight Space 11

2.2.3 An Exact Approach: MILP Formulations of STEP 14

3 Inverse Shortest Path Routing **19**

3.1 The Shortest Path Problem 19

3.2 Inverse Shortest Path Problems 21

3.2.1 A Polynomial Formulation of ISP 22

3.2.2 Some Comments on SP-Graphs 23

3.3 Inverse Shortest Path Routing Problems 24

3.4 Mathematical Formulations of ISPR Problems 26

3.4.1 Internal Consistency 28

3.4.2 A Polynomial Formulation of ISPR 31

3.4.3 A Formulation of ISPR with Non-Spanning Ingraphs 36

3.4.4 A Formulation of ISPR with Spanning SP-graphs 38

3.5 Algorithms to Transform Path Sets to SP-graphs 38

4 Inverse Partial Shortest Path Routing 41
4.1 Inverse Partial Shortest Path Routing Problems 41
4.2 Mathematical Formulations of IPSPR Problems 44
4.2.1 A Model for Partial Compatibility 44
4.2.2 Models for Realizability 45
4.3 Differences Between the IPSPR Models 48
4.4 Complexity of IPSPR Problems 56
4.4.1 Discussion 67

5 Infeasible Routing Patterns 69
5.1 Problem Formulation 69
5.1.1 The Partial Compatibility Model 70
5.1.2 The Partial Realizability Model 71
5.2 Classes of Infeasible and Forcing Structures 75
5.2.1 The General Structure 76
5.2.2 The Binary, Unitary and Simplicial Structures 80
5.2.3 A Canonical Circuit Decomposition 82
5.3 Extreme Rays and Generators 83
5.3.1 Representation of Polyhedral Cones 84
5.3.2 Extreme Rays and Generators of the Closure of Θ 85
5.3.3 Irreducible Solutions of the Closure of Θ 88
5.4 Structures Involving at Most Two SP-Graphs 89
5.4.1 Saturating Solutions with Two SP-Graphs 93
5.4.2 Non-Saturating Solutions with Two SP-Graphs 96
5.4.3 Algorithms to Find Generalized Valid Cycles 98
5.5 On the Relation Between Classes of Structures 103

6 Simplicial Cycle Families 105
6.1 A Characterization of the Simplicial Structure 105
6.1.1 Graph Embeddings Yield Simplicial Cycle Families 107
6.1.2 Simplicial Cycle Families Yield Graph Embeddings 114
6.2 The Simplicial Dependency Graph 116
6.2.1 A Simplicial Cycle Family Induces a Dependency Graph 116
6.2.2 A Dependency Graph Induces Simplicial Cycle Family 120
6.2.3 The Equivalence Class Induced by a Dependency Graph 123
6.2.4 Characterization of Irreducible Generators 123
6.3 The Hierarchy of Infeasible Structures 127

7 Cycle Bases: Models and Methods 135
7.1 Modelling Circulation Problems with Cycle Bases 136
7.1.1 Oriented Circuits, Circulations and Cycle Bases 136
7.1.2 Fundamental Cycle Bases 137
7.1.3 Modelling Circulations with Cycle Bases 138
7.1.4 The Minimum Cost Circulating Flow Problem 139
7.1.5 Multicommodity Minimum Cost Circulating Flow 140
7.2 IPSPR Formulations Based on Cycle Bases 141

7.2.1 Modelling Partial Realizability with Cycle Bases 141
7.2.2 Modelling Partial Compatibility with Cycle Bases 145
7.2.3 The Cycle Basis Model in Matrix Notation 147
7.3 Properties of the Cycle Basis Formulation 148
7.3.1 Constraint Redundancy 148
7.4 The Farkas System of the Cycle Basis Model 153
7.5 Numerical Examples for the Cycle Basis Model 156
7.6 Cycle Basis Computations 161
7.6.1 Computing the Cycle Matrix 162
7.6.2 Computing Cycle Matrix Vector Multiplications 163

8 Shortest Path Routing Design 169
8.1 Modelling SPRD Problems 169
8.1.1 SPRD Problems are Hard 172
8.1.2 A Branch and Cut Approach to SPRD Problems 173
8.2 The Set of Feasible Routing Patterns 174
8.2.1 Vectors that Induce Partially Realizable SP-graphs 175
8.2.2 Valid Inequalities From Routing Conflicts 176
8.2.3 Independence Systems and Transitive Packings 180
8.2.4 The Conflict Hypergraph for Routing Patterns 182
8.2.5 The Acyclic Ingraph Polytope 191
8.3 Separation of Some Combinatorial SPR Cuts 193
8.3.1 Separation of SPR Cuts in General 194
8.3.2 Directed Cycle Cuts 194
8.3.3 Cuts from Subpath Inconsistence Conflicts 196
8.3.4 Cuts from Valid Cycles 198

9 Conclusion 205
9.1 Summary 205
9.2 Further Work 207

Bibliography 209

1 Introduction and Overview

THIS thesis is concerned with problems related to shortest path routing (SPR) in Internet protocol (IP) networks. All activities on the Internet require that data is sent from a source to a destination. The determination of the path to use from the source to the desitnation is called routing. In IP routing, it is very common that the data traffic is routed in accordance with an SPR protocol. That is, the routing paths are shortest paths w.r.t. some artificial metric. This implies that the majority of the Internet traffic is directed by SPR.

The Internet is steadily growing and efficient utilization of its resources is of major importance for the quality of service (QoS) provided to customers. Therefore, it is well motivated to study SPR in IP networks and especially how to utilize resources when the traffic is routed by an SPR protocol. In this thesis, these issues and related problems are approached by mathematical programming.

1.1 Background

Mathematical programming is used to solve several planning problems in telecommunications. Many 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 often that the objective is to minimize an estimated total design and routing cost. Several matematical models have been developed for these strategic problems, and some of them also take other issues into account, e.g. survivability. These models are often based on (capacitated) facility location and multicommodity design and routing models that have been studied much in the litterature.

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. That is, 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 (TE). In this thesis, the focus will be limited to TE in IP networks.

There exist several matematical models for TE problems; they are often closely related to (capacitated) multicommodity routing problems. However, TE in IP networks has not been studied as much, and is in fact a lot harder than ordinary TE problems without IP routing since the complicating shortest path aspect have to be taken into account.

In a TE problem in an IP network the routing is performed in accordance with an SPR protocol that depends on a metric, the so called set of administrative weights. The major difference between ordinary TE problems and TE in IP networks is that *all routing paths must be simultaneously realizable as shortest paths w.r.t. this metric.*

Example 1.1

Consider some TE problem where the traffic is not routed with SPR. Suppose that the graph has four nodes and that there are two origin-destination (OD) pairs. The first one associated with a one unit demand from node 1 to node 4 and the second with a one unit demand from node 2 to node 4. Assume that the solution depicted in Figure 1.1 is an optimal solution to this problem. That is, the solution described by the flow variables

$$x^1_{12} = 1,\ x^1_{24} = 1,\ x^2_{23} = 1,\ x^2_{34} = 1, \tag{1.1}$$

where x^k_{ij} is the flow on arc (i,j) of OD pair k (in this instance, this also happens to correspond to the OD pair with origin k).

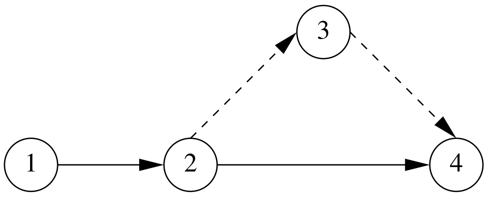


Figure 1.1: *A flow assignment where the flow from node 1 to node 4 is indicated by solid arrows and the flow from 2 to 4 by dashed arrows. This assignment is not realizable in an SPR protocol since the two different subpaths from 2 to 4 can not simultaneously be (unique) shortest paths.*

This routing pattern contains two paths that are not simultaneously realizable as shortest paths. All weights that yield $1-2-4$ and $2-3-4$ as shortest paths also yield the subpath $2-4$ as a shortest path to 4 which is infeasible since it is not consistent with the routing pattern described by (1.1). The following modification of (1.1) makes the flow solution realizable,

$$\begin{array}{llll} x_{12}^1 = 1, & x_{23}^1 = 0.5, & x_{24}^1 = 0.5, & x_{34}^1 = 0.5 \\ x_{23}^2 = 0.5, & x_{34}^2 = 0.5, & x_{24}^2 = 0.5. & \end{array} \tag{1.2}$$

Note that in the routing pattern induced by (1.2) the flow of OD pair 1 is divided into two 0.5 unit flows on the paths $1-2-3-4$ and $1-2-4$ and the flow of OD pair 2 into two 0.5 unit flows on $2-3-4$ and $2-4$. All these paths are simultaneously realizable as indicated in the begining of the example. The following set of weights suffice.

$$w_{12} = 1, w_{23} = 1, w_{24} = 2, w_{34} = 1. \tag{1.3}$$

The restriction that all routing paths must be simultaneously realizable as shortest paths implies that the set of feasible routing patterns is significantly reduced. More importantly, it also implies that the only means available to adjust and control the routing is indirectly, via the administrative weights. These issues are adressed further when the complicating SPR constraints are analyzed in this thesis.

1.2 Outline

Besides this introduction, this thesis essentially contains two parts. The first part consists of Chapters 2 to 7 which treat inverse shortest path routing (ISPR) problems. The second part consists of Chapter 8 which deals with SPR design problems. A very brief outline of the chapters is as follows.

Chapter 1 This introduction and overview.

Chapter 2 The technical background is covered and the structure of the Internet is described. It is also explained how SPR protocols usually works. An outline of the two major solution approaches for TE problems in IP networks is given.

Chapter 3 ISPR problems are introduced. Some previously known models are presented and some new models are derived.

Chapter 4 It is shown that the commonly used formulation of the inverse shortest path routing problem for partial ingraphs is incomplete. A complete model is proposed and it is shown that the problem of determining if a family of partial ingraphs is realizable in an SPR protocol is NP-complete. An improved model for the inverse shortest path routing problem with partial ingraphs is also derived. This new model yields stronger necessary conditions for realizability than the commonly used old model, but it is still not complete.

Chapter 5 A characterization of (potentially) infeasible routing patterns is obtained by analyzing the Farkas systems of the models from Chapter 4. This characterization is the foundation of the valid inequalities that prohibit parts of routing patterns that are not realizable. In particular, the solutions that involve at most two SP-graphs are considered an efficient algorithm is developed.

Chapter 6 A description of the class of simplicial solution is given. A concept similar to graph duality is used to characterize simplicial extreme and irreducible solutions. The relation between the infeasible routing patterns from Chapter 5 is also analyzed further.

Chapter 7 The multicommodity circulation structure of the Farkas systems of the models from Chapter 4 is exploited to derive novel ISPR models based on fundamental cycle bases.

Chapter 8 Finally, the class of SPR design problems is considered. A mixed integer linear programming formulation without weight variables is presented. The characterization from Chapter 5 is used to derive valid inequalities that prohibit infeasible routing patterns. Separation of the important class of valid inequalities based on two destinations is considered in detail. Efficient separation algorithms are given for some settings.

1.3 Contributions

The main contributions of this thesis are as follows.

1. A major contribution is to just adress the issues of realizability. That is, to determine if a family of partial ingraphs is realizable in an SPR protocol. A related problem, denoted by partial compatibility, is to determine if there is a metric and a set of node potentials such that all specified shortest path arcs are tight and all specified non-shortest path arcs are not tight. Earlier, only this latter problem was considered. However, unless the node potentials are tight (that is, the tight arcs induce a spanning arborescence), a solution can not be used to verify realizability. Once this distinction between realizability and partial compatibility is made; we prove that the realizability problem is NP-complete. This is a significant theoretical result.

2. An improvement of the common partial compatibility model is derived by including some valid inequalities for realizability. This yields a partial realizability model that is superior to the ordinary partial compatibility model. Using partial realizability it is possible to detect more SPR infeasibility earlier and also derive more and stronger SPR valid inequalities. By analyzing the mulitcommodity structured Farkas system of the partial realizability model a combinatorial characterization of a very large class of SPR conflicts is derived. This characterization yields an explanation for a large class of combinatorial cuts that prohibit SPR conflicts. Some of these cuts are also stronger than the corresponding partial compatibility cuts since they have in a sense automatically been lifted and projected.

3. Some empirical evidence suggest that a very important subclass of combinatorial SPR cuts are based on conflicts that involve exactly two partial ingraphs. The partial realizability based cuts with two partial ingraphs subsumes and explains all combinatorial cuts from the litterature based on conflicts with two partial ingraphs. We show how to efficiently separate a fractional solution from a most violated cut and also how to efficiently find a violated cut of minimal support for this important subclass of cuts.

4. Finally, a novel modelling approach for the partial realizability problem is proposed based on fundamental cycle bases. This yields a more compact model without flow conservation constraints that can be solved more efficiently than the ordinary model. Some important theoretical insights about the models can also be derived from the cycle basis structure.

2

Traffic Engineering in IP Networks

THIS thesis is primarily concerned with TE in IP networks. To formally describe this class of problems it is necessary to give a brief presentation of the technical background; we describe the structure of the internet and how SPR usually works.

We also consider some previous work on this class of problem presented in the litterature, especially we describe the two main solution approaches that have been used, heuristics and integer programming. The common heuristic approach is to simply search in the (administrative) weight space and evaluate the resulting solutions. All exact solution approaches that we have encountered in the litterature involve mixed integer linear programming (MILP). Recent MILP formulations do not include the weights as variables since these tend to yield very weak LP-relaxations.

2.1 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, among other things, responsible of the routing of the traffic within the domain. That is, the operator must determine 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. Network operators often relies on routing protocols, which is a specification of how the traffic is routed in the network, to perform these decisions. The single most important task for the operator is to select a routing protocol and a set of routing parameters to provide an acceptable level of the QoS.

Within an AS, the routing is conducted by routers via static or dynamic routing tables. Static routing implies that paths may be configured manually, which may be 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. Most of the IGPs use SPR and send traffic along the shortest paths from the origins to the destinations, w.r.t. an artificial metric. The link weights are called administrative weights and are part of the routing protocol parameters. Actually, they 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 (IS-IS), it is required that the link weights are integral and in the interval 1 to 65536 (15). The shortest paths are easily determined given the weights, e.g. by Dijkstras algorithm. The routing paths are implicitly stored by a set of forwarding tables, one for each router. This is obviously much more efficient than calculating the shortest path for each package. In practice, at a given router, the next router on the path for a package is determined by a lookup in the forwarding table depending on the destination of the package.

A standard of how to deal with the case were there are several shortest paths is not specified in the current OSPF [74], nor in the IS-IS [33], specification. Because of this and other reasons several authors that consider TE in IP networks restrict the number of shortest paths between an origin and a destination to 1. This version of the problem is called the single path case, or unique shortest path routing. Lately, some authors have also considered multiple shortest paths. The common assumption used in the mathematical modelling of these protocols is then the folloing equal cost multi-path (ECMP) splitting rule. If, at a node, there are several shortest paths to a destination, then the ingoing traffic to this node is divided evenly among all the outgoing arcs that are on a shortest path to the destination. Note that this is in general not the same as an even distribution of the traffic on *all* shortest paths, cf. Example 2.1 below.

Example 2.1

The ECMP principle is demonstrated for the set of administrative weights in the left of Figure 2.1. The induced flow from node O to node D is shown in the right of Figure 2.1. There are 3 shortest paths, and two of them carry 0.25 units of flow and the last carries 0.5 units of flow.

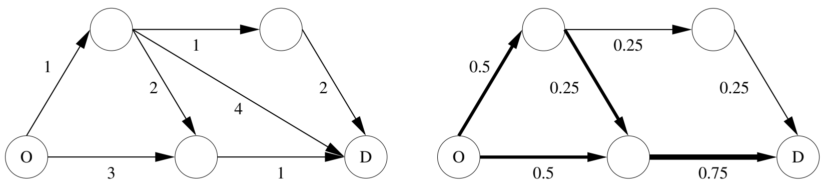


Figure 2.1: *The weights and induced flow according to the ECMP principle.*

We have pointed out above that most of the traffic on the Internet is routed by SPR protocols and that the activities on the Internet requires more and more resources, that is, bandwidth. The resources have to be utilized efficiently to provide an acceptable QoS level to customers. The routing paths alone determine how much bandwidth that is used on all link and the only way for the network operator to affect these routing paths is implicit, via the administrative weights. Given this information it is clear that TE in IP networks is a very important area of research. Let us now consider this class of problems.

2.2 Traffic Engineering

During the last decade it has become more common that IP network operators determine the administrative weights by mathematical programming methods. But it is also still common that some default weight settings are used. The simplest idea is to use the hop count, that is, just to set each link weight to 1. An apparently more sophisticated choice of link weights, recommended in [36], is to use a weight that is inversly proportional to the capacity of the link. This yields lower weights and therefore more traffic on high capacity links. However, it turns out that both these suggested settings often perform poor in terms of minimizing link load, cf. [45, 50].

Suppose that a network administrator does not use the default settings above but somehow assigns administrative weights. It is easy to determine the induced routing and then use some simulation procedure to measure the network performance in different senses. Unfortunatly, it is not clear how to adjust a weight setting if the shortest paths or performance measure are not satisfactory. Trial and error will yield good enough results if the administrator is lucky or has enough time. Given a limited amount of time, sufficiently good results can not be guaranteed. A problem in this process is clearly that the control of the flow distribution is only indirect which makes it hard to foresee or estimate all effects of the adjustments without potentially expensive calculations.

It was early realized that instead of configuring administrative weights manually, they can be adjusted algorithmically by a computer. Given a performance measure, it is possible to evalutate the measure for a collection of weights and simply select the best setting. In theory, all weight settings can be evaluated and the best one selected. This is not possible in practice, since the number of settings will be too large. In practice, and from an engineers 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.

Suppose that we are not satisfied with this heuristic approach but actually require an optimal solution. Explicit evaluation of all settings is often not possible in a reasonable amount of time, so we will use mathematical programming and implicit enumeration schemes to solve this problem. Both the heuristic and mathematical programming approach are outlined below.

2.2.1 A Conceptual Mathematical Model

Several network design and TE problems come with additional constraints on the routing. This implies that some routing plans are not realizable which may reduce the QoS. In the simplest case, it is assumed that any path can be use and that any amount of flow may be sent along the path. In this problem setting all traffic may be routed between origins and destinations as a multicommodity flow without restrictions. Another common (more realistic) restriction is that any path can be used, but all flow along must be sent along a single path. Several variants of these two problems have been studied extensively in the litterature and good matematical formulations and solution methods exist.

TE in IP networks is much more complicated since the shortest path requirement yields restrictions both on the paths that can be used and the amount of flow that may be sent along the paths. It is very unlikely that a given multicommodity flow solution is realizable in an SPR protocol (e.g. a solution that is optimal in a network design and TE problem with any multicommodity flow). Therefore tailored mathematical models that take SPR into account have to be developed. A major problem with such models is that the control of the flow distribution is indirect via the routing protocol parameters. In fact, it is not trivial to model the set of feasible routing patterns explicitly in a MILP such that the formulation is reasonably strong.

We now formally introduce the TE problem for IP networks. The problem is usually referred to as the shortest path TE problem (STEP) in the litterature were an optimal routing that is realizable in an SPR protocol is sought in an existing network.

Let $G = (N, E)$ be a graph that represents the network. The set of nodes, N, corresponds to routing devices and the set of edges, E, corresponds to links between routing devices. If the direction of an edge matters, it is called an arc and the set of arcs is denoted by A. The standard notations $n = |N|$ and $m = |E|$ or $m = |A|$ are used for the number of nodes and edges or arcs, respectively. The capacity of link (i, j) is denoted by u_{ij}. To model traffic flow and demand, a set of commodities, C, is introduced, one for each origin-destination pair (OD-pair). For each commodity, $k \in C$, there is an origin, o_k, a destination, d_k, and a traffic demand, h_k, between the origin and the destination.

The conceptual mathematical model of STEP given below only use the administrative weights as *decision* variables. There are two variables associated with each edge, w_{ij} and w_{ji}, one for each direction. The *auxilliary* variable y^k_{ij} is 1 if the arc (i, j) is on a shortest path from node o_k to node d_k and 0 otherwise. Finally, the amount of flow of commodity k on arc (i, j) is x^k_{ij}.

Clearly, the *auxilliary* flow variables x are implicitly determined by the shortest path variables y, which in turn are implicitly determined by the weights, w. Suppose that there are path and flow functions, P and F, that describe these relations as $y = P(w)$ and $x = F(y)$. If the performance measure is described by the function $f(x, y)$ the following conceptual problem formulation is obtained.

$$\begin{array}{lll} \max & f(x,y) & \\ s.t. & y & = & P(w) \\ & x & = & F(y) \\ & w \in W,\ x \in X,\ y \in Y. \end{array} \tag{STEP-C}$$

Here, $W \subseteq \mathbb{Z}_+^m$, $X \subseteq \mathbb{R}_+^{|C|\times m}$ and $Y \subseteq \mathbb{B}^{|C|\times m}$ describe the feasible regions for the corresponding variables. The set X should guarantee that the traffic demand is satisfied for all OD-pairs and includes constraints that ensure that the capacity is not violated (unless this is implicitly handled by the objective). Additional constraints on paths are handled by Y, e.g. there may be a hop limit. The collection of feasible weight settings varies between applications; e.g. in OSPF and IS-IS the weights must be integral and between 1 and 65535, and 1 and 63, respectively, so $W \subseteq (\mathbb{Z} \cap [1, 65535])^m$ for OSPF and $W \subseteq (\mathbb{Z} \cap [1, 63])^m$ for IS-IS applications.

Note that the general model (STEP-C) can include both design and routing depending on how the objective function is specified. The set of shortest path arcs can easily be constrained to allow ECMP routing or single path routing.

From model (STEP-C) it is seen that it is conceptually easy to describe the STEP problem by an implicit mathematical model. The provided model is very flexible since we may elaborate on the sets X and Y. It is however not useful in practice unless the functions P and F can be modelled explicitly and the model fit into some well behaved mathematical framework, e.g. as a MILP. It turns out that it is far from trivial to give good explicit models for the STEP problem. But, since the implicit functions are easy to evaluate, the conceptual formulation indicates that it may be a good idea to tackle STEP problems by heuristics. This approach is outlined in the next subsection, then an exact MILP framework is presented in the following subsection.

2.2.2 A Heuristic Approach: Search in the Weight Space

In general, metaheuristics have been used succesfully on many classes of problems and is certainly a good candidate to approximately solve STEP. Approaching STEP by metaheuristics is further motivated by the fact that it seems to be rather hard to explicitly model, but very easy to evaluate, the functions P and F above. Therefore most early and current approaches to solve STEP are by metaheuristics that search in the weight space.

Several metaheuristics have been succesfully applied on STEP ever since its first usage in [48, 50]. The following papers describe some approaches: genetic algorithms are considered in [45, 29] and local search in [18, 81, 97]. We have not studied this part of the litterature extensively. The reader is referred to the recent surveys [14] and [3] to get more accurate information about the succes of the metaheuristics approach to solve STEP.

Let us now outline a general metaheuristic scheme to solve STEP. Suppose an initial set of weights is given (e.g. all weights equal to 1). Then, there are essentially three elements of an algorithm that is based on a metaheuristic search in the weight space:

1. Determine the shortest paths and the flow induced by the weights.
2. Determine the objective value from the flow.
3. Update the set of weights.

These steps are repeated until some stopping criteria is fulfilled, e.g. reaching a prespecified iteration count or time limit. A very brief description of these steps are given below. A more thorough treatment of this method can be found in the survey [14].

Determining Shortest Paths and the Induced Flow

It is straightforward to determine the flow of all commodities given a set of administrative weights. A simple algorithm to calculate the resulting flow is outlined here.

Algorithm 2.2.1.
Given a set of commodities, C, and a metric, w, determine the induced arc flow x.

Notation: the commodity $k \in C$ represents the OD-pair (o_k, d_k) with traffic demand h_k, the flow on arc (i, j) is x_{ij}^k for commodity k, the set of destinations is L, the set of all commodities with destination $l \in L$ is C_l and the inflow of commodity k to node i is $x_k^+(i)$.

1. For each destination $l \in L$ do the following
   - Determine the ingraph G_l that is the union of all shortest path arborescences *to* l by storing all predecessor indices in Dijkstras algorithm.
   - Find a topological ordering of the nodes in G_l.
2. For each destination $l \in L$ do the following
   - For each commodity $k \in C_l$, set the inflow to the origin of the commodity to the traffic demand,

$$x_k^+(o_k) := h_k.$$

   - Process the nodes in the topological order of G_l. For each node i and each commodity $k \in C_l$
     – distribute the inflow evenly on all outgoing arcs in G_l

$$x_{ij}^k := x_k^+(i)/|\delta^-(i)|, \quad j \in \delta^-(i),\ (i,j) \in G_l,$$

     – increase the inflows accordingly

$$x_k^+(j) := x_k^+(j) + x_{ij}^k, \quad j \in \delta^-(i),\ (i,j) \in G_l.$$

■

The calculation of the ingraphs requires $\mathcal{O}(nm)$ time since the modified Dijkstras algorithm requires $\mathcal{O}(m)$ time for reasonably dense graphs. In the flow distribution step, each arc in each ingraph is considered once, hence, this step also requires $\mathcal{O}(mn)$ time. Thus, the overall time complexity of the algorithm is $\mathcal{O}(mn)$.

In practice, the computations above can be carried out implicitly via the reduced costs, but we give the description with ingraphs since we will use them later on. It is important to notice that the above implementation may be inefficient in practice. One should definitely consider using a dynamic version of the shortest path method used to speed up the search algorithm, cf. [30] and [50].

A completely different, and seemingly less known, approach of determining the flow induced by a metric is by solving an optimization problem, e.g. a linear program is given in [42]. Other formulations may be of interest to obtain some desired property. The major drawback with solving an optimization problem is that it is very likely to be more time consuming. The advantage is that it can yield useful additional information, e.g. sensitivity analysis may be used as guidance when the weights are updated. It is an interesting idea to base heuristic schemes on the above. We have not encountered such methods for STEP in the litterature, nor tried it ourselfs.

Determining the Objective Value From the Flow

There are several classes of objective functions that can be used for STEP, cf. [56, 62] for a treatment of this subject. A few commonly used objective functions are: minimizing the maximal link load, minimizing the sum of convex and increasing functions of the amount of traffic on each link or an approximation of the latter via piecewise linear functions.

Evaluating most (including all of the above mentioned) objectives often involves a single, straightforward calculation. However, sometimes the objective value has to be determined by computationally expensive simulations of some performance measure based on the traffic flow. Such objective functions may be costly to evaluate which should be taken into account when the strategy for updating the weights is developed. The situation with complicated objective functions that are hard (or impossible) to describe mathematically illuminates an important strength of the metaheuristic approach: it suffice to be able to determine the objective value.

Updating the Administrative Weights

There are numerous of strategies for updating the link weights. Basically, any search method can be used, we recommend the book [31] for a presentation of several old and new methods.

An important aspect when choosing a search method is to obtain a good balance between intensification and diversification of the search. Improvement based search methods, e.g. local and tabu search, mainly focus the search in promissing areas of the solution space by trying to achieve objective improvement in most iterations. Exploratory search methods, e.g. simulated annealing and genetic algorithms, especially initially, rely on randomness to breifly explore as much as possible of the solution space and then intensify the search. This implies that improvement based methods often achieve a steep improvement of the objective value and find a good solution quickly, but have a tendency to get stuck in local optimas more often. The exploratory methods have less chance at finding good solutions initially but are compensated by their reduced risk of getting stuck at local optimas. This implies that they may actually have a higher probability of ending up at a better solution, given enough time.

It is in fact possible to prove that simulated annealing will find an optimal solution to a problem with probability 1 given enough iterations if the neighbourhood is properly designed. The idea behind this is to interpret simulated annealing as a time-independent irreducible Markov chain. This implies that in the unique stationary state distribution there is a possible probability of obtaining any given solution; in particular, an optimal one. The actual results can be found in [31] (outline) and [1] (detailed).

Combining objective improvement and exploration is a key ingredient of a succesful search method. It is sometimes a good idea to try to steer the search by clever modifications of solutions designed to improve the objective value. This must however be done with care since it could inhibit the positive exploratory effects caused by randomness. An example where it seems natural to use an improvement based search method is for instance when the maximal link load should be minimized. Here a reasonable idea is to reduce the weight on the edge with the most traffic to reduce traffic on this edge and hopefully distribute the traffic more evenly. No matter how plausible this may seem, it is also dangerous. Whether this is good or not is probably best decided via computational experiments.

A case where it may be necessary to use an improvement based search method is when it is costly to evaluate the objective, e.g. if it has to be done by a simulation procedure. Now, there will probably not be time for enough iterations and the balance has to be toward intensification over exploration. It also becomes much more important to select the next iterate carefully, e.g. by using methods from experimental design. One measure that definitely should be taken is to guarantee that the flow is changed for at least one commodity. This can be achieved by a straightforward reduced cost analysis for each destination.

In conclusion, the metaheuristic approach is very flexible and has been succesfully applied to find good solutions to several STEP instances. But it comes with a major drawback: it is not known how far from an optimal solution we are. Therefore, optimality can not be guaranteed, nor verified. Because of this we only consider exact solution methods from now on.

2.2.3 An Exact Approach: MILP Formulations of STEP

It is actually not trivial to develop exact MILP formulations for shortest path routing design problems (SPRD) such as STEP. To the best of our knowledge, the first MILP formulations were given in [15], without ECMP and in [63] with ECMP. These research reports have later been published in [18] and [64, 98], respectively. The ECMP is modelled via scenario-based splitting constraints in [63]. These constraints were improved in later papers, e.g. in [40] and [82] (due to Tomaszewski). Nowadays, most models use the constraints that are presented below in model (SPRD). This improvement significantly stregthens and reduce the size of the models. The reader is referred to the book [80] for an overview of SPRD problems where several examples are given and the structure of early models is revealed. Some MILP approaches encountered in the litterature include [9, 19, 84, 64, 96, 42, 79, 40].

In principle, an SPRD problem may be handled as an ordinary network design problem with side constraints on the routing. An important part in this modelling approach is the metric. There are two approaches to handle it: directly by including the metric and SPR constraints in the model or indirectly by prohibiting infesible routing patterns.

At first, the direct approach to use the metric in the model seems natural. This is used in all early SPRD models were the protocol is simulated by introducing a set of variables for the arc weights, w, and node potentials, π. These variables are then used together with binary shortest path routing design variables, y, to form the following bilinear constraints

$$\begin{array}{rcll} w_{ij} + \pi_i^l - \pi_j^l + y_{ij}^l & \geq & 1 & (i,j) \in A, l \in L \\ \left(w_{ij} + \pi_i^l - \pi_j^l\right) y_{ij}^l & = & 0 & (i,j) \in A, l \in L \\ w_{ij} & \geq & 1 & (i,j) \in A. \end{array} \tag{2.1}$$

Any $0/1$ fixation of the binary variables corresponding to an acyclic ingraph yields an instance of the inverse shortest path routing (ISPR) problem covered in detail in Chapter 3 trough 7. To obtain a linear formulation, these constraints can be linearilized with big-M constraints, for example as follows.

$$\begin{array}{rcll} w_{ij} + \pi_i^l - \pi_j^l & \geq & 1 - y_{ij}^l & (i,j) \in A, l \in L \\ w_{ij} + \pi_i^l - \pi_j^l & \leq & M(1 - y_{ij}^l) & (i,j) \in A, l \in L \\ w_{ij} & \geq & 1 & (i,j) \in A. \end{array} \tag{2.2}$$

The major drawback with this approach is the big-M:s, which in general weakens the LP relaxation. This is not necessarily crucial when M is small, as it often is in practice. However, for SPRD problems the big-M may have to be as large as a shortest longest path in the graph, cf. [9]. This big-M is typically huge, (recall that arc weights can be as large as $2^{16} - 1$ in the OSPF protocol). Hence, *the LP relaxation of early SPRD models are typically very weak.*

Several researchers realized the problem with the naive modelling approach that includes the weights in the model. This lead to the current (indirect) approach to solve SPRD problems exactly; now the shortest path routing constraints (2.1) that cause the big-M:s are replaced by shortest path compability constraints that only involve binary design variables. We believe that the first model without the weight variables was given for SPRD without ECMP in [19] (later published in [16]).

The Mathematical Framework

A brief description of what currently seems to be the most promising exact solution method for SPRD problems is outlined here. The mathematical model given below is the core of most recent SPRD models. We want to concentrate solely on the shortest path routing aspect and ECMP. Therefore, no objective function is included, nor any other constraints, e.g. capacity related constraints.

The definitions required to model SPRD feasibility are as follows. As usual, $G = (N, A)$ is a directed graph. There is a set of destinations, $L \subseteq N$, and a set of origins for each destination, $N_l \subseteq N$. For an OD pair, $(o, d) \in N \times L$, there is a demand on how much traffic that should be routed from the origin, o, to the destination, d.

Two kinds of variables are used: binary shortest path routing variables, y, and continuous flow variables, x. For a destination node, $l \in L$, and an arc $(i, j) \in A$,

$$y_{ij}^l = \begin{cases} 1 & \text{if } (i,j) \text{ is on a shortest path to } l, \\ 0 & \text{otherwise.} \end{cases} \tag{2.3}$$

For a destination, $l \in L$, an origin, $k \in N_l$, and an arc $(i, j) \in A$ the variable x_{ij}^{kl} is the fraction of the traffic demand from k to l that is routed along (i, j).

To describe the collection of feasible routing patterns, a mapping, $Y^l(w) : \mathbb{N}^{|A|} \rightarrow \mathbb{B}^{|A|}$, is introduced. For any vector of link weights, $w \geq 1$, $Y^l(w)$ is the incidence vector of the induced (acyclic) ingraph to node l. That is,

$$Y^l(w)_{ij} = \begin{cases} 1 & \text{if } (i,j) \text{ is in some shortest path (w.r.t. } w) \text{ to node } l \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

The ingraph induced by $Y^l(w)$ can be thought of as the union of all reverse shortest path trees rooted at l that are obtained from w. Since $w \geq 1$, no induced ingraph can contain a directed cycle.

Using the map $Y^l(w)$ defined above, a feasible routing pattern can be defined as a collection of ingraphs that are obtained from the same vector of link weights. This yields that the set of incidence vectors corresponding to collections of simultaneously realizable ingraphs becomes

$$\mathcal{Y} = \left\{ y = \left(y^l\right)_{l \in L} \mid \text{there exist a } w \in \mathbb{N}^{|A|} \text{ such that } y^l = Y^l(w) \text{ for all } l \in L \right\}. \tag{2.5}$$

This is sufficient to give a matematical model of the core of an SPRD problem.

$$\begin{array}{rcll}
\sum_{(j,i)\in A} x_{ji}^{kl} - \sum_{(i,j)\in A} x_{ji}^{kl} & = & b_i^{kl}, & i \in N,\ k \in N_l,\ l \in L \\
x_{ij}^{kl} & \leq & y_{ij}^{l} & (i,j) \in A,\ k \in N_l,\ l \in L \\
x_{ij}^{kl} - x_{ij'}^{kl} & \leq & 1 - y_{ij'}^{l} & (i,j) \in A,\ (i,j') \in A,\ k \in N_l,\ l \in L \\
y & \in & \mathcal{Y} & \\
x_{ij}^{kl} & \in & [0,1] & (i,j) \in A,\ k \in N_l,\ l \in L \\
y_{ij}^{l} & \in & \mathbb{B} & (i,j) \in A,\ l \in L,
\end{array} \tag{SPRD}$$

where,

$$b_i^{kl} = \begin{cases} -1 & \text{if } i = k \\ 1 & \text{if } i = l \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

Let us verify that this model correctly models an SPRD problem. The node balance constraints implies that the flow variables, x^{kl}, form paths from node k to node l. This fact and the coupling constraints implies that the corresponding shortest path routing variables, y^l, form an ingraph to node l. Its acyclicity is forced by $y \in \mathcal{Y}$. The traffic split constraints guarantees that the flow w.r.t. an OD-pair is equal on all arcs that emanate from a node and carry a positive amount of flow. Finally, the routing compability set $\mathcal{Y}$ makes sure that there are no routing conflicts between ingraphs.

To actually solve (SPRD) with a MILP solver, such as CPLEX, the compability set $\mathcal{Y}$ has to be described by linear inequalities. In the first part of this thesis routing conflicts between ingraphs will be analysed and described via inverse shortest path routing problems. This yields a characterization of a sufficiently large class of routing conflicts. Prohibing all these conflicts yields a description of the compability set $\mathcal{Y}$ with linear inequalities (given that the remaining constraints in (SPRD) are satisfied).

Since the number of linear constraints required to describe $\mathcal{Y}$ is generally exponential in the size of the graph it is in practice not possible to include all constraints in (SPRD). The natural approach to solve (SPRD) is with a constraint generation and branch and bound and cut (B&B&C) scheme.

To use this approach one has to be able to determine if a tentative routing pattern is realizable. That is, given a y, determine if $y \in \mathcal{Y}$. When $y \notin \mathcal{Y}$, a valid inequality that separates y from $\mathcal{Y}$ must be provided. In theory, it is sufficient to be able to decide if $y \in \mathcal{Y}$ for a binary y that solves the remaining constraints of (SPRD). In practice, it is however necessary to also separate fractional y:s from $\mathcal{Y}$. These issues are further addressed in Chapter 8 where some special separation problems are considered.

The approach oulined above will be more thoroughly presented in Chapter 8, but first inverse shortest path routing problems will be analysed to get a profound understanding of routing conflicts. This is necessary in order to derive good valid inequalities for $\mathcal{Y}$ and efficient separation algorithms.

3 Inverse Shortest Path Routing

In the introduction some approaches to solve traffic engineering problems for IP networks were considered. Currently, most MILP models do not include the administrative weights. As a consequence, one must be able to determine if a (partial) routing pattern is realizable in an SPR protocol. This leads to special kinds of inverse (partial) shortest path routing problems that we model in this chapter and analyze further in the following chapters.

The outline of the chapter is as follows. An introduction to the ordinary shortest path problem including optimality conditions is given in Section 3.1. Then, inverse shortest path (ISP) problems are considered in general and a model based on the shortest path reduced cost optimality conditions is presented in Section 3.2. A closely related problem in the IP network design context is the inverse shortest path routing (ISPR) problem which is considered in Section 3.3. In Section 3.4 some new and old models are introduced for ISPR. Finally, in Section 3.5 some associated algorithms are given.

3.1 The Shortest Path Problem

One of the most fundamental problems in combinatorial optimization is probably the ordinary shortest path problem, where a graph, $G = (N, A)$, and a set of arc costs, c, are given. The problem is to determine the shortest paths w.r.t. these arc costs from a root node, s, to some (possibly all) other destination node(s).

For our purposes, we may assume that c is nonnegative and that there is a path from s to all other nodes in G. The following flow based formulation of the shortest path problem is well known.

$$\begin{array}{llrcll}
\min & \sum_{(i,j)\in A} c_{ij}x_{ij} & & & & \\
s.t. & \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} & = & b_i & i \in N & \\
 & x_{ij} & \geq & 0 & (i,j) \in A. &
\end{array} \tag{SPP}$$

The x variables can be interpreted as the flow. The node balances, b, depend on which version of the shortest path problem we consider. When a shortest path from s to t is sought, let

$$b_i = \begin{cases} 1 & \text{if } i = s \\ -1 & \text{if } i = t \\ 0 & \text{otherwise,} \end{cases} \tag{3.1}$$

and when a shortest path from s to all other nodes are sought, set

$$b_i = \begin{cases} n-1 & \text{if } i = s \\ -1 & \text{otherwise.} \end{cases} \tag{3.2}$$

The properties of the polyhedron formed by the feasible set of solutions to (SPP) can be considered as common knowledge. This polyhedron can be decomposed into a polytope and a pointed cone. Since we assumed that $c \geq 0$, there exist an optimal solution within the polytope. In an extremal point of the polytope we know that all x are integral and that the arcs associated with variables that have a strictly positive value forms a shortest path tree rooted at s. From linear programming theory we also have that a convex combination of optimal solutions is an optimal solution. This implies that, in any optimal solution, an arc associated with an x variable that is strictly positive is on a shortest path from s to some other node. Further, all shortest paths are unique if and only if there is no fractional optimal solution.

Several algorithms exists for solving the shortest path problem efficiently. The most well known is probably Dijkstras algorithm, originally given in [41]. Textbook descriptions of this and other algorithms along with different implementations and complexity analysis, are given in [2].

Most shortest path algorithms are based on duality and the solution of the reduced cost optimality conditions that we present below. The terminology, definitions and theorems can be found in [52].

Definition 3.1
A node potential, π, is feasible if

$$c_{ij} + \pi_i - \pi_j \geq 0, \quad (i,j) \in A. \tag{3.3}$$

The left hand side in (3.3) can be denoted by $\hat{c}_{ij}$ and is called the reduced cost of arc (i,j). A shortest path solution can be deduced from the following well known theorem, which is just an adaption of dual feasibility and complementary slackness from the theory of linear programming to (SPP).

Theorem 3.1
An arc (i, j) is in some shortest path from the root node, s, if and only if there exist a feasible node potential, π, where

$$c_{ij} + \pi_i - \pi_j = 0. \tag{3.4}$$

In a primal-dual pair, (x, π), the dual solution, π, is dual feasible if all reduced costs are nonnegative. An arc is on a shortest path if and only if the reduced cost of that arc is 0, such an arc is said to be tight. These optimality conditions may be used in the modelling of ISP and related problems which we will see later on.

If the dual variable of the root s is set to 0, then the dual variable π_i may be interpreted as a lower bound on the shortest path distance from s to i in any dual feasible solution. The maximal lower bound yields the actual distance, which is also obtained by the dual variable value in an optimal solution. In general, $\pi_j - \pi_i$ is a lower bound on the distance from i to j and the actual distance is the maximal lower bound.

We also have that for any directed path p from k to l

$$\sum_{(i,j)\in p} \hat{c}_{ij} = \sum_{(i,j)\in p} (c_{ij} + \pi_i - \pi_j) = \sum_{(i,j)\in p} c_{ij} + \pi_k - \pi_l. \tag{3.5}$$

Especially, if p is a shortest path, both sides are 0 since all reduced costs are 0.

Given the above well known facts about the ordinary shortest path problem, models of the inverse shortest problems can be formulated. These are relevant to SPR applications as a starting point when mathematical models are developed.

3.2 Inverse Shortest Path Problems

The ordinary shortest path problem is familiar. Given arc costs, determine the shortest paths. Conceptually, the inverse shortest path problem is just the other way around. Given a collection of paths, $\mathcal{Q}$, the problem is to determine arc costs such that all paths in $\mathcal{Q}$ become shortest paths. To obtain an optimization problem, additional information have to be specified, e.g. an objective function. Some problems also come with extra requirements on the paths not in $\mathcal{Q}$; this is especially true when it comes to routing. We concretize this below.

Burton and Toint [32] is the first reference to ISP that we have encountered in the litterature. They motivate the relevance of the problem by two practical applications, mathematical traffic modelling and seismic tomography which leads to their choice of objective, to minimize the deviation from some ideal arc costs. Their formulation is as follows.

Given a directed graph, $G = (N, A)$, a nonnegative cost vector, $\tilde{c}$, and a path collection $\mathcal{Q}$, find a minimal modification of $\tilde{c}$ such that all paths in $\mathcal{Q}$ are shortest paths. Their mathematical model of ISP is

$$\begin{aligned}
\min\ & \frac{1}{2} \sum_{(i,j)\in A} (c_{ij} - \tilde{c}_{ij})^2 \\
s.t.\ & \sum_{(i,j)\in q} c_{ij} \geq \sum_{(i,j)\in p} c_{ij} \qquad q \in \mathcal{Q}_p,\ p \in \mathcal{Q} \\
& c_{ij} \geq 0 \qquad (i,j) \in A,
\end{aligned} \tag{ISP-BT}$$

where $\mathcal{Q}_p$ is the collection of all paths with the same origin and destination as the path p. The model is a convex quadratic program since the objective is quadratic and all constraints are linear. Note that the number of constraints is potentially exponential in the size of the graph since paths are enumerated in the first constraint in (ISP-BT). In practice, it may be necessary to use a constraint generation scheme to solve (ISP-BT).

In the complexity analysis in [32], it is pointed out that it is possible to overcome the problem with exponentially many constraints by modelling the distance between the pairs of nodes instead. This yields a polynomial number of constraints. Perhaps it is better to say that we can use the dual variables and the optimality conditions of the shortest path problem in the modelling instead. We give such a model of ISP in the next section.

3.2.1 A Polynomial Formulation of ISP

Recall that $G = (N, A)$ is a directed graph and $\tilde{c}$ a nonnegative cost vector. We seek a minimal modification of $\tilde{c}$ such that all paths in a collection $\mathcal{Q}$ are shortest paths.

The polynomial formulation of ISP we present is based on the reduced cost optimality conditions from the ordinary shortest path problem, cf. Theorem 3.1. This modelling approach is based on arcs and not paths, which is the source of the exponential number of constraints. It is shown how to transform the collection of required shortest paths $\mathcal{Q}$ to a collection of subgraphs of G that represents the collection of shortest paths below.

Call a subgraph that represents required shortest paths a shortest path graph, or an SP-graph for short. The indata to our formulation of ISP will be a family of SP-graphs instead of the path collection $\mathcal{Q}$. This impose no loss of generality.

Let $\{(o_k, d_k)\}_{k\in K} \subseteq N \times N$ be the set of OD-pairs induced by $\mathcal{Q}$. Denote the OD-pairs with the same destination, l, by $D_l = \{k \mid d_k = l\}$ and the set of all destinations by L, where $N \supseteq L = \bigcup_k d_k$. Sometimes, we use $o(k)$ and $d(k)$ instead of o_k and d_k, respectively, when that is notationally more convenient.

The collection of all simple paths between o_k and d_k in G is denoted by $\mathcal{P}_k$ and the desired paths by $\mathcal{Q}_k = \mathcal{Q} \cap \mathcal{P}_k$. The collection of all (relevant) paths in G is $\mathcal{P} = \bigcup_k \mathcal{P}_k$. Let $N(A)$ be the set of nodes spanned by the set of arcs, A, and $N(p)$ the set of nodes spanned by the path p. Consider the map, $\bar{C} : 2^{\mathcal{P}} \to 2^A$, that yields the set of arcs covered by some path in a collection of paths, formally defined by

$$\bar{C}(\mathcal{S}) = \{(i,j) \mid (i,j) \in p \text{ for some } p \in \mathcal{S}.\}\,. \tag{3.6}$$

This map can be used to transform all path collections into arc sets, as follows,

$$S_k = \bar{C}\left(\mathcal{Q}_k\right), \quad k \in K. \tag{3.7}$$

Now, all sets of arcs originating from path collections with the same destination, l, can be collected in a shortest path (in)graphs, $G_l = (N(A_l), A_l)$, to node l where

$$A_l = \bigcup_{k \in D_l} S_k, \quad l \in L. \tag{3.8}$$

The optimality conditions from the shortest path problem states that an arc is on a shortest path if and only if there is a feasible node potential where the reduced cost of the arc is 0. Introduce a node potential, π^l, for each shortest path graph, G_l.

The following two sets of constraints,

$$c_{ij} + \pi_i^l - \pi_j^l \geq 0, \quad (i,j) \notin A_l. \tag{3.9}$$

and

$$c_{ij} + \pi_i^l - \pi_j^l = 0, \quad (i,j) \in A_l. \tag{3.10}$$

makes π^l feasible and also guarantee that the paths in S_k are shortest paths for $k \in D_l$. This yields the following, polynomial, arc based formulation of ISP

$$\begin{array}{lrcll}
\min & \frac{1}{2} \sum\limits_{(i,j) \in A} (c_{ij} - \tilde{c}_{ij})^2 & & & \\
s.t. & c_{ij} + \pi_i^l - \pi_j^l & = & 0 & (i,j) \in A_l,\ l \in L \\
 & c_{ij} + \pi_i^l - \pi_j^l & \geq & 0 & (i,j) \notin A_l,\ l \in L \\
 & c_{ij} & \geq & 0 & (i,j) \in A.
\end{array} \tag{ISP-A}$$

This alternative modelling technique yields that the ISP is solvable in polynomial time by an interior point algorithm. It is however not clear which is more efficient in practice: constraint generation or direct solution of the polynomial formulation.

Theorem 3.2 (Burton and Toint [32])
ISP is solvable in polynomial time.

In our opinion (ISP-A) has two major advantages over the formulation in (ISP-BT); it is in general smaller and the structure is better revealed. We need to mention that the comment about using shortest path distances in [32] was due to Vavasis, cf. Section 5 in [32]. Essentially, our model is what is outlined there, adapted to SP-graphs.

3.2.2 Some Comments on SP-Graphs

The SP-graphs were introduced by Broström and Holmberg in [25] and has since been used elsewhere, e.g. [23, 26, 14]. As seen above, they can sometimes be used instead of collections of paths. When that is possible, we strongly encourage the usage of SP-graphs to present input (output), since this makes it more evident which paths should be (are) shortest paths. The collection of required shortest paths can also be large and very often contains overlaps, and this “problem” is reduced by using SP-graphs.

Whether to use path collections or SP-graphs in a mathematical model depends on other factors as well, e.g. the strength and the size of the formulation. In the above case, both formulations are equally strong, but the latter is significantly smaller. Several linear integer programming formulations of network design and routing problems become stronger and larger when a path formulation is used. In that case, it is a tradeoff, but often the stronger path formulation is preferred.

Formally, an SP-graph, $G_l = (N(A_l), A_l)$, is defined in [25] as a subgraph of G with the following properties:

1. The set of origins in G_l are the nodes with indegree 0.
2. The set of destinations in G_l are the nodes with outdegree 0.
3. There is a directed path in G_l from every origin to every destination.
4. There is no directed cycle in G_l.

These properties implies that there is at least one origin, one destination and no isolated nodes in an SP-graph.

Two important special cases which are related to the structure of optimal solutions to shortest path problems are the rooted ingraph and the rooted outgraph where we have a single destination, and origin, respectively. The graphs defined in (3.8) are examples of rooted ingraphs.

In practice, we will use ingraphs since this gives us the additional information that from every node there is a path to the root. This can be very useful, both in the modelling of the ISPR problem and of STEP, to be discussed later, cf. Section 3.3 and Chapter 8. However, SP-graphs are more general, so we will give results for SP-graphs when that is suitable. Also, in other applications it may be more natural to work with some other structure than ingraphs.

3.3 Inverse Shortest Path Routing Problems

When routing by shortest paths is considered in applications, e.g. telecommunications, it may be crucial that no path that is not specified to be shortest becomes a shortest path. This can for instance yield undesired traffic along such a path. In an ISPR problem it has to be guaranteed that no undesired path becomes a shortest path. The above models are certainly a good foundation, but some modifications are required. We use the notation from the previous section and formulate the ISPR problem below.

Our primary concern is telecommunication applications. Therefore it is assumed that a path that is not in the collection of desired shortest paths, $\mathcal{Q}_k$, must not be a shortest path.

Depending on the application and which routing protocol is used, there may be restrictions on which link weights that are allowed, e.g. in OSPF (IS-IS) all link weights must be positive integers not larger than 65535 (15) cf. [74]. There may also be upper bounds on path lengths, e.g. in RIP the length of a path must not exceed 15. However, in most applications, there is no natural objective function, all *feasible* weight settings are equally good. Thus, our primary concern is the feasible set of weights.

Assume that the restrictions implied by the application and the routing protocol is modelled via the set W and denote the link weights by w. This yields the following general feasibility model for ISPR.

$$\begin{aligned} \sum_{(i,j)\in q} w_{ij} &= \sum_{(i,j)\in p} w_{ij} && q,p \in \mathcal{Q}_k,\ k \in K \\ \sum_{(i,j)\in q} w_{ij} &\geq \sum_{(i,j)\in p} w_{ij} + \epsilon && q \in \mathcal{P}_k \setminus \mathcal{Q}_k,\ p \in \mathcal{Q}_k,\ k \in K \\ w &\in W. \end{aligned} \tag{ISPR-G}$$

Here, ϵ is a strictly positive number that must be determined by the application, e.g. in OSPF the integrality of weights yields that $\epsilon = 1$ can be used.

It may be undesirable to handle the restrictions implied by the routing protocol explicitly by the set W. An implicit strategy is to use the objective function instead. This has two advantages. First, the description of the feasible region may be better revealed which can make it easier to find a feasible solution. Second, if an instance is infeasible, it is easy to find out if the infeasibility is due to a conflict among the collection of desired paths or a restriction implied by the routing protocol, like an upper bound. This approach is demonstrated for OSPF and IS-IS below. The model becomes

$$\begin{aligned} \min \max\ & w_{ij} \\ s.t. \sum_{(i,j)\in q} w_{ij} &= \sum_{(i,j)\in p} w_{ij} && q,p \in \mathcal{Q}_k,\ k \in K \\ \sum_{(i,j)\in q} w_{ij} &\geq \sum_{(i,j)\in p} w_{ij} + 1 && q \in \mathcal{P}_k \setminus \mathcal{Q}_k,\ p \in \mathcal{Q}_k,\ k \in K \\ w_{ij} &\geq 1 && (i,j) \in A \\ w_{ij} &\in \mathbb{Z} && (i,j) \in A. \end{aligned} \tag{ISPR-MIN-W}$$

If (ISPR-MIN-W) is feasible the optimal solution is a feasible weight setting in OSPF (IS-IS) if and only if the optimal value is not larger than 65535 (15). If it is infeasible, the infeasibility is due to a conflict between some desired and undesired collection of shortest paths. In [17] it is shown that (ISPR-MIN-W) and the similar problem to minimize the maximal path length are NP-hard, actually they are even APX-hard.

Several variants of ISPR have been considered in the litterature. The flexibility in the set W yields that all of them are essentially covered by (ISPR-G). Also note that many constraints may be handled by the desired collections of shortest paths. For instance, one of the most studied variants of ISPR is OSPF with unique shortest paths. This may be modelled via the SP-graphs G_l induced by $\mathcal{Q}$. If the arcs in the SP-graph, A_l, is a reversly directed arborescence then all shortest paths must be unique. Alternativly, put in terms of the collections of paths, all collections $\mathcal{Q}_k$ must be singletons. Another common variant is where the desired shortest paths from i to j should be the reversal of the desired shortest paths from j to i for all node pairs. This is easily handled by including the forward path in G_j and the backward path in G_i, cf. [26].

From now on, only feasibility of (ISPR-MIN-W) is considered. OSPF inrealizability due to large weights is of no practical concern, cf. [9] and the rounding procedure in [6].

A drawback of model (ISPR-MIN-W) is that the number of constraints may be exponential in the size of the graph. An equivalent model where the number of constraints is polynomially bounded is developed in the next section.

3.4 Mathematical Formulations of ISPR Problems

The contribution of this section is a new modelling idea to develop a polynomial formulation of the ISPR problem. That is, to find a point in the feasible set of (ISPR-MIN-W), or proving that it is empty. Our model is derived from the straightforward feasibility model in [26]. Similar path based feasibility models have been given by several authors, e.g. [9, 6]. In fact, the path based formulations seem more popular, despite the fact that the number of constraints may be exponential.

The conditions are as in the previous section, $G = (N, A)$ is a directed graph and we seek a set of administrative weights, w, such that the paths in $\mathcal{Q} = \bigcup_{k \in K} \mathcal{Q}_k$ are shortest paths. The following model was given in [26].

$$\begin{array}{rcll}
\sum\limits_{(i,j)\in q} w_{ij} & = & \sum\limits_{(i,j)\in p} w_{ij} & q, p \in \mathcal{Q}_k,\ k \in K \\
\sum\limits_{(i,j)\in q} w_{ij} & \geq & \sum\limits_{(i,j)\in p} w_{ij} + 1 & q \in \mathcal{P}_k \setminus \mathcal{Q}_k,\ p \in \mathcal{Q}_k,\ k \in K \\
w_{ij} & \geq & 1 & (i,j) \in A \\
w_{ij} & \in & \mathbb{Z} & (i,j) \in A.
\end{array} \qquad \text{(ISPR-F)}$$

Any feasible solution to (ISPR-F) corresponds to a set of administrative weights, w, that makes all paths in $\mathcal{Q}$ shortest paths. We say that the metric, w, is compatible with $\mathcal{Q}$.

If we wish to avoid the exponential number of constraints it is not feasible to enumerate paths. Therefore, the arc-based reduced cost optimality conditions are used instead. There are however some consistency issues that have to be take into account when a collection of paths are transformed into sets of arcs. Consider the following two examples.

Example 3.1

Consider the graph in Figure 3.1 and the OD pair $(o_1, d_1) = (1, 7)$. Let

$$\mathcal{Q}_1^1 = \{1-2-4-5-7,\ 1-3-4-6-7\} \text{ and}$$
$$\mathcal{Q}_1^2 = \{1-2-4-6-7,\ 1-3-4-5-7\}.$$

The path collection $\mathcal{Q}_1^1$ alone implies that the collection of all paths, $\mathcal{P}_1$, between node 1 and 7, are shortest paths. Hence, $\mathcal{Q}_1^1$ can not be consistent in the inverse path routing sence. This conclusion holds for $\mathcal{Q}_1^2$ as well, since it also induce all paths in $\mathcal{P}_1$ to be shortest paths.

However, the path collection $\mathcal{Q}_1 \cup \mathcal{Q}_2 = \mathcal{P}_1$ is consistent, since it does not induce any other shortest paths.

Also note that these path collections would not cause a conflict in the ordinary ISP, since there, we do not mind if some unspecified path (also) becomes a shortest path.

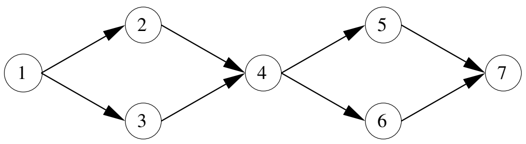


Figure 3.1: *An example where two collections of paths are not consistent.*

Example 3.2

Consider the graph in Figure 3.1 from Example 3.1 and let

$$\mathcal{Q}_1 = \{1-2-4-5-7\},\ \mathcal{Q}_2 = \{3-4-6-7\} \text{ and } \mathcal{Q}_3 = \{3-4-5-7\}.$$

All path collections have the same destination so we would like to use $\bar{C}$ to transform these collections of paths into arc sets as we did in ISP, cf. equation (3.8). First, let

$$S_k = \bar{C}\left(\mathcal{Q}_k\right), \quad k = 1, \ldots, 3. \tag{3.11}$$

Now consider the SP-graphs in Figure 3.2, defined by $G_1 = (N(A_7^{12}), A_7^{12})$ and $G_2 = (N(A_7^{13}), A_7^{13})$, where

$$A_7^{12} = S_1 \cup S_2 \text{ and } A_7^{13} = S_1 \cup S_3.$$

Here we see that the aggregation of S_1 and S_2 implies that $3-4-5-7$ should be a shortest path, but this path is not induced by $\mathcal{Q}_1$, nor $\mathcal{Q}_2$. However, if we combine S_1 and

S_3 we will not create a path that is not induced by $\mathcal{Q}_1$ or $\mathcal{Q}_3$. Again, note that this was not a problem in ISP.

The cause of the problem with the aggregation of S_1 and S_2 is that they are not subpath consistent as defined in [26]. However, S_1 and S_3 are subpath consistent and may be combined, cf. [26].

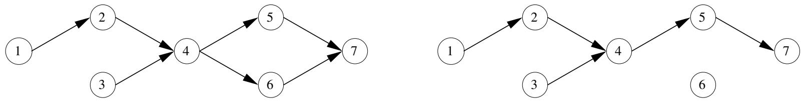


Figure 3.2: *SP-graphs with arc sets A_7^{12} and A_7^{13}.*

The idea of the conflicts presented in the two examples above are generalized in the next subsection. Necessary conditions for collections of paths to be transformable and combinable into an equivalent SP-graph are given. When these are not satisfied, (ISPR-F) is infeasible. Thereafter, when this consistency issue is settled, we continue to derive our polynomial formulations of ISPR.

3.4.1 Internal Consistency

Recall that $\mathcal{P}_k$ denotes the collection of all simple paths between o_k and d_k in G. Also, let $\mathcal{P}_{st}$ be the collection of all simple paths from s to t. The collection of desired shortest paths, $\mathcal{Q}$, can be partitioned as $\mathcal{Q} = \bigcup_{k \in K} \mathcal{Q}_k$, where $\mathcal{Q}_k = \mathcal{Q} \cap \mathcal{P}_k$.

As before, the map $\bar{C} : 2^{\mathcal{P}} \to 2^A$ yields all arcs covered by a collection of paths,

$$\bar{C}(\mathcal{S}) = \{(i,j) \mid (i,j) \in p \text{ for some } p \in \mathcal{S}\}. \tag{3.12}$$

For a collection of paths, $\mathcal{S}$, an arc is in $\bar{C}(\mathcal{S})$ if it is in some of the paths in $\mathcal{S}$. Also define the (right inverse) map $\bar{P} : 2^A \to 2^{\mathcal{P}}$ that yields all the collection of paths that only use arcs in the arc set U as

$$\bar{P}(U) = \left\{p \in \mathcal{P} \mid \bar{C}(\{p\}) \subseteq U\right\}. \tag{3.13}$$

For a set of arcs, U, a path is in $\bar{P}(U)$ if all its arcs are in U. We obviously have $\bar{C}\left(\bar{P}(U)\right) = U$ for any set of arcs. Also note that $\mathcal{S} \subseteq \bar{P}\left(\bar{C}(\mathcal{S})\right)$, but not necessarily $\mathcal{S} = \bar{P}\left(\bar{C}(\mathcal{S})\right)$, since a path in $\mathcal{S}$ is covered by its own arcs, but equality only holds when no additional (s,t)-path is induced, cf. Example 3.1 above. When a collection of (s,t)-paths does not induce any other path to be shortest, we say that it is complete. Formally we have the following.

Definition 3.2
A collection of (s,t)-paths, $\mathcal{S}$, is complete if

$$\mathcal{S} = \bar{P}\left(\bar{C}\left(\mathcal{S}\right)\right) \cap \mathcal{P}_{st}. \tag{3.14}$$

Example 3.1: continued

Recall from above that both path collections

$$\mathcal{Q}_1^1 = \{1-2-4-5-7,\ 1-3-4-6-7\} \text{ and}$$
$$\mathcal{Q}_1^2 = \{1-2-4-6-7,\ 1-3-4-5-7\}$$

alone imply that all paths between 1 and 7 are shortest paths. Hence, neither $\mathcal{Q}_1^1$, nor $\mathcal{Q}_1^2$, is complete. However, $\mathcal{Q}_1 \cup \mathcal{Q}_2 = \mathcal{P}_1$ is complete.

Completeness of all path collections is a necessary condition for the feasibility of system (ISPR-F) as the following lemma states.

Lemma 3.1
System (ISPR-F) *is infeasible if $\mathcal{Q}_k$ is not complete for some k.*

Proof: If $\mathcal{Q}_k$ is not complete, there is a path $q \in \bar{P}\left(\bar{C}(\mathcal{Q}_k)\right) \setminus \mathcal{Q}_k$. That is, a path that only uses arcs in the set of arcs induced by shortest paths. This implies that q has the same length as a path in $\mathcal{Q}_k$ by Theorem 3.1. But q is in $\mathcal{P}_k \setminus \mathcal{Q}_k$ since $\bar{P}\left(\bar{C}(\mathcal{Q}_k)\right) \subseteq \mathcal{P}_k$ and

$$\sum_{(i,j)\in q} w_{ij} = \sum_{(i,j)\in p} w_{ij} \not\geq \sum_{(i,j)\in p} w_{ij} + 1, \quad p \in \mathcal{Q}_k. \tag{3.15}$$

Hence, (ISPR-F) is infeasible. □

When $\mathcal{Q}_k$ is complete for all k, the map $\bar{C}$ may be used to transform all collections of paths to sets of arcs. Let S_k be the set of arcs induced by the corresponding collection of paths,

$$S_k = \bar{C}\left(\mathcal{Q}_k\right), \quad k \in K. \tag{3.16}$$

All SP-graphs induced by S_k are single origin, single destination SP-graphs that can be used to solve an SP-graph based version of (ISPR-F), cf. model (ISPR-AP) and also P3 and P4 in [26]. However, it was demonstrated in [26] that ISRP can be solved much faster when SP-graphs are combined. Therefore, we would like to bunch some SP-graphs together into larger SP-graphs whenever that is possible.

We deviate slightly from the notation used in the previous section since SP-graphs will not longer necessarily be ingraphs. Hence, the index l can no longer be interpreted as the destination. The set L is now an indexing of the SP-graphs induced by S_k and the sets $D_l \subseteq K$ are sets of OD-pair indices such that $\bigcup_{l\in L} D_l = K$.

Define the arcs in the SP-graph G_l for each $l \in L$ as follows

$$A_l = \bigcup_{k\in D_l} S_k, \quad l \in L. \tag{3.17}$$

This aggregation potentially yields paths induced by A_l that are not in $\mathcal{Q}_k$, nor induced by $\mathcal{Q}_k$ for some k, cf. Example 3.2. Therefore the sets D_l have to be choosen carefully, cf. Proposition 3.1.

To avoid invalid combination of SP-graphs we introduce the definition of internal consistency for SP-graphs below. It is closely related to the concept subpath consistency in [26], but we use our mappings from above to define it. The definition of the index set of relevant OD-pairs in A_l is also required, let

$$R_l = \{k \mid d_k \text{ is reachable from } o_k \text{ in } A_l\}. \tag{3.18}$$

So, k is in R_l if there is a path from o_k to d_k in A_l, note that $D_l \subseteq R_l$.

Definition 3.3
A family of SP-graphs, $\{G_l\}_{l\in L}$, defined by

$$\begin{aligned} G_l &= (N(A_l), A_l),\ l \in L, \\ S_k &= \bar{C}(\mathcal{Q}_k),\ k \in K, \text{ and} \\ A_l &= \bigcup_{k\in D_l} S_k \end{aligned}$$

is internally consistent if $\mathcal{Q}_k$ is complete for all k and

$$S_k = \bar{C}(\mathcal{P}_k) \cap A_l, \quad k \in R_l,\ l \in L. \tag{3.19}$$

Condition (3.19) in the above definition states that no SP-graph must induce additional shortest paths for any relevant OD-pair. Put in terms of combination of SP-graphs in [26] we would say that two subpath inconsistent SP-graphs can not be combined.

Internal consistency of the SP-graphs is a stronger necessary condition for the feasibility of (ISPR-F) then completeness.

Proposition 3.1
If a family of SP-graphs, $\{G_l\}_{l\in L}$, is not internally consistent, then (ISPR-F) *is infeasible.*

Proof: Assume that the family of SP-graphs, $\{G_l\}_{l\in L}$, is not internally consistent. If $\mathcal{Q}_k$ is not complete for all k we are done. Otherwise, there is an $l' \in L$ and a $k' \in R_{l'}$ such that

$$S_{k'} \subset \bar{C}(\mathcal{P}_{k'}) \cap A_{l'}, \tag{3.20}$$

since $S_{k'} \subseteq A_{l'}$ and $S_{k'} \subseteq \bar{C}(\mathcal{P}_{k'})$. This implies that there is a path $q \in \mathcal{P}_{k'} \setminus \mathcal{Q}_{k'}$ that only uses arcs on shortest paths from $o_{k'}$ to $d_{k'}$. This implies that q has the same length as a path $p \in \mathcal{Q}_{k'}$. Hence,

$$\sum_{(i,j)\in q} w_{ij} = \sum_{(i,j)\in p} w_{ij} \not\geq \sum_{(i,j)\in p} w_{ij} + 1. \tag{3.21}$$

and (ISPR-F) is infeasible. □

Above, we have given necessary conditions for collections of paths to be transformable and combinable into equivalent SP-graphs. Hence, we have settled the consistency issues and can continue to derive our polynomial formulation of ISPR. First, we give some practical comments.

Note that we can use the trivial SP-graphs $G_k = (N(S_k), S_k)$, but that is probably inefficient. A better idea is to use the natural idea to collect all S_k with the same destination (origin) into an SP-graph as we did in Section 3.2.1, cf. equation (3.8). We consider an algorithm with this latter approach that determines if a family of SP-graphs is internally consistent in the last section of this chapter.

We think that it is reasonable to assume that the indata is consistent in the sense of Lemma 3.1 but that it is more doubtful if one can assume that the indata is consistent in the sense of Proposition 3.1. However, it is definitly more convenient to give the indata as a set of SP-graphs. Therefore, it is assumed that a family of ingraphs, $\{G_l : l \in L\}$, is given and that it should be determined if there is a feasible solution to (ISPR-F) for the induced sets of desired shortest paths.

3.4.2 A Polynomial Formulation of ISPR

Due to Proposition 3.1 and the algorithms in the next section it is feasible to assume that $\{G_l : l \in L\}$ is a family of internally consistent SP-graphs.

Let us now consider the feasibility problem (ISPR-F) of the induced sets of desired shortest paths. Recall that we are going to use the reduced cost optimality conditions to avoid the exponential number of constraints in (ISPR-F). Define a set of node potentials, π^l, for each $l \in L$. A potential, π^l, is feasible if the reduced costs, $\hat{w}^l_{ij}$, are nonnegative for all arcs, that is,

$$\hat{w}^l_{ij} = w_{ij} + \pi^l_i - \pi^l_j \geq 0, \quad (i,j) \in A,\ l \in L. \tag{3.22}$$

A directed path, p, is a shortest path if and only if all arcs in p are tight, that is, they have reduced cost 0. So for all paths in $\mathcal{Q}$ to be shortest paths, we must have

$$\hat{w}^l_{ij} = w_{ij} + \pi^l_i - \pi^l_j = 0, \quad (i,j) \in A_l,\ l \in L. \tag{3.23}$$

Equivalently, a directed path p is not a shortest path if some arc in p is not tight, that is, some arc in p have reduced cost at least 1, since w is integral. Hence, $p \in \mathcal{P}_{st}$ is not a shortest path if and only if

$$1 \leq \sum_{(i,j)\in p} \hat{w}^l_{ij} = \sum_{(i,j)\in p} \left(w_{ij} + \pi^l_i - \pi^l_j\right) = \sum_{(i,j)\in p} w_{ij} - \pi^l_t + \pi^l_s. \tag{3.24}$$

Constraint (3.24) must hold for all paths in $\mathcal{P}_k \setminus \mathcal{Q}_k$ for all $k \in K$ and $l \in L$ where $k \in D_l$. This yields a combined arc and path based feasibility model for (ISPR-F) which may also be found in [26].

$$\begin{array}{rcll} w_{ij} + \pi_i^l - \pi_j^l & = & 0 & (i,j) \in A_l,\ l \in L \\ \sum_{(i,j)\in p} w_{ij} - \pi_{d(k)}^l + \pi_{o(k)}^l & \geq & 1 & p \in \mathcal{P}_k \setminus \mathcal{Q}_k,\ k \in D_l,\ l \in L \\ w_{ij} & \geq & 1 & (i,j) \in A \\ w_{ij} & \in & \mathbb{Z} & (i,j) \in A. \end{array} \tag{ISPR-AP}$$

Even though the reduced cost optimality conditions are used in (ISPR-AP) paths are still enumerated, so it may contain an exponential number of constraints. To avoid the enumeration of all paths associated with an OD-pair, we make the following crucial observation. If the arc lengths are given by the reduced costs, cf. (3.24), then the length of a path that is not a shortest path w.r.t. w, i.e. in $\mathcal{P}_k \setminus \mathcal{Q}_k$, must be at least 1.*This is accomplished if and only if the shortest path in $\mathcal{P}_k \setminus \mathcal{Q}_k$ is at least of length 1.* We elaborate on this observation to develop a polynomial model below.

Consider an SP-graph, A_l, and an associated OD-pair k. The above observation may be formulated as

$$\sum_{(i,j)\in p} w_{ij} - \pi_{d(k)}^l + \pi_{o(k)}^l \geq 1, \quad p \in \mathcal{P}_k \setminus \mathcal{Q}_k \tag{3.25}$$

if and only if

$$\min_{p \in \mathcal{P}_k \setminus \mathcal{Q}_k} \sum_{(i,j)\in p} \hat{w}_{ij}^l \geq 1. \tag{3.26}$$

The latter constraints in this equivalence, (3.26), can be interpreted as a restricted shortest path problem where some paths are excluded. To solve this problem, observe that it can be divided into several smaller shortest path problems on arcs outside the SP-graph, since all arcs in the SP-graph have reduced cost 0.

Let V_l be the set of node pairs spanned by A_l,

$$V_l = \{(s,t) \mid s,t \in N(A_l)\}\,. \tag{3.27}$$

and denote the collection of (s,t)-paths completely outside A_l by $\mathcal{P}_{st}^l$,

$$\mathcal{P}_{st}^l = \{p \in \mathcal{P}_{st} \mid N(p) \cap N(A_l) = \{s,t\}\} = \mathcal{P}_{st} \setminus \bar{P}(A_l). \tag{3.28}$$

Consider an OD-pair, k, where $(o_k, d_k) = (s,t) \in V_l$ and an (s,t)-path, p, not completely inside A_l, that is, $p \in \mathcal{P}_k \setminus \mathcal{Q}_k$. This means that p contains at least one subpath, q, completely outside A_l. That is, $q \in \mathcal{P}_{uv}^l$ for some $(u,v) \in V_l$. Since, there are two zero length paths, w.r.t. the reduced costs, in A_l from s to u and from v to t, there is an (s,t)-path, with the same length as q. Hence (3.25) and (3.26) holds when q has length at least 1.

This argument implies that the restricted shortest path problem distance in (3.26) can be obtained as the minimum over all simple (u,v)-paths outside A_l over all node pairs $(u,v) \in V_l$. We have,

$$\min_{p \in \mathcal{P}_k \setminus \mathcal{Q}_k} \sum_{(i,j)\in p} \hat{w}^l_{ij} = \min_{(s,t)\in V_l} \min_{p \in \mathcal{P}^l_{st}} \sum_{(i,j)\in p} \hat{w}^l_{ij}. \tag{3.29}$$

Hence, (3.25), (3.26) and (3.29) implies that

$$\sum_{(i,j)\in p} w_{ij} - \pi^l_{d_k} + \pi^l_{o_k} \geq 1, \quad p \in \mathcal{P}_k \setminus \mathcal{Q}_k, \tag{3.25}$$

holds if and only if

$$\min_{p \in \mathcal{P}^l_{st}} \sum_{(i,j)\in p} \hat{w}^l_{ij} \geq 1, \quad (s,t) \in V_l. \tag{3.30}$$

In this equivalence, (3.30) can be interpreted as ordinary shortest path problems on the subgraph of G that contains all paths from s to t and no arcs from A_l.

By the nature of the shortest path problem, all pairs in V_l with a common destination may be handled simultaneously. We utilize the dual version of the shortest path problem to obtain our polynomial formulation.

Let $H_{lt} = (N(A_{lt}), A_{lt})$ be the subgraph that contains all paths to t completely outside G_l that is obtained from

$$A_{lt} = \bigcup_{(s,t)\in V_l} \bar{C}\left(\mathcal{P}^l_{st}\right). \tag{3.31}$$

To model the shortest path problem on H_{lt} with arc cost $\hat{w}$, introduce a second set of node potentials, μ^{lt}. They are feasible if

$$\hat{w}^l_{ij} + \mu^{lt}_i - \mu^{lt}_j \geq 0, \quad (i,j) \in A_{lt}. \tag{3.32}$$

Feasible node potentials yields lower bounds on path distances, therefore the required shortest paths can be forced to be long enough. The distance from s to t in H_{lt} is at least $\mu^{lt}_t - \mu^{lt}_s$ for all $s \in N(A_{lt})$. Any path from s to t also yields an upper bound on $\mu^{lt}_t - \mu^{lt}_s$. To guarantee that no path in H_{lt} is shorter than 1 there must be a feasible node potential where the lower bound is at least 1 since this implies that the upper bound is also at least 1. An upper bound of 0 implies that there is a path outside A_l where all reduced costs are 0 which is not allowed. This is accomplish by the constraints

$$\mu^{lt}_t - \mu^{lt}_s \geq 1, \quad (s,t) \in V_l. \tag{3.33}$$

Summarizing the above yields the following polynomial feasibility model for ISPR

$$
\begin{array}{rcll}
w_{ij} + \pi_i^l - \pi_j^l & = & 0 & (i,j) \in A_l,\ l \in L \\
w_{ij} + \pi_i^l - \pi_j^l & \geq & 0 & (i,j) \in A \setminus A_l,\ l \in L \\
w_{ij} + \pi_i^l - \pi_j^l + \mu_i^{lt} - \mu_j^{lt} & \geq & 0 & (i,j) \in A_{lt},\ t \in N(A_l),\ l \in L \\
\mu_t^{lt} - \mu_s^{lt} & \geq & 1 & (s,t) \in V_l,\ t \in N(A_l),\ l \in L \\
w_{ij} & \geq & 1 & (i,j) \in A \\
w_{ij} & \in & \mathbb{Z} & (i,j) \in A.
\end{array}
\qquad \text{(ISPR-P)}
$$

Thus, from the derivation above we get the following theorem.

Theorem 3.3
If the family of SP-graphs $\{G_l\}_{l \in L}$ is internally consistent, then (ISPR-F) *has a feasible solution if and only if* (ISPR-P) *has a feasible solution.*

Proof: Let the SP-graphs be internally consistent and w a feasible solution to (ISPR-F). The derivation above yields that the solution obtained from w and the tight potentials induced by w is feasible in (ISPR-P). This proves the necessity.

Let (w, π, μ) be a feasible solution to (ISPR-P). It is again clear from the derivation above that w is feasible in (ISPR-F) if π and μ are tight. Suppose that they are not. Then, let $\bar{\pi}$ and $\bar{\mu}$ be the tight potentials induced by w. The integrality of w implies that $\bar{\pi}$ and $\bar{\mu}$ are integral.

Assume that $(w, \bar{\pi}, \bar{\mu})$ is not feasible in (ISPR-P). Since $\bar{\pi}$ and $\bar{\mu}$ are feasible and tight the only possible source of infeasibility is a distance constraint,

$$\bar{\mu}_t^{lt} - \bar{\mu}_s^{lt} \geq 1,$$

for some l and a node pair $(s,t) \in V_l$. When this constraint is violated, $\bar{\mu}_t^{lt} = \bar{\mu}_s^{lt}$. Consider a shortest path, p, from s to t w.r.t. the reduced cost induced by $\bar{\pi}^l$. We have

$$\sum_{(i,j) \in p} \left(w_{ij} + \bar{\pi}_i^l - \bar{\pi}_j^l + \bar{\mu}_t^{lt} - \bar{\mu}_s^{lt} \right) = \sum_{(i,j) \in p} \left(w_{ij} + \bar{\pi}_i^l - \bar{\pi}_j^l \right) - \bar{\mu}_s^{lt} + \bar{\mu}_t^{lt} = 0.$$

Since $\mu_t^{lt} = \mu_s^{lt}$ and $w_{ij} + \bar{\pi}_i^l - \bar{\pi}_j^l \geq 0$,

$$w_{ij} + \bar{\pi}_i^l - \bar{\pi}_j^l = 0, \quad (i,j) \in p.$$

Also, when $s, t \in N(A_l)$, any destination, $d \in N(A_l)$, yields

$$
\begin{aligned}
\pi_s^l - \pi_d^l &= \bar{\pi}_s^l - \bar{\pi}_d^l \\
\pi_t^l - \pi_d^l &= \bar{\pi}_t^l - \bar{\pi}_d^l.
\end{aligned}
$$

Which implies

$$
\begin{aligned}
\sum_{(i,j) \in p} \left(w_{ij} + \pi_i^l - \pi_j^l \right) &= \sum_{(i,j) \in p} w_{ij} - \pi_s^l + \pi_t^l + \pi_d^l - \pi_d^l = \\
&= \sum_{(i,j) \in p} w_{ij} - \bar{\pi}_s^l + \bar{\pi}_t^l + \bar{\pi}_d^l - \bar{\pi}_d^l = \sum_{(i,j) \in p} \left(w_{ij} + \bar{\pi}_i^l - \bar{\pi}_j^l \right) = 0,
\end{aligned}
$$

hence

$$w_{ij} + \pi_i^l - \pi_j^l = 0, \quad (i,j) \in p.$$

Now consider the constraint

$$w_{ij} + \pi_i^l - \pi_j^l + \mu_i^{lt} - \mu_j^{lt} \geq 0$$

for the arcs in path p. Since $w_{ij} + \pi_i^l - \pi_j^l = 0$ when $(i,j) \in p$, we get

$$\mu_s^{lt} \geq \ldots \geq \mu_i^{lt} \geq \mu_j^{lt} \geq \ldots \geq \mu_t^{lt},$$

where the arcs in p are considered from s to t. But $\mu_s^{lt} \geq \mu_t^{lt}$ contradicts

$$\mu_t^{lt} - \mu_s^{lt} \geq 1.$$

Thus, the tight solution, $(w, \bar{\pi}, \bar{\mu})$, is feasible in (ISPR-P) which completes the proof. □

All coefficients in (ISPR-P) are integral and the convex hull of the feasible region is a convex polyhedral cone, therefore a feasible non-integral solution can be scaled into an integral solution. Hence the integrality constraints may be dropped. An alternative, more constructive, way to see this is to consider the LP-relaxation of (ISPR-P) when a suitable lid is attatched to the cone. If there is a feasible solution, then there is a feasible extreme point which corresponds to some basic feasible solution, e.g. obtained by the simplex method. If this solution is multiplied with the determinant of the basis matrix it remains feasible, and by Cramer's rule, it becomes integral. This reasoning and Theorem 3.3 proves the following corollary.

Corollary 3.1
It is possible to determine if (ISPR-F) *has a feasible solution for a family of desired shortest path collections or a family of SP-graphs in polynomial time.*

If there is a unique path from s to t in H_{lt} one may consider to use the constraint in (ISPR-AP) that compare the path length with the terminal node potentials instead of the constraints in (ISPR-P). If the unique path consists of the single arc (s,t) we can do even better since that arc must have reduced cost at least 1. That is,

$$\mathcal{P}_{st}^l = \{(s,t)\} \Rightarrow w_{st}^l + \pi_s^l - \pi_t^l \geq 1. \tag{3.34}$$

A particulary nice case is when all paths outside G_l consist of single arcs for all l. This happens when all SP-graphs are spanning which is considered in Section 3.4.4.

Another size reducing preprocessing step is to check if the constraints associated with some node pair, $(s,t) \in V_l$, is redundant for some l. Suppose that all paths from s to t in G_l only include backward arcs. This implies that the difference between the node potentials, $\pi_s^l - \pi_t^l$, is the nonegative path distance from s to t and

$$\sum_{(i,j)\in p} w_{ij} - \pi_t^l + \pi_s^l \geq 1, \tag{3.35}$$

for a path p outside A_l since all weights are at least 1. Hence, the constraint associated with (s,t) is redundant. It is easy to determine which node pairs in an SP-graph that satisfy the above condition. A special case are the node pairs $(s,t) \in V_l$ where s is a destination in A_l. Their corresponding constratints are always redundant.

We have presented a polynomial model of ISPR for connected non-spanning SP-graphs above. The generality of SP-graphs over ingraphs implies that more combination strategies are allowed. This could yield smaller models for some instances.

However, as pointed out before, it is often more natural to work with ingraphs instead of general SP-graphs. The information that there is a path to the root node may be very useful. We present a formulation for non-spanning ingraphs without the second set of node potentials in the next section.

3.4.3 A Formulation of ISPR with Non-Spanning Ingraphs

The directed graph $G = (N, A)$ and the set of destinations $L \subseteq N$ are given along with a family of ingraphs $\{G_l : l \in L \subseteq N\}$, possibly obtained from the path collection $\mathcal{Q}$. We assume that is the family of ingraphs is internally consistent and consider the feasibility problem (ISPR-P). Since an ingraph, $G_l = (N(A_l), A_l)$, contains a reversely directed arborescence to its destination, l, there is a path from all nodes in $N(A_l)$ to l.

To obtain a formulation without the second set of node potentials introduced in (ISPR-P) the following fact is utilized: there is a node potential where all arcs in a spanning reversely directed arborescence to the destination are tight.

The arcs outside G_l are partitioned into outarcs and external arcs. The set of outarcs from G_l are defined as

$$O_l = \{(i,j) \in A \mid i \in N(A_l),\ j \notin N(A_l)\}, \quad l \in L, \tag{3.36}$$

and the sets of external arcs are defined as

$$I_l = \{(i,j) \in A \mid i \notin N(A_l)\}, \quad l \in L. \tag{3.37}$$

For an outarc, the startnode is in G_l and the endnode is outside G_l, that is, it leaves G_l. An external arc has the startnode outside G_l and the endnode may be in- or outside A_l. That is, it either enters or is completely outside G_l. The definitions in (3.36) and (3.37) implies that A_l, O_l and I_l are disjont and that $A = A_l \cup O_l \cup I_l$ for all $l \in L$.

We say that a node potential, π^l, is tight if for each node, i, the difference $\pi^l_l - \pi^l_i$ equals the shortest path distance from i to l. Clearly, a weight vector w induce a set of tight potentials, $\bar{\pi}^l$ say, which in turn induces a set of tight potentials, $\bar{\mu}^{lt}$ say. The following lemma states that (ISPR-P) is feasible if and only if it has a tight solution.

Lemma 3.2
If w is a feasible solution to (ISPR-F) *and $\bar{\pi}$ and $\bar{\mu}$ are the induced tight potentials, then $(w,\ \bar{\pi},\ \bar{\mu})$ is a feasible solution to* (ISPR-P).

Proof: When $\bar{\pi}$ and $\bar{\mu}$ are feasible and tight they are integral. Hence, the only possible source of infeasibility is a constraint

$$\mu_t^{lt} - \mu_s^{lt} \geq 1 \tag{3.38}$$

for some $(s,t) \in V_l$ and $l \in L$. But, if this nonnegative difference is not at least 1, it is 0. Hence, all reduced costs along some path from s to t outside A_l are 0. This induce a new shortest path and contradicts that w is feasible in (ISPR-F). □

This lemma implies that there exists a feasible solution to (ISPR-P) that is tight. Our model builds upon this observation and the following property which holds when we consider ingraphs and a tight solution.

Lemma 3.3
Let $(w,\ \bar{\pi},\ \bar{\mu})$ be tight. If $w_{ij} + \bar{\pi}_i^l - \bar{\pi}_j^l = 0$, then there is a path $p \in \mathcal{P}_{il}$ from i to l where

$$w_{ij} + \bar{\pi}_{i'}^l - \bar{\pi}_{j'}^l = 0, \quad (i',j') \in p. \tag{3.39}$$

Proof: Assume that $(w,\ \bar{\pi},\ \bar{\mu})$ is tight and $w_{ij} + \bar{\pi}_i^l - \bar{\pi}_j^l = 0$. There is a spanning in-arborescence, F say, to l where all arcs are tight. Since F is spanning, the end node of the arc (i,j) is in F and we know that there is a path from j to l in F where all reduced costs are 0. Hence, the path from i to l via j satisfies (3.39). □

Lemma 3.3 yields a path from i to l where all reduced costs are 0 w.r.t. a tight potential if the reduced cost of arc (i,j) is 0. Hence, there must not be an outarc from A_l where the reduced cost is zero. Therefore it is feasible to add the constraints

$$w_{ij} + \pi_i^l - \pi_j^l \geq 1, \quad (i,j) \in O_l. \tag{3.40}$$

Observe that a path completely outside A_l must contain at least one outarc from G_l. This implies that all paths completely outside G_l have an arc with reduced cost at least 1. Hence, all constraints associated with the μ-potentials are redundant.

This derivation yields the following model, which is often significantly smaller than (ISPR-P).

$$\begin{array}{rcll} w_{ij} + \pi_i^l - \pi_j^l & = & 0 & (i,j) \in A_l,\ l \in L \\ w_{ij} + \pi_i^l - \pi_j^l & \geq & 1 & (i,j) \in O_l,\ l \in L \\ w_{ij} + \pi_i^l - \pi_j^l & \geq & 0 & (i,j) \in I_l,\ l \in L \\ w_{ij} & \geq & 1 & (i,j) \in A. \end{array} \tag{ISPR-NS}$$

Theorem 3.4
There exist a set of administrative weights if and only if (ISPR-NS) *has a feasible solution.*

The structure of (ISPR-NS) is clean and very similar to the model that is obtained for spanning SP-graphs below and in other papers, e.g. [25, 26]. Because of this, and the other benefits induced by ingraphs, model (ISPR-NS) is preferred. A forwarding-model, very similar to (ISPR-NS), is given in [9]. This model is analyzed in Section 4.2 where formulations for partial SP-graphs are considered. For completeness, we also provide the formulation of ISPR when all SP-graphs are spanning below.

3.4.4 A Formulation of ISPR with Spanning SP-graphs

In this section we assume that $\{G_l : l \in L\}$ is a family of spanning SP-graphs, not necessarily ingraphs, for an index set L and consider the feasibility problem (ISPR-P).

First of all, if all SP-graphs are ingraphs, we may use (ISPR-NS). Since there are no external arcs, we get

$$\begin{array}{rcll} w_{ij} + \pi_i^l - \pi_j^l & = & 0 & (i,j) \in A_l,\ l \in L \\ w_{ij} + \pi_i^l - \pi_j^l & \geq & 1 & (i,j) \in A \setminus A_l,\ l \in L \\ w_{ij} & \geq & 1 & (i,j) \in A \end{array} \qquad \text{(ISPR-S)}$$

Now assume that an SP-graph, $G_l = (N(A_l), A_l) = (N, A_l)$, is spanning, but not an ingraph. This implies that any subgraph, H_{lt}, with the paths outside G_l to t becomes an instar. From (3.34) it is clear that all arcs not in A_l must have reduced cost at least 1,

$$w_{ij} + \pi_i^l - \pi_j^l \geq 1, \quad (i,j) \in A \setminus A_l. \qquad (3.41)$$

This also implies that all μ-related constraints vanish. Since all SP-graphs are spanning, (3.41) must hold for all l. This yields the same reduced feasibility model as above, (ISPR-S), for ISPR when all SP-graphs are spanning. This comes as no surprise, since the model (ISPR-S) have been given in several papers before, e.g. [25, 26, 80, 9, 6, 42].

3.5 Algorithms to Transform Path Sets to SP-graphs

From Proposition 3.1 and Theorem 3.3 we know that it is necessary that SP-graphs are internally consistent for model (ISPR-F) to be feasible. Further, when they are internally consistent, model (ISPR-F) has a feasible solution if and only if (ISPR-P) has a feasible solution.

In this final section of the chapter we consider two algoritms that can be used to transform a collection of paths into an equivalent SP-graph, or give a certificate that model (ISPR-F) is infeasible. The first algoritm is used to check if a path collection is complete. Our second algoritm determines if a family of single origin, single destination SP-graphs with the same destination are internally consistent.

The idea behind the algoritm that checks if a collection of paths for an OD-pair is complete is to simply calculate the number of paths induced by the graph that covers all arcs in all paths. Given a topological sorting of the nodes in this graph the number of paths from the origin to each node can be propagated in the topological order. If the number of paths in $\mathcal{Q}_k$ equals the number of paths induced by the graph, then $\mathcal{Q}_k$ is complete.

Algorithm 3.5.1. Given an OD-pair, (o_k, d_k), determine if the collection of desired shortest paths, $\mathcal{Q}_k$ is complete.

1. Calculate $S_k = \bar{C}(\mathcal{Q}_k)$.
2. Determine a topological sorting of the nodes $N(S_k)$, $o_k = i_1, \ldots, i_T = d_k$.

3. Set $p^{in}(i_t) = 1$ for $t = 1$ and $p^{in}(i_t) = 0$ for all other t.

4. For $t = 1, \dots, T$
   - Set $p^{out} = \sum_{j \in \delta_k^-(i_t)} p^{in}(j)$.
   - For each $j \in \delta_k^+(i_t)$, increase $p^{in}(j)$ by p^{out}.

5. If $p^{in}(d_k) = |\mathcal{Q}_k|$ return **complete** else return **not complete**.

■

Steps 2-4 have time complexity $\mathcal{O}(N(S_k) + S_k) = \mathcal{O}(m + n)$ since both the topological sort and the propagation loop consider a node and an arc once. The implementation and complexity of step 1 depends on the representation of the path collection.

The algoritm that determines if a family of SP-graphs with the same destination are internally consistent is of more practical importance. A straightforward way to check this is to apply the valid cycle algoritm in [26] and combine SP-graphs repeatedly, two at a time. This brute force approach is probably rather inefficient in practice. Further, it does not utilize the information that there is a path from any given node to the common destination.

Our idea is to form the union of all SP-graphs in the familiy and then check the outdegrees. If there is a node, i say, where the outdegree in an initial SP-graph is not zero, but less than the outdegree in the union, then there is a (shortest) path from i to the destination that is not in the initial SP-graph. The algoritm below performs this check for the necessary node-SP-graph combinations. The outdegree is $|\delta_k^+(i)|$ in SP-graph k and $|\delta^+(i)|$ in the union of the SP-graphs.

Algorithm 3.5.2. Given the arcs of a family of single origin, single destination SP-graphs, $\{G_k = (N(S_k), S_k) : k \in D_l\}$, with the same destination, l, determine if they are internally consistent.

1. Set $G_l = (N(A_l), A_l)$, where

$$A_l = \bigcup_{k \in D_l} S_k.$$

2. For each node $i \in N(A_l)$ and each k such that $i \in N(S_k)$ do
   - if $|\delta_k^+(i)| < |\delta^+(i)|$ then return **internally inconsistent**.

3. Return **internally consistent**.

■

For each SP-graph, each arc is considered a constant number of times, so the work load is proportional to $\sum_{k \in D_l} A(S_k) \leq |D_l| \cdot |A|$, hence, the time complexity is $\mathcal{O}(mn)$.

If the actual conflict must be found; follow all paths from the node i in step 2 that leave S_k and continue in G_l until they come back to S_k. This is easily accomplished by a depth-first or breadth-first search. The breadth-first approach have the desirable property that it yields the smallest conflict (w.r.t. the number of arcs) for the pair i and k.

In the single path case, step 2 in the algoritm can be simplified. Note that the outdegrees, $|\delta_k^+(i)|$ and $|\delta^+(i)|$, are bounded above by 1, therefore it is sufficient to check the outdegrees for all nodes in $N(A_l)$. If some $|\delta^+(i)| > 1$, then there are two internally inconsistent SP-graphs. This improvement does however not change the total time complexity because of step 1.

In the following chapters we will not consider path based version of ISPR problem. Since the algoritms given in this section can be used to convert path based indata into SP-graphs this is not a severe restriction.

4

Inverse Partial Shortest Path Routing

A very important generalization of the inverse shortest path routing (ISPR) problem is the inverse *partial* shortest path routing (IPSPR) problem. In IPSPR, the definition of an SP-graph is revised to better fit into the shortest path routing design (SPRD) framework considered in Chapter 8 of this thesis. A partial SP-graph is a subgraph of an SP-graph, hence it may be both non-spanning and disconnected. An IPSPR problem involves to decide if a family of partial SP-graphs can be completed to a family of spanning SP-graphs that is realizable in a shortest path routing (SPR) protocol.

An outline is as follows. The required definitions are given and two versions of the IPSPR problem are formally introduced in Section 4.1. Mathematical formulations are given in Section 4.2 and the computational complexity of the IPSPR problems is considered in Section 4.4. Finally, in Section 4.3 the differences between these IPSPR problems and models are discussed and illustrated via some examples.

4.1 Inverse Partial Shortest Path Routing Problems

The motivation behind IPSRP problems primarily comes from the solution procedure for the class of SPRD problems, e.g. the shortest path traffic engineering problem (STEP). When these problems are solved by the constraint generation scheme outlined in Section 2.2.3 of Chapter 2 it is very important to be able to determine if a partial routing pattern is realizable. At an intermediate node of an enumeration tree only a subset of the routing pattern has been specified, namely the subset corresponding to binary valued routing design variables.

If such a partial routing pattern induces a routing conflict, then valid cuts that prohibit *the part of the partial routing pattern that induces the conflict* should be added to the formulation. Including such cuts (locally or globally) improves the formulation and may yield that the enumeration tree can be pruned in advance. A more complicated case occurs when a partial routing pattern that does not directly induce a routing conflict *can not be completed* into a routing pattern *because all possible completions induce a routing conflict*. In this case, the conflict inducing part of the partial routing pattern should (preferably) also be prohibited.

In practice it is extremely important for efficiency reasons to be able to solve IPSPR problems so that violated valid cuts that improve the formulation can be generated. This brief discussion motivates the study of IPSPR problems. In fact, it suggests that IPSPR problems are much more relevant than ISPR problems (which involve handle connected SP-graphs) in the context of solving SPRD problems.

The definition of a partial SP-graph is as follows.

Definition 4.1
A partial SP-graph with destination l is a pair of arc subsets, $(A_l, \bar{A}_l)$.

In this definition, the arc (u, v) is in A_l when it is required that (u, v) is in some shortest path to l and (u, v) is in $\bar{A}_l$ when (u, v) is not allowed to be in any shortest path to l. Therefore, the arcs in A_l are called SP-arcs and the arcs in $\bar{A}_l$ non-SP-arcs.

Let $G = (N, A)$ be a strongly connected directed graph, $L \subseteq N$ a set of destinations and $\mathcal{A}_L$ a family of partial SP-graphs. The first version of IPSPR to be considered is to decide if $\mathcal{A}_L$ is *partially compatible*, that is, to *determine if there is a routing conflict among the SP-arcs and non-SP-arcs in $\mathcal{A}_L$*.

Definition 4.2
Let $\mathcal{A}_L = \left\{(A_l, \bar{A}_l)\right\}_{l \in L}$ be a family of partial SP-graphs for the set of destinations, L, and let w be a set of administrative weights.

- *If there exist a feasible node potential, π^l, for each $l \in L$ such that*
  - *π^l is tight w.r.t. w for all arcs in A_l, and*
  - *π^l is not tight w.r.t. w for any arc in $\bar{A}_l$,*

  then $\mathcal{A}_L$ is partially compatible with w.
- *If $\mathcal{A}_L$ is partially compatible with some set of administrative weights, then it is partially compatible.*

A second version of IPSPR is to decide if a family of partial SP-graphs, $\mathcal{A}_L$, is *realizable*, that is, to *determine if $\mathcal{A}_L$ can be completed to a spanning family of SP-graphs*.

To formally define realizability the mappings $\mathcal{I}_l : \mathbb{Z}_+^A \to 2^A$ are introduced. For a set of administrative weights, w say, $\mathcal{I}_l(w)$ is the set of all arcs that are on some shortest path to destination l w.r.t. the link weights w. Note that the arc set $\mathcal{I}_l(w)$ induces a spanning SP-graph to destination l since G is strongly connected.

Definition 4.3
Let $\mathcal{A}_L = \{(A_l, \bar{A}_l)\}_{l \in L}$ be a family of partial SP-graphs for the set of destinations, L, and let w be a set of administrative weights.

- *If $A_l \subseteq \mathcal{I}_l(w)$ and $\bar{A}_l \cap \mathcal{I}_l(w) = \emptyset$ for $l \in L$, then w is fully compatible with $\mathcal{A}_L$.*
- *If $\mathcal{A}_L$ is fully compatible with some set of administrative weights, then it is realizable.*

These definitions yield the two versions of IPSPR.

1. Determine if a family of partial SP-graphs is partially compatible.
2. Determine if a family of partial SP-graphs is realizable.

The first version of IPSPR has earlier been considered in the literature, e.g. in [9, 14, 80, 6, 42]. It has in fact (incorrectly, as shown below) been claimed in [14] that these problems are equivalent. If a family of partial SP-graphs is not partially compatible, then there is a routing conflict and clearly, it is not possible to complete the partial SP-graphs into spanning ingraphs that are simultaneously realizable. However, partial compatibility, that is, the absence of a routing conflict does not in general imply that it is possible to complete a family of partial SP-graphs. It will be shown below that the second version of IPSRP is actually NP-complete.

Remark 4.1. We may have to clarify what is meant by a routing conflict. If it is not possible to complete a family of partial SP-graphs there is of course a routing conflict in some sense. However, when this can not be deduced solely from the the SP-arcs and non-SP-arcs and Definition 4.3 it will not be called a routing conflict here. In the examples in Section 4.3 the first two examples do not contain a routing conflict in the sense defined here, but the remaining examples do.

Before the mathematical formulations of these problems are given consider the following observations. There are no restrictions at all on a partial SP-graph, $(A_l, \bar{A}_l)$, in Definition 4.1. However, some trivial requirements must be fulfilled if it is to be part of a realizable family of SP-graphs. Four obvious conditions are given in the following lemma.

Lemma 4.1
A partial SP-graph can not be in a realizable family of SP-graphs if some of the following conditions are satisfied:

1. *The sets of SP-arcs and non-SP arcs are not disjoint.*
2. *The set of SP-arcs contains a directed cycle.*
3. *There is an SP-arc that emanates from the destination.*
4. *All arcs that emanate from a node different from the destination are non-SP-arcs.*

In the rest of this chapter it is assumed that all partial SP-graphs satisfy these necessary conditions. In particular, no partial routing patterns that arise from the SPRD framework in Chapter 8 will be infeasible due to one of these conditions. Note that it is easy to efficiently determine if the conditions are satisfied or not, *if necessary*, as a preprocessing step.

4.2 Mathematical Formulations of IPSPR Problems

Mathematical models for the two IPSPR problems are given below. As mentioned above, some authors have considered the version of IPSPR where it is to be decided if a family of partial SP-graphs is partially compatible. A crude model that can be used to determine if a family of partial SP-graphs is realizable is also presented. Finally, an important relaxation of this realizability model is derived.

4.2.1 A Model for Partial Compatibility

A model that can be used to determine if a routing pattern is partially compatible is given in [9]. Actually, the unique shortest path (USP) case where the traffic must not be split at any node is considered; it is straightforward to generalize that model to the ECMP case.

In [9], forwardings are used to describe the shortest paths. A forwarding is a special kind of partial SP-graph, $(A_l, \bar{A}_l)$, adopted to the USP case. Since there is no splitting, the non-SP-arcs must satisfy

$$\bar{A}_l \subseteq \bigcup_{(u,v)\in A_l} \left(\delta^+(u) \setminus A_l\right). \tag{4.1}$$

The non-SP-arcs are often *defined* from (4.1) with equality, but inclusion is more general and appropriate in the implicit enumeration context.

It is valid to define the non-SP-arcs as in (4.1) with equality in the ECMP case under a certain completeness assumption, cf. page 67 in [9]. It has to be assumed that, for all destinations, $l \in L$ and any node $u \in N(A_l)$ that has an emanating arc, there should be no additional arcs that emanate from u but the ones already in A_l. Basically, what is assumed is that where we have decided how to split the traffic (or not to split it), the decision stands, and no additional splitting is allowed to take place at that node.

The model given in [9] is feasible if and only if there is a metric that is partially compatible with a family of forwardings. It is straightforward to adapt that model to a partial compatibility model for a family of partial SP-graphs. This yields the following model to check if the family of partial SP-graphs, $\mathcal{A}_L = \left\{(A_l, \bar{A}_l)\right\}_{l\in L}$, is partially compatible.

$$\begin{array}{rcll}
w_{ij} + \pi_i^l - \pi_j^l & = & 0 & (i,j) \in A_l,\ l \in L \\
w_{ij} + \pi_i^l - \pi_j^l & \geq & 1 & (i,j) \in \bar{A}_l,\ l \in L \\
w_{ij} + \pi_i^l - \pi_j^l & \geq & 0 & (i,j) \in A \setminus \left(A_l \cup \bar{A}_l\right),\ l \in L \\
w_{ij} & \geq & 1 & (i,j) \in A.
\end{array} \tag{IPSPR-PC}$$

Equivalent or similar models can also be found elsewhere, e.g. in [14, 6, 42, 26]. The following theorem is well known.

Theorem 4.1
A family of partial SP-graphs, $\mathcal{A}_L$, is partially compatible if and only if (IPSPR-PC) *is feasible.*

Note how model (IPSPR-PC) connects the IPSPR problem for (non-spanning) ingraphs to (non-connected) partial SP-graphs. The ingraph model in Section 3.4.3, (ISPR-NS), is obtained from (IPSPR-PC) by simply using (4.1), with equality which yields

$$O_l = \bar{A}_l \quad \text{and} \quad I_l = A \setminus \left(A_l \cup \bar{A}_l\right). \tag{4.2}$$

Remark 4.2. Model (IPSPR-PC) and (IPSPR-PR) below are primarily solved to determine if a partial routing pattern contains a routing conflict. The infeasibility certificates are used to generate cuts for an SPRD problem. However, it may often be the case that the weights induce large parts of a desired partial routing pattern. Therefore, some, or all, weight settings can be used to produce *complete and realizable routing patterns that are similar to a currently desired partial routing pattern.* Obviously, the cost of generating the complete routing patterns also has to be taken into account. The routing patterns may be determined by repeatedly solving (dynamic) shortest path problems cf. Section 2.2.2 in Chapter 2.

4.2.2 Models for Realizability

Consider the realizability version of IPSPR. That is, is it possible to *complete a family of partial SP-graphs into spanning ingraphs that are simultaneously realizable in an SPR protocol.* As mentioned above, partial compatibility is a necessary condition for realizability.

Proposition 4.1
If a family of partial SP-graphs, $\mathcal{A}_L$, is not partially compatible then it is not realizable.

A sufficient condition for realizability is obtained from the following observation: the arc set $\mathcal{I}_l(w)$ corresponds exactly to the arcs with reduced cost 0 w.r.t. *a tight node potential.* Hence, it is sufficient and necessary to find a set of weights and corresponding tight node potentials that verify partial compatibility.

Theorem 4.2
A family of partial SP-graphs, $\mathcal{A}_L$, is realizable if and only if there is a metric w and a tight node potential, π, induced by w such that (w, π) is feasible in (IPSPR-PC)*,*

In the spanning case all SP-graphs contain an arborescence which implies that all node potentials are tight. This observation yields the following corollary.

Corollary 4.1
Let $\mathcal{A}_L$ be a family of partial SP-graphs, if all partial SP-graphs in $\mathcal{A}_L$ are spanning ingraphs, then $\mathcal{A}_L$ is realizable if and only if $\mathcal{A}_L$ is partially compatible.

It is straightforward to derive a bilinear integer program to check if a family of partial SP-graphs is realizable as follows. Let the binary variable y^l_{ij} be 1 if the arc (i, j) is in $\mathcal{I}_l(w)$ and 0 otherwise. Force the (in)graph induced by the y^l variables to contain an l-arborescence as a subgraph and all reduced costs in the arborescence to 0. This yields tight node potentials.

It is well known that a graph contains a spanning l-arborescence if all directed l-cuts contain at least one arc. Note that this is exactly the technique used to describe the dominant of the rooted arborescence polytope, cf. Chapter 52 in [87], in particular the corollaries of Theorem 52.3. This is one way to forcing the existence of an l-arborescence among the arcs associated with y^l.

Given a set $S \subset N \setminus \{l\}$, let $K = (S, N \setminus S)$ be the directed l-cut that contains the arcs where $i \in S$ and $j \in N \setminus S$. If the family of all l-cuts is denoted by $\mathcal{K}_l$, the following model is obtained.

$$\begin{array}{rcll}
w_{ij} + \pi_i^l - \pi_j^l + y_{ij}^l & \geq & 1 & (i,j) \in A,\ l \in L \\
\left(w_{ij} + \pi_i^l - \pi_j^l\right) y_{ij}^l & = & 0 & (i,j) \in A,\ l \in L \\
\sum_{(i,j)\in K} y_{ij}^l & \geq & 1 & K \in \mathcal{K}_l,\ l \in L \\
y_{ij}^l & = & 1 & (i,j) \in A_l,\ l \in L \\
y_{ij}^l & = & 0 & (i,j) \in \bar{A}_l,\ l \in L \\
w_{ij} & \geq & 1 & (i,j) \in A \\
y_{ij}^l & \in & \mathbb{B} & (i,j) \in A,\ l \in L.
\end{array} \qquad \text{(IPSPR-R)}$$

Corollary 4.2
A family of partial SP-graphs, $\mathcal{A}_L$, is realizable if and only if (IPSPR-R) *has a feasible solution.*

Remark 4.3. In model (IPSPR-R), the induced arborescences are modelled via l-cuts independently of the SPR reduced cost constraints. However, if these latter constraints are used a smaller model can be obtained. It is actually sufficient to keep only l-cuts that separate a single node at a time. That is, all cuts are of the form $K = (\{i\}, N \setminus \{l, i\})$. This is verified by the conjunction of two simple observations: (1) these cuts force the outdegree of non-root nodes to be at least one, (2) the SPR reduced cost constraints and strictly positive weights prohibit directed cycles. Therefore, the graph induced by the y^l-variables is a spanning ingraph, which naturally contains a spanning arborescence.

Model (IPSPR-R) is given for completeness. It is certainly not suggested that it should be solved in practice since the bilinear constraints may be very difficult to handle, cf. the discussion in Section 2.2.3 in Chapter 2.

To obtain a relaxation of model (IPSPR-R) that is stronger than (IPSPR-PC) the following method can be used. Introduce a set of distance variables, d. The variable d_{st} is supposed to serve as a lower bound on the distance from s to t for each node pair $(s,t) \in N \times N$. This idea yields the following distance based valid inequalities.

Proposition 4.2
The (in)equalities

$$\begin{array}{rcll}
d_{st} & = & \pi_t^t - \pi_s^t & s,t \in N \\
d_{st} & \geq & \pi_t^l - \pi_s^l & s,t \in N,\ l \in L \\
d_{st} & \geq & 1 & s \neq t,\ s,t \in N
\end{array} \qquad (4.3)$$

are valid for (IPSPR-R)

Proof: Let w and π be part of a feasible solution to (IPSPR-R). The distance from node s to node t is then $d_{st} = \pi_t^t - \pi_s^t \geq 1$ since π is tight and $w \geq 1$. Assume that there is a node l such that $d_{st} < \pi_t^l - \pi_s^l$ and consider a shortest path, p, from s to t. The relation (3.5) between shortest paths and node potentials in Section 3.1 yields

$$0 \leq \sum_{(i,j)\in p} \hat{w}_{ij}^l = \sum_{(i,j)\in p} w_{ij} + \pi_s^l - \pi_t^l = d_{st} - \left(\pi_t^l - \pi_s^l\right) < 0 \tag{4.4}$$

which is a contradiction, hence $d_{ij} \geq \pi_j^l - \pi_i^l$. □

If these distance-based valid inequalities are augmented to (IPSPR-PC) the following (stronger) relaxation of (IPSPR-R) is obtained.

$$\begin{array}{rcll}
w_{ij} + \pi_i^l - \pi_j^l & = & 0 & (i,j) \in A_l,\ l \in L \\
w_{ij} + \pi_i^l - \pi_j^l & \geq & 1 & (i,j) \in \bar{A}_l,\ l \in L \\
w_{ij} + \pi_i^l - \pi_j^l & \geq & 0 & (i,j) \in A \setminus \left(A_l \cup \bar{A}_l\right),\ l \in L \\
d_{il} + \pi_i^l - \pi_l^l & = & 0 & i \in N,\ l \in L \\
d_{ij} + \pi_i^l - \pi_j^l & \geq & 0 & i,j \in N,\ l \in L \\
w_{ij} & \geq & 1 & (i,j) \in A \\
d_{ij} & \geq & 1 & i \neq j,\ i,j \in N.
\end{array} \tag{IPSPR-PR}$$

Definition 4.4
A family of partial SP-graphs, $\mathcal{A}_L$, is partially realizable if (IPSPR-PR) *is feasible.*

The relation between the feasible regions of the formulations above is summarized in the following theorem.

Proposition 4.3
Let $\mathcal{A}_{PC}$, $\mathcal{A}_{PR}$ and $\mathcal{A}_R$ denote the collections of partially compatible, partially realizable and realizable families of partial SP-graphs, respectively. Then,

$$\mathcal{A}_{PC} \supset \mathcal{A}_{PR} \supset \mathcal{A}_R. \tag{4.5}$$

Proof: From above it trivially follows that $\mathcal{A}_{PC} \supseteq \mathcal{A}_{PR} \supseteq \mathcal{A}_R$. It is shown by the examples in the next section that the inclusions are strict, cf. Example 4.1-4.5 and in particular Example 4.2 and 4.5. □

A primary role of the new constraints introduced in Proposition 4.2 is to force induced shortest paths to be shortest so that the corresponding reduced costs are 0 and the node potentials tight. However, it is not necessary to add all the inequalities. It is actually a bad idea.

Proposition 4.4
If there are nodes s and t and a destination l such that there is a path from s to t in A_l, then all constraints involving the variable d_{sl} are redundant in (IPSPR-PR).

Proof: Let s, t and l be the nodes and destination, respectively, and p a path in A_l from s to t. Consider a solution, (w, π), to (IPSPR-PR) and sum up the constraints in the path p for destination l, this yields

$$0 = \sum_{(i,j)\in p} \hat{w}^l_{ij} = \sum_{(i,j)\in p} w_{ij} - \left(\pi^l_t - \pi^l_s\right) = d_{st} - \left(\pi^l_t - \pi^l_s\right). \tag{4.6}$$

Since $d_{st} = \pi^l_t - \pi^l_s$ from (4.6) and $d_{tl} = \pi^l_l - \pi^l_t$ by definition, we get

$$d_{sl} = \pi^l_l - \pi^l_s + \pi^l_t - \pi^l_t = d_{st} + d_{tl} \geq \pi^l_t - \pi^l_s + \pi^l_l - \pi^l_t = \pi^l_l - \pi^l_s. \tag{4.7}$$

□

This implies that the inequalities for d_{sl} should not be added if there is an arc that emanates from s in A_l since they do not strengthen the model and also yield (more) degeneracy. This can cause two problems: the model may be harder to solve in practice and additional alternate dual solutions are introduced. Because of the above, the distance constraints from Proposition 4.2 associated with d_{sl} should be included in (IPSPR-PR) when the outdegree of s in A_l is 0.

Note that the new constraints can be interpreted to be associated with new artificial (possibly parallell) arcs. When these new arcs are included the partial SP-graphs are augmented to spanning ingraphs, cf. Section 5.1.2.

Concerning the size of the models, the number of constraints is approximately $2|N|^3$ in the worst case in (IPSPR-PR) which is twice the number of constraints in (IPSPR-PC). Note that for spanning ingraphs, all constraints from Proposition 4.2 are redundant according to Proposition 4.4, hence the size is not affected in this case.

Our conclusion from above is that model (IPSPR-PR) is to be preferred over (IPSPR-PC) since it is stronger without being significantly larger. The partial compatibility and realizability models are considered further in later chapters. There, it will also be seen that the new artificial arcs sometimes provide very useful information. Especially, when valid inequalities and separation algorithms are considered for the SPRD framework outlined in Chapter 2 and further discussed in Chapter 8.

4.3 Differences Between the IPSPR Models

To illustrate the definitions and differences between the models introduced in this chapter several examples are given below. The first two examples are used to prove that there are instances that are partially realizable (hence partially compatible) but not realizable. This proves that $\mathcal{A}_{PR} \neq \mathcal{A}_R$ in Proposition 4.3. In the other examples, several partially compatible instances are given that are not partially realizable. This proves that $\mathcal{A}_{PR} \neq \mathcal{A}_{PC}$ in Proposition 4.3.

First consider the following trivial example where a single partial SP-graph can not be realized.

Example 4.1

Let $(A_l^1, \bar{A}_l^1)$ be a partial SP-graph where $\bar{A}_l^1$ contains all arcs emanating from a node different from the destination. Also consider the partial SP-graph, $(A_l^2, \bar{A}_l^2)$, where $\bar{A}_l^2$ contains all arcs entering the destination node.

Clearly, both the above SP-graphs alone imply that the y^l variables can not form an l-arborescence. These types of conflicts are not discovered by model (IPSPR-PC), nor by (IPSPR-PR), hence $(A_l^1, \bar{A}_l^1)$ and $(A_l^2, \bar{A}_l^2)$ may be part of a partially compatible and partially realizable family of partial SP-graphs despite the fact that they are not realizable.

In practice, the conflicts in this example are of little importance since a master design problem almost always contains constraints that guarantee that it is possible to form some l-arborescence. That is, constraints implying that not all y^l-variables that emanate from a node (different from l) must be zero. Supposing that instances do occur when this is not true; it is easy to check if a partial SP-graph violates the conditions and simply reject the instance without solving (IPSPR-PR).

The next example may be considered to be more relevant in practice since it does not suffer from the drawback above. It is still not realizable, but partially realizable. This is a better way to show that $\mathcal{A}_{PR} \neq \mathcal{A}_R$.

Example 4.2

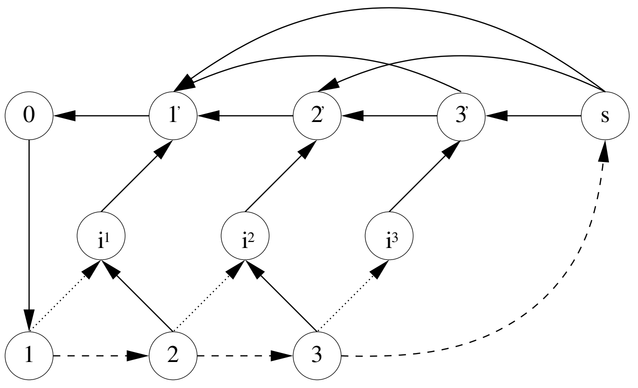


Figure 4.1: *A graph and four non-spanning SP-graphs that can not be realized. Dashed arcs belong to SP-arc set A_0. Dotted arcs belong to the SP-arc sets $A_{1'}$, $A_{2'}$ or $A_{3'}$.*

Consider the graph in Figure 4.1 and the four partial SP-graphs given as follows. The dashed arcs belong to the SP-graph with destination 0. The dotted arc emanating from node 1, 2 and 3, respectively, belongs to the SP-graph with destination 1', 2' and 3', respectively. The USP case is considered (this is not really a restriction, but since no splitting is allowed the non-SP-arcs can be defined by (4.1) with equality). Hence, the SP-arcs and non-SP-arcs are given by

$$A_0 = \{(1,2),\ (2,3),\ (3,s)\}, \quad \bar{A}_0 = \{(1,i^1),\ (2,i^1),\ (2,i^2),\ (3,i^2),\ (3,i^3)\}, \tag{4.8}$$

and

$$\begin{array}{ll} A_{1'} = \{(1,i^1)\}, & \bar{A}_{1'} = \{(1,2)\}, \\ A_{2'} = \{(2,i^2)\}, & \bar{A}_{2'} = \{(2,3),\ (2,i^1)\}, \\ A_{3'} = \{(3,i^3)\}, & \bar{A}_{3'} = \{(3,s),\ (3,i^2)\}. \end{array} \tag{4.9}$$

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 since splitting is not allowed.

Now consider partial realizability (compatibility). It is clear that setting the weights on the arcs entering node i^1, i^2 and i^3 to a large number and the weights on all other arcs to 1 is feasible in (IPSPR-PR) (and (IPSPR-PC)). Note that the node potentials in such a feasible solution are not tight and that the correct, tight node potentials are not feasible w.r.t. the SPR constraints.

It is easy to generalize this example to an example with arbitrarily many nodes that have the same property.

Let us now focus on the difference between partial compatibility and partial realizability. The additional distance constraints in model (IPSPR-PR) serve two important purposes: they force more node potentials to become tight and express the fact that *there will be some shortest path from a node to the destination.* The remaining examples especially illuminate that this yields the following two main features of the partial realizability model:

1. It may find an un-specified shortest path that must be a shortest path in any completion of an SP-graph.
2. It may find a conflict that will arise when incomplete paths to their respective destinations are completed.

Our first example shows that it is sometimes easy to deduce that a path that is not specified to be shortest path must be a shortest by the subpath optimality argument. Such paths can easily be found with a preprocessing step. It is also illustrated that it is crucial to perform this step when the partial compatibility model is used, but that it is unnecessary when the partial realizability model is used. This is further explained by the second example where a path that is not specified to be a shortest path is proven to be a shortest path with a more general reduced cost argument instead of the subpath optimality argument. It is then showed how the reduced cost argument can be used in the partial realizability model for more complicated routing patterns.

Example 4.3

First, it is shown how non-specified shortest paths are induced in the following instance on the complete graph with 4 nodes. The sets of SP-arcs for a family of partial SP-graphs are given in Figure 4.2 and are defined by

$$\begin{array}{lcl} A_1 = \{(2,3)\}, & \text{and} & A_2 = \{(1,2),\ (3,4),\ (4,1)\}, \\ \bar{A}_1 = \emptyset, & & \bar{A}_2 = \emptyset. \end{array} \tag{4.10}$$

The shortest path 3-4-1-2 induced by the SP-graph with destination 2 implies that the subpath 3-4-1 is also a shortest path to destination 1. Therefore, the arcs in this path can be included in A_1 since any completion of A_1 to a spanning ingraph always yields an SP-graph where the arcs $(3,4)$ and $(4,1)$ are SP-arcs. The arcs $(3,4)$ and $(4,1)$ are induced to be in A_1 by A_2.

An alternative formulation of this conclusion is the following reduced cost argument. In any feasible completion of A_1 to a spanning ingraph, the reduced costs of all arcs in the path 3-4-1 will be 0 (w.r.t. the node potential associated with destination 1).

It is straightforward to check for subpaths of shortest paths to enlarge some sets of SP-arcs, this is called the preprocessing step. Let us now see how this affects the partial compatibility and realizability.

Example 4.3: continued

Suppose that an arc is added to a non-SP-arc set in the example. The modified family of partial SP-graphs becomes

$$\begin{array}{lcl} A_1 = \{(2,3)\}, & \text{and} & A_2 = \{(1,2),\ (3,4),\ (4,1)\}, \\ \bar{A}_1 = \{(4,1)\}, & & \bar{A}_2 = \emptyset. \end{array} \tag{4.11}$$

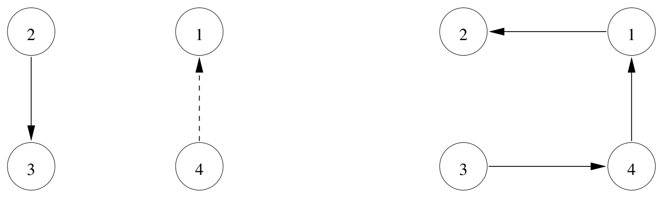


Figure 4.2: An example of a family of partial SP-graphs that is not realizable. The solid arcs on the left and right represent the SP-arcs A_1 and A_2, respectively. The dashed arcs represent the non-SP-arcs, $\bar{A}_1$ and $\bar{A}_2$.

These partial SP-graphs are also given in Figure 4.2.

It was established above that the arcs $(3,4)$ and $(4,1)$ are induced to be in A_1 by A_2. But, since $(4,1) \in \bar{A}_1$, this yields a trivial conflict (which is found by model (IPSPR-PC)). Therefore, there are no compatible administrative weights for this family of partial SP-arcs.

If model (IPSPR-PC) is naively used *without the preprocessing step*, it does however yield a feasible solution. Obviously, no (suggested) set of administrative weights are fully compatible since they all imply that $(4,1)$ is on a shortest path to node 1.

However, if model (IPSPR-PR) is used, a the following subset of constraints that involves the arcs $(3,4),(4,1)$ and the distance variable $(3,1)$ will be given as an infeasibility certificate.

$$\begin{array}{ll} d_{31}+\pi_3^1-\pi_1^1=0, & d_{31}+\pi_3^2-\pi_1^2\geq 0,\\ w_{34}+\pi_3^1-\pi_4^1\geq 0, & w_{34}+\pi_3^2-\pi_4^2=0,\\ w_{41}+\pi_4^1-\pi_1^1\geq 1, & w_{41}+\pi_4^2-\pi_1^2=0. \end{array} \tag{4.12}$$

If the inequalities are summed up and the equalities are subtracted, the aggregated inequality becomes $0\geq 1$.

The conclusion from this example is that there may be additional shortest path information extractable via preprocessing. If this information is used it may verify that an instance is not partially compatible. It is however possible to deduce the same facts with the partial realizability model without the preprocessing.

In fact, it is quite easy to see that the preprocessing implies that model (IPSPR-PC) will find any induced conflict that involves exactly two inconsistent paths. This kind of conflict is called subpath inconsistency and is discussed further in the next chapter. In example 4.5 it will be seen that the preprocessing is insufficient for more complicated induced conflicts. Before that, we will elaborate on the reduced cost argument to deduce when unspecified shortest paths must be shortest paths.

Example 4.4

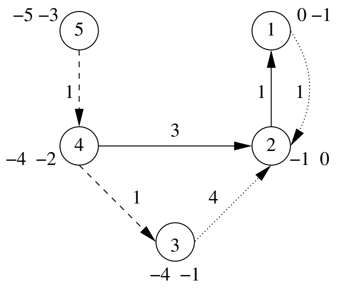


Figure 4.3: *An instance where a family of partial SP-graph is partially compatible but not (partially) realizable. The solid arcs are SP-arcs for destiantion 1 and the dashed arcs for destiantion 2. A solution that verifies partial compatibility is provided; link weights are given next to the relevant arcs and node potentials next to the nodes. The left potential is associated with destination 1 and the right with destination 2.*

The complete graph on 5 nodes and the family of partial SP-graphs with the specified SP-arcs below are considered for the USP case. Since no splitting is allowed, the non-SP-arcs can be defined by (4.1) with equality.

$$\begin{aligned} A_1 &= \{(4,2),\ (2,1)\}, \\ A_2 &= \{(5,4),\ (4,3)\}. \end{aligned} \tag{4.13}$$

A feasible solution to the partial compatibility model is given in Figure 4.3. The interpretation is given in the figure text, weights on arcs not drawn are not relevant and may be set to some large number.

The subset of constraints given below corresponds to the arcs drawn in Figure 4.3. These are the constraints that are not trivially satisfied. It is easily verified that the weights and node potentials from Figure 4.3 satisfies them.

$$\begin{aligned} w_{12} + \pi_1^1 - \pi_2^1 &\geq 0, & w_{12} + \pi_1^2 - \pi_2^2 &\geq 0, \\ w_{21} + \pi_2^1 - \pi_1^1 &= 0, & w_{21} + \pi_2^2 - \pi_1^2 &\geq 0, \\ w_{32} + \pi_3^1 - \pi_2^1 &\geq 0, & w_{32} + \pi_3^2 - \pi_2^2 &\geq 0, \\ w_{42} + \pi_4^1 - \pi_2^1 &= 0, & w_{42} + \pi_4^2 - \pi_2^2 &\geq 1, \\ w_{43} + \pi_4^1 - \pi_3^1 &\geq 1, & w_{43} + \pi_4^2 - \pi_3^2 &= 0, \\ w_{54} + \pi_5^1 - \pi_4^1 &\geq 0, & w_{54} + \pi_5^2 - \pi_4^2 &= 0. \end{aligned} \tag{4.14}$$

However, note that the node potentials are not tight. Hence, the set of weights can not be used to determine if the family of partial SP-graphs is realizable or not. By inspection it is rather easy to see that the family is in fact not realizable.

To deduce this mathematically, include the distance constraints and a conflict arises. Especially, consider the distance constraints associated with the node pair $(3,2)$ and the original constraints associated with the arcs $(4,2)$ and $(4,3)$,

$$\begin{aligned} d_{32} + \pi_3^1 - \pi_2^1 &\geq 0, & d_{32} + \pi_3^2 - \pi_2^2 &= 0, \\ w_{42} + \pi_4^1 - \pi_2^1 &= 0, & w_{42} + \pi_4^2 - \pi_2^2 &\geq 1, \\ w_{43} + \pi_4^1 - \pi_3^1 &\geq 1, & w_{43} + \pi_4^2 - \pi_3^2 &= 0. \end{aligned} \tag{4.15}$$

This set of constraints yields a contradiction in the same manner as in Example 4.3. When the inequalities are summed up and the equalities are subtracted, the aggregated inequality becomes $0 \geq 2$. Hence, the partial realizability model gives a certificate that the family of partial SP-graphs is not realizable, but the partial compatibility model does not.

Example 4.4: continued

Suppose that we had not considered the USP case above. Let A_1 and A_2 be as in (4.13) and set $\bar{A}_1 = \bar{A}_2 = \emptyset$. This implies that the family of SP-graphs is realizable.

The preprocessing step can be used to deduce that $4-2$ is a a shortest path to node 2 since it is a subpath of $4-2-1$, which is a shortest path to node 1. Therefore, the reduced cost of $(4,2)$ is 0 w.r.t. both node potentials for all feasible sets of administrative weights.

Let us deduce this without the subpath argument. Consider the aggregated constraint that caused a conflict above. When $\bar{A}_1 = \bar{A}_2 = \emptyset$, the constraints in (4.15) become

$$\begin{array}{ll} d_{32} + \pi_3^1 - \pi_2^1 \geq 0, & d_{32} + \pi_3^2 - \pi_2^2 = 0, \\ w_{42} + \pi_4^1 - \pi_2^1 = 0, & w_{42} + \pi_4^2 - \pi_2^2 \geq 0, \\ w_{43} + \pi_4^1 - \pi_3^1 \geq 0, & w_{43} + \pi_4^2 - \pi_3^2 = 0. \end{array} \tag{4.16}$$

If the inequalities are summed up and the equalities are subtracted, the aggregated inequality becomes $0 \geq 0$. Since the left hand side of this constraint is 0, independently of $\bar{A}_1$ and $\bar{A}_2$, the right hand side must also be 0. Hence, all inequality constraints are binding and all reduced costs have to be 0, that is all arcs must be shortest path arcs! This is a more general way to deduce that $4-2$ is a shortest path to node 2 since the argument does not rely on $4-2$ being a subpath of some other shortest path.

From above it is clear that whenever it is possible to use preprocessing to find an induced shortest path or a conflict with the partial compatibility model it is possible draw the same conclusion with the partial realizability model. The converse is however not true. This is shown by the following example. Note that it also verifies that $\mathcal{A}_{PC} \neq \mathcal{A}_{PR}$ in Proposition 4.3 even for preproocessed families of partial SP-graphs.

Example 4.5

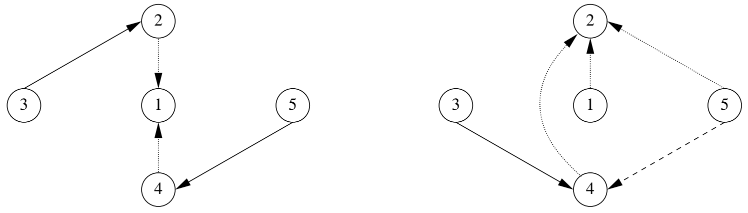


Figure 4.4: *A family of partial SP-graphs that is not realizable. The left partial SP-graph has destination 1 and the right destination 2. The solid arcs are SP-arcs, the dashed arcs are non-SP-arcs and the dotted arcs are distance arcs.*

Consider the complete graph on 5 nodes and the family with the two partial SP-graphs given in Figure 4.4. The SP-arcs and non-SP-arcs are

$$\begin{array}{ll} A_1 = \{(3,2),\ (5,4))\}, & \bar{A}_1 = \emptyset, \\ A_2 = \{(3,4)\}, & \bar{A}_2 = \{(5,4)\}. \end{array} \tag{4.17}$$

Note that the arc $(3,2)$ must be an SP-arc for the partial SP-graph with destination 2. This is deduced by the preprocessing step. However, even when this arc is included it is easy to verify that the family is partially compatible, but as we shall see, not partially realizable. The subset of constraints from model (IPSPR-PR) that yield an infeasibility certificate is given below.

$$\begin{array}{rr} w_{32} + \pi_3^1 - \pi_2^1 = 0, & w_{32} + \pi_3^2 - \pi_2^2 \geq 0, \\ w_{34} + \pi_3^1 - \pi_4^1 \geq 0, & w_{34} + \pi_3^2 - \pi_4^2 = 0, \\ d_{52} + \pi_5^1 - \pi_2^1 \geq 0, & d_{52} + \pi_5^2 - \pi_2^2 = 0, \\ w_{54} + \pi_5^1 - \pi_4^1 = 0, & w_{54} + \pi_5^2 - \pi_4^2 \geq 1. \end{array} \tag{4.18}$$

These constraints yield a contradiction when the inequalities are summed up and the equalities are subtracted, the aggregated inequality becomes $0 \geq 1$.

The information contained in the distance constraints associated with arc $(5, 2)$ is that there will be some shortest path(s) from node 5 to node 2 when the partial SP-graph to destination 2 is completed. No matter how such a path is completeted, it will be involved in an SPR conflict. This additional information was required to deduce that the family is not (partially) realizable.

In this example the subpath preprocessing step is insufficient to make the partial compatibility model infeasible which motivates that (IPSPR-PR) is in general to be preferred over (IPSPR-PC). Note that the conflict in the example is actually not that complicated. It involves only two partial SP-graphs and corresponds to an infeasible structure called a valid cycle. Since these conflicts are easily found one could argue that the conflict in the example could have been found by a slightly more sophisticated preprocessing step. However, this argument fails in general. Given the examples and methods in Chapter 5, it is straightforward to come up with more and more complicated conflicts that can not be found by a (current) preprocessing algorithm. Suppose that an even more sophisticated preprocessing step is suggested, then the process can be repeated. Eventually, a most sophisticated preprocessing is obtained that can be used to solve the original problem without solving model (IPSPR-PR). This justifies our choice of a small and less complicated example.

A final word about preprocessing. A more sophisticated procedure that finds any (potential) conflict that involve two partial SP-graphs is easily derived from Algorithm 5.4.1 on page 100 in Chapter 5. This preprocessing step can be performed efficiently and should definitely be applied since many conflicts and induced shortest paths may be found this way. If no conflict is found, but an induced shortest path is, then this improves the model; unnecessary distance arcs can be removed and degeneracy can be reduced, both these facts suggest that the preprocessing may often reduce the overall computational burden.

The conclusion about model (IPSPR-PR) is that it takes some non-specified shortest paths into account. When this induce an SPR conflict, the conflict is found. If a non-specified shortest path is a shortest path in all possible completions and this does not lead to a conflict, then all reduced cost constraints for the arcs in the path are binding in all feasible solutions to (IPSPR-PR) which implies that the corresponding arcs are on shortest paths.

The above examples illustrate that model (IPSPR-PC) is in some sense inadequate and that (IPSPR-PR) is to be preferred. Even though (IPSPR-PC) solves many instances when the preprocessing step is first applied, the stronger model, (IPSPR-PR), also solves other instances where the conflicts are more complicated. This is and advantage of both practical and theoretical significance. From a theoretical point of view, it is a significant advantage that induced shortest paths and induced infeasibility can be derived from (IPSPR-PR) without the preprocessing step. The fact that this can be deduced directly from the model is very important when valid inequalities are derived for the master problem in Chapter 8.

4.4 Complexity of IPSPR Problems

Let us conclude this chapter with a discussion about the unclosed gap between partial realizability and realizability in practice and in theory. First, the computational complexity is considered.

From Theorem 4.1 and Definition 4.4 it is clear that partial compatibility and partial realizability can be determined in polynomial time by solving linear (feasibility) programs. The complexity of the problem to decide if a family of partial SP-graphs is realizable has been open until now.

To prove that *it is NP-complete to decide if a family of SP-graphs is realizable* or not a polynomial reduction from the exclusive 1 in 3 boolean satisfiability (X3SAT) problem will be described.

Recall that the satisfiability problem (in conjunctive normal form) is to determine if there is an assignment to the boolean variables, $\{x_1, \ldots, x_n\}$, such that each clause, $\{C_1, \ldots, C_m\}$, is true. Here a litteral is either a variable or the negation of a variable and a clause is the disjunction of litterals. It is well known that the satisfiability problem is NP-complete [38]. A closely related problem is the X3SAT problem where each clause is restricted to contain at most three litterals and the boolean assignment is required to make *exactly one litteral in each clause is true.* This problem is also NP-complete.

The following restricted case is used in our proof below.

Definition 4.5
A X3SAT instance, $I = (X, \mathcal{C})$, given by 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 $C = (x_i \vee x_j \vee x_k) \in \mathcal{C}$ it holds that $i < j < k$.*
- *No pair of variables is included in two or more clauses.*

That is, a canonical X3SAT instance only contains sorted clauses where all variables are positive and different and no two clauses share more than one variable. It is in fact no restriction to only consider canonical X3SAT instances.

Proposition 4.5
It is NP-complete to decide if an X3SAT instance in canonical form is satisfiable.

Proof: X3SAT is in principle equivalent to feasibility of set partitioning instances. Since it is NP-complete to determine if a set partitioning instance is feasible even when each row contains at most three ones it suffices to show how to reduce such an instance to the case where no two rows share two variables. Suppose that I is an arbitrary instance where renaming varibles yields the constraints

$$y_1 + y_2 + y_3 = 1 \quad \text{and} \quad y_1 + y_2 + y_4 = 1. \tag{4.19}$$

Clearly, this instance can be solved by solving the smaller and equivalent instance obtained by setting $y_3 = y_4$. Iteratively applying this argument yields that the instance I can be solved by solving an equivalent and smaller instance where no two rows share two variables. That is, by solving an X3SAT instance in canonical form. □

The next operator will be used in our proof below. Intuitively, it yields the modulowise next variable in a clause.

Definition 4.6
If $C = (z_i \vee z_j \vee z_k) \in \mathcal{C}$ is a clause in a canonical 3-SAT instance, $I = (X, \mathcal{C})$, where z_l is a literal associated with the corresponding variable x_l, then the next operator, $n : X \times \mathcal{C} \to X$, is defined by

$$n(x, C) = \begin{cases} x_j & \text{if } x = x_i \\ x_k & \text{if } x = x_j \\ x_i & \text{if } x = x_k. \end{cases} \tag{4.20}$$

Example 4.6

The following X3SAT instance where the set of variables is

$$\{a, b, c, d, e, f, g\} \tag{4.21}$$

and the collection of clauses is

$$\{C_1, C_2, C_3, C_4\} = \{a \vee b \vee c,\ a \vee d \vee e,\ a \vee f \vee g,\ b \vee d \vee f\} \tag{4.22}$$

is in canonical form. The problem is to determine if there is an assignment so that

$$\bigwedge_{i=1}^{4} C_i = (a \vee b \vee c) \wedge (a \vee d \vee e) \wedge (a \vee f \vee g) \wedge (b \vee d \vee f) \tag{4.23}$$

is true.

In this case there are 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 to be in canonical form if the clause $(c \vee e \vee f)$ is added, but not if the clause $(b \vee e \vee f)$ is added since C_4 already contains both b and f.

The next operator takes the following values for clause C_1

$$n(A, C_1) = B, \quad n(B, C_1) = C \quad \text{and} \quad n(C, C_1) = A. \tag{4.24}$$

Let us now describe how to construct a realizability instance from an 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, $\mathcal{A}(I)$, is created as follows. First, it is described how to determine the node set, N, and then, the arc set, A. Finally, an SP-graph is determined for each variable $l \in X$ by partitioning the arcs into SP-arcs, A_l, non-SP-arcs $\bar{A}_l$ and unrestricted arcs U_l is described. The procedure is illustrated in Example 4.7.

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

To determine the set of arcs consider each variable $x \in X$ and add the arcs

$$(S, x^+),\ (S, x^-), \tag{4.25}$$
$$(x^+, x),\ (x^-, x), \tag{4.26}$$
$$(x^+, y),\ (x^-, y),\ (x, y), \tag{4.27}$$
$$(C, x^-) \tag{4.28}$$
$$(C_{ij}, x^+),\ (C_{ik}, x^+),\ (C_{jk}, x^+), \tag{4.29}$$

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

It remains to construct the family of SP-graph. For each variable, $x \in X$, form an SP-graph to the node $l = x$ with SP-arcs, A_l, non-SP-arcs $\bar{A}_l$ and unrestricted arcs U_l determined as follows.

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^-, x) to U_l.
3. Add all arcs emanating from S except (S, x^+) and (S, x^-, x) to $\bar{A}_l$ 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 $C = (x_i \vee_j \vee x_k)$ 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 for $l = x_i$. Also add (C_{ij}, x_i^+) and (x_i^+, x_j) to A_l for $l = x_j$.

6. Add (C_{ik}, x_k^+) and (x_k^+, x_i) to A_l for $l = x_i$. Also add (C_{ik}, x_i^+) and (x_i^+, x_k) to A_l for $l = x_k$.

7. Add (C_{jk}, x_k^+) and (x_k^+, x_j) to A_l for $l = x_j$. Also add (C_{jk}, x_j^+) and (x_j^+, x_k) to A_l for $l = x_k$.

Arcs not put in A_l or $\bar{A}_l$ due to one of the rules above is put in U_l.

Remark 4.4. Note that the SP-arcs for destination $l = x$ induce a tree that spans all nodes in G that associated with a variable or clause that is connected to x via some clause. The starting node S is not contained in any tree and all emanating arcs are either in U_l or $\bar{A}_l$.

Example 4.7

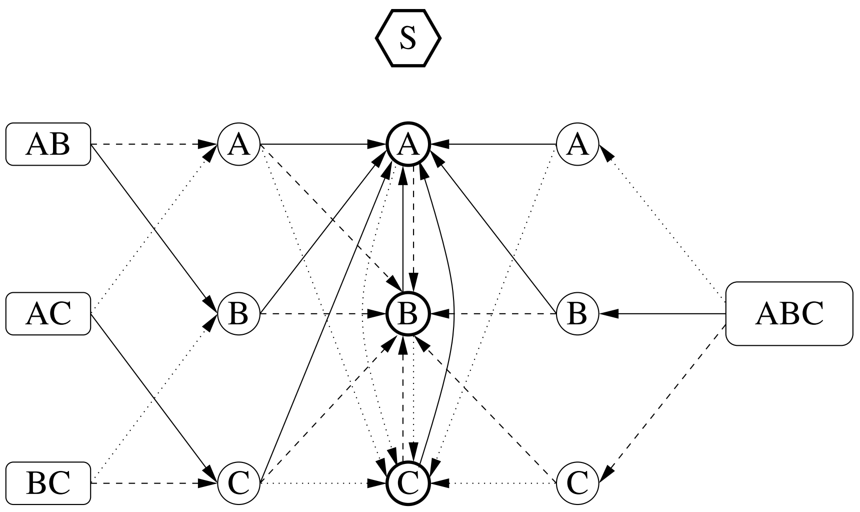


Figure 4.5: *Part of the SP-graphs associated with variables A, B and C in the realizability instance corresponding to an X3SAT instance containing the clause ABC. The solid, dashed and dotted arcs are required to be SP-arcs for destination A, B and C, respectively.*

Consider an X3SAT instance containing the clause $(A \vee B \vee C)$. Using the procedure above to construct the SP-graphs associated with destination nodes A, B and C yields the family of SP-graphs in Figure 4.5.

Breifely, the intuition behind our construction is that the SP-arcs from the starting node are used to determine a feasible assignment. Then, the SP-arcs from the ABC clause node force at least one of the variables in the clause to be true. The auxilliary clause nodes AB, AC and BC are used to guarantee that at most one of the variables is true.

Using a realizability instance created as above it is possible to determine if an X3SAT instance is feasible.

Theorem 4.3
Given a canonical X3SAT instance, $I = (X, \mathcal{C})$, let $G(I) = (N, A)$ and $A_l \cup U_l \cup \bar{A}_l$ be constructed from rules 1-7 above for each variable $x_l \in X$. Denote the induced family of SP-graphs by $\mathcal{A}_L$. Then, the X3SAT instance $I = (X, \mathcal{C})$ is feasible if $\mathcal{A}_L$ is realizable.

To prove this theorem some lemmas are required. Lemma 4.2 and 4.3 do not rely on the particular structure of the realizability instance induced by an X3SAT instance, but rather on the concept of routing conflicts to be thorougly discussed in Chapter 5.

Lemma 4.2
Given an SP-graph family, $\mathcal{A}_L$, and a set of administrative weights, w, that verifies that $\mathcal{A}_L$ is realizable, let $a, b, c \in L$ be three destinations. Consider two (start) nodes, $s_1, s_2 \in N$ and three (end) nodes, $e_3, e_4, e_5 \in N$. If the SP-arc sets are as follows,

$$\begin{array}{lll} \mathcal{I}_l & \supseteq & \{(s_1, e_3)\} \cup \{(s_2, e_5)\}, \textit{ for destination } l = a \\ \mathcal{I}_l & \supseteq & \{(s_1, e_4)\} \cup \{(s_2, e_3)\}, \textit{ for destination } l = b \\ \mathcal{I}_l & \supseteq & \{(s_1, e_5)\} \cup \{(s_2, e_4)\}, \textit{ for destination } l = c \end{array} \tag{4.30}$$

Then, the induced SP-arc sets must also be as follows,

$$\begin{array}{lll} \mathcal{I}_l & \supseteq & \{(s_1, e_5)\} \cup \{(s_2, e_3)\}, \textit{ for destination } l = a \\ \mathcal{I}_l & \supseteq & \{(s_1, e_3)\} \cup \{(s_2, e_4)\}, \textit{ for destination } l = b \\ \mathcal{I}_l & \supseteq & \{(s_1, e_4)\} \cup \{(s_2, e_5)\}, \textit{ for destination } l = c. \end{array} \tag{4.31}$$

Proof: The SP-arc sets are illustrated in Figure 4.6.

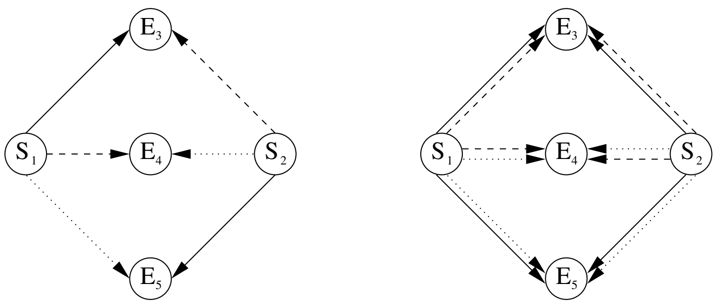


Figure 4.6: *The essential part of the graph in Lemma 4.2. The SP-arc and the shortest subpath to destination l_1, l_2 and l_3, respectively, are drawn with solid, dashed and dotted arrows, respectively. The assumed setting is illusttrated on the left and the induced SP-arcs are drawn on the right. In the right part, the upper arc of the parallell arcs is the original SP-arc and the lower is the induced arc.*

By assumption, there is a set of administrative weights and therefore, no set of SP-arcs must induce a routing conflict. In particular, the subset of constraints of model (IPSPR-R) given below in (4.32) must be feasible. The node potentials associated with a, b and c, are denoted by π^a, π^b and π^c, respectively.

$$\begin{array}{rcll} w_{ij} + \pi_i^l - \pi_j^l + y_{ij}^l & \geq & 1 & (i,j) \in A,\ l \in \{a,b,c\} \\ (w_{ij} + \pi_i^l - \pi_j^l)y_{ij}^l & = & 0 & (i,j) \in A,\ l \in \{a,b,c\} \\ y_{13}^a = y_{14}^b = y_{13}^c & = & 1 & \\ y_{25}^a = y_{23}^b = y_{24}^c & = & 1. & \end{array} \tag{4.32}$$

Simplifying this and keeping only a relevant subset of constraints yields

$$\begin{array}{rclrcl} w_{13} + \pi_1^a - \pi_3^a & = & 0, & w_{13} + \pi_1^b - \pi_3^b + y_{13}^b & \geq & 1, \\ w_{14} + \pi_1^b - \pi_4^b & = & 0, & w_{14} + \pi_1^c - \pi_4^c + y_{14}^c & \geq & 1, \\ w_{15} + \pi_1^c - \pi_5^c & = & 0, & w_{15} + \pi_1^a - \pi_5^a + y_{15}^a & \geq & 1, \\ w_{25} + \pi_2^a - \pi_5^a & = & 0, & w_{25} + \pi_2^c - \pi_5^c + y_{25}^c & \geq & 1, \\ w_{23} + \pi_2^b - \pi_3^b & = & 0, & w_{23} + \pi_2^a - \pi_3^a + y_{23}^a & \geq & 1, \\ w_{24} + \pi_2^c - \pi_4^c & = & 0, & w_{24} + \pi_2^b - \pi_4^b + y_{24}^b & \geq & 1. \end{array} \tag{4.33}$$

Now, add all inequality constraints in (4.33) and remove the equality constraints. This yields the surrogate constraint

$$y_{13}^b + y_{14}^c + y_{15}^a + y_{25}^c + y_{23}^a + y_{24}^b \geq 6, \tag{4.34}$$

which is true if only if

$$y_{13}^b = y_{14}^c = y_{15}^a = y_{25}^c = y_{23}^a = y_{24}^b = 1. \tag{4.35}$$

That is, the corresponding arcs are SP-arcs w.r.t. w and (4.31) follows. □

Lemma 4.3
Given an SP-graph family, $\mathcal{A}_L$, and a set of administrative weights, w, that verifies that $\mathcal{A}_L$ is realizable, let $a, b \in L$ be two destinations. Consider two (start) nodes, $s_1, s_2 \in N$ and two (end) nodes, $e_3, e_4 \in N$. If the SP-arc sets are as follows,

$$\begin{array}{rcl} \mathcal{I}_l & \supseteq & \{(s_1,e_3)\} \cup \{(s_2,e_4)\}, \textit{ for destination } l = a \\ \mathcal{I}_l & \supseteq & \{(s_1,e_4)\} \cup \{(s_2,e_3)\}, \textit{ for destination } l = b \end{array} \tag{4.36}$$

Then, the induced SP-arc sets must also be as follows,

$$\begin{array}{rcl} \mathcal{I}_l & \supseteq & \{(s_1,e_3)\} \cup \{(s_2,e_4)\}, \textit{ for destination } l = b \\ \mathcal{I}_l & \supseteq & \{(s_1,e_4)\} \cup \{(s_2,e_3)\}, \textit{ for destination } l = a. \end{array} \tag{4.37}$$

The proof of the latter lemma is analogous (but simpler) to the proof of Lemma 4.2. A pictorial explanation of Lemma 4.3 similar to the one given for Lemma 4.2 in Figure 4.6 is given in Figure 4.7.

These two lemmas can be used to derive properties of administrative weights for realizability instances obtained as above for X3SAT instances. These properties are summarized in Lemma 4.4 which is the foundation of the proof of Theorem 4.3.

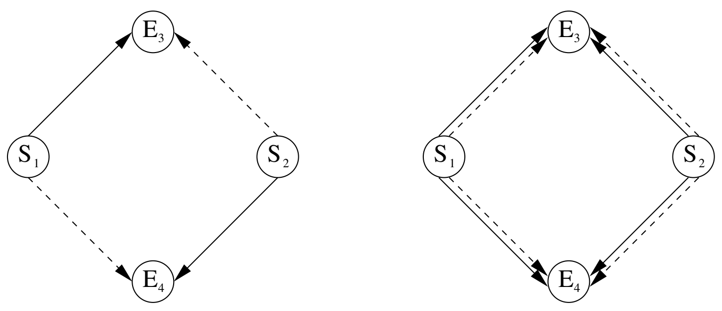


Figure 4.7: *The essential part of the graph in Lemma 4.3. The SP-arc and the shortest subpath to destination l_1 and l_3, are drawn with solid and dashed arrows, respectively. The assumed setting is illusttrated on the left and the induced SP-arcs are drawn on the right. In the right part, the upper arc of the parallell arcs is the original SP-arc and the lower is the induced arc.*

Lemma 4.4
Let $\mathcal{A}_L$ an SP-graph family induced by an X3SAT instance, $I = (X, \mathcal{C})$. Let w be a set of administrative weights that verifies that $\mathcal{A}_L$ is realizable, i.e.,

$$A_l \subseteq \mathcal{I}_l(w) \quad \textit{and} \quad \bar{A}_l \cap \mathcal{I}_l(w) = \emptyset, \quad \textit{for all } l \in L. \tag{4.38}$$

Then, the following properties of $\mathcal{I}_l(w)$ 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 $\mathcal{I}_l(w)$ and non-SP-arcs are not.

1. *For any clause, $(A \vee B \vee C)$ say, at least one of the arcs*

$$(S, A^+),\ (S, B^+) \textit{ and } (S, C^+) \tag{4.39}$$

   is an SP-arc to destination A, B and C respectively.

2. *For any clause, $(A \vee B \vee C)$ say, it holds that. At most one of the arcs*

$$(S, A^+) \textit{ and } (S, B^+), \tag{4.40}$$

   is an SP-arc to destination A and B, respectively, at most one of the arcs

$$(S, A^+) \textit{ and } (S, C^+) \tag{4.41}$$

   is an SP-arc to destination A and C, respectively, and at most one of the arcs

$$(S, B^+) \textit{ and } (S, C^+) \tag{4.42}$$

   is an SP-arc to destination B and C, respectively.

3. *For any clause, $(A \vee B \vee C)$ say, exactly one of the arcs*

$$(S, A^+),\ (S, B^+) \text{ and } (S, C^+) \tag{4.43}$$

is an SP-arc to destination A, B and C respectively.

4. *For any variable, X say, exactly one of the arcs*

$$(S, X^-) \text{ and } (S, X^+) \tag{4.44}$$

is an SP-arc to destination X.

Proof: All properties essentially follows from Lemma 4.2 and 4.3. By construction, given a variable X, it holds that any arc (S, i) emanating from the starting node is a non-SP-arc unless i equals X^+ or X^-. Therefore, at least one and at most two of the arcs

$$(S, X^-) \text{ and } (S, X^+) \tag{4.45}$$

are SP-arcs to destination X. Using this, we prove Property 1-4.

1. Consider a clause, $(A \vee B \vee C)$ 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 is a contradiction.

2. Consider a clause, $(A \vee B \vee C)$ 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 4.3 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 is a contradiction. The cases AC and BC are proved analogously.

3. Consider a clause, $(A \vee B \vee C)$ say. Combining the three constraints in 2 yields that at most on 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 $C = (X \vee Y \vee Z)$. 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.

$\square$

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 4.4 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, $I = (X, \mathcal{C})$, let $G(I) = (N, A)$ and $A_l \cup U_l \cup \bar{A}_l$ be constructed from rules 1-7 above for each variable $x_l \in X$. Denote the induced family of SP-graphs by $\mathcal{A}_L$. Then, $\mathcal{A}_L$ is realizable if the X3SAT instance $I = (X, \mathcal{C})$ is feasible.

Proof: It suffices to find a set of administrative weights, $w(\widetilde{X})$, from a given boolean assignment, $\widetilde{X}$, that verifies the realizability of $\mathcal{A}_L$ in $G(I) = (N, A)$.

The following rules are used to determine w from $\widetilde{X}$.

1. For each variable x, set the weights according to (4.46).

(i,j)	w_{ij} if x is true	w_{ij} if x is false
(S, x^+)	1	1
(S, x^-)	1	1
(x^+, x)	1	5
(x^-, x)	5	1
(x, y)	5	5

(4.46)

   where y is any variable different from x.

2. For a clause, $C = (x \vee y \vee z)$ say, set the weights according to (4.47) and (4.48).

(i,j)	w_{ij} if x is true	w_{ij} if y is true	w_{ij} if z is true
(C_{xy}, x^+)	3	1	3
(C_{xy}, y^+)	1	3	3
(C_{xz}, x^+)	3	3	1
(C_{xz}, z^+)	1	3	3
(C_{yz}, y^+)	3	3	1
(C_{yz}, z^+)	3	1	3
(C, x^-)	1	2	3
(C, y^-)	3	1	2
(C, z^-)	2	3	1

(4.47)

(i,j)	w_{ij}
(x^+, y)	2
(y^+, x)	2
(x^+, z)	2
(z^+, x)	2
(y^+, z)	2
(z^+, y)	2
(x^-, z)	2
(y^-, x)	2
(z^-, y)	2

(4.48)

3. Set the weight of an arc not covered by a rule above to 5.

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, $C = (x \vee y \vee z)$, from x^-, y^- or z^- to x, y or z. However, since no other clasue can contain two of the variables x, y or z this is of no concern. This "independence" property yields that it essentially suffices to consider one clause in isolation.

The weight setting obtained from (4.46), (4.47) and (4.48) is illustrated in Figure 4.8 for the part of the graph involving the clause ABC when A is assigned to true.

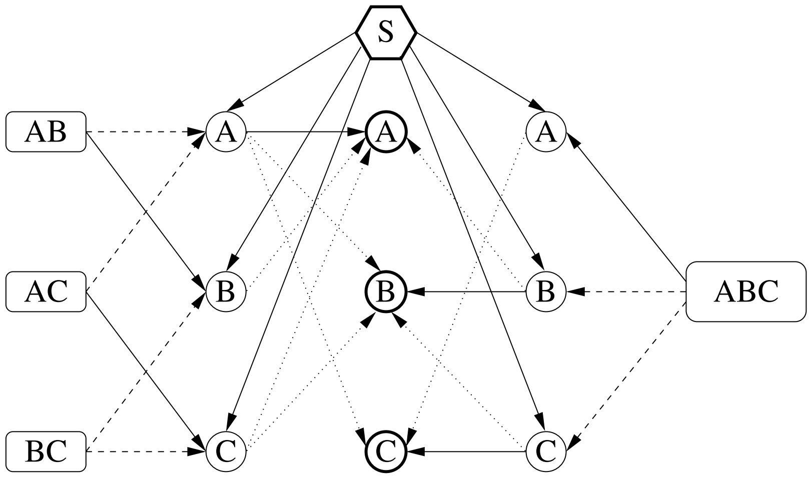


Figure 4.8: *The part of the graph that involves nodes associated with the clause ABC. When A is true, the weight of an arc is 1, 2 or 3, if it is solid, dotted or dashed, respectively. An arc that is not drawn have weight 5.*

From these rules it is straightforward to derive the tight node potential for any destination. Thanks to the independence there are only three cases to consider for each clause, $C = (x \vee y \vee z)$, 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 4.9. 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.

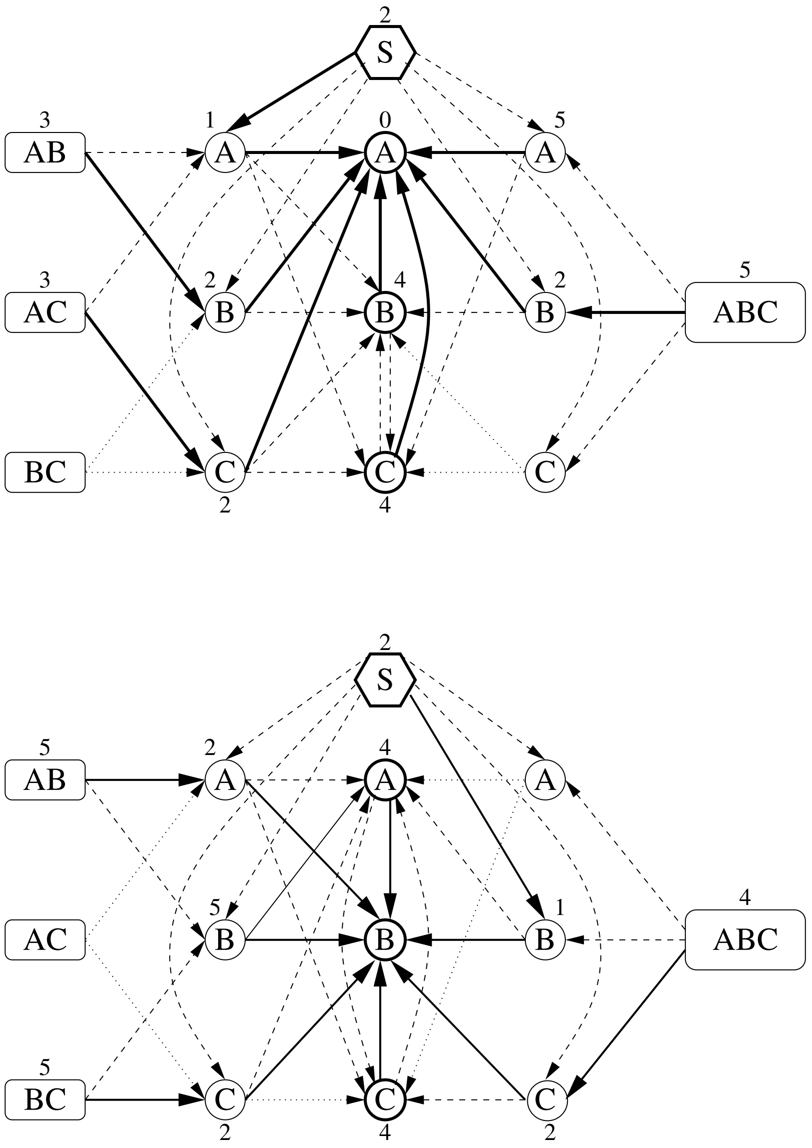


Figure 4.9: *The parts of the shortest path tree to destinations A (top) and B (bottom) that involve nodes associated with the clause ABC which is assumed to be satisfied by A. Solid arcs represent SP-arcs and dotted arcs represent non-SP-arcs. The dotted arcs have not been SP-arcs or non-SP-arcs.*

□

Theorem 4.5
It is NP-complete to decide if a family of SP-graphs is realizable when splitting is allowed.

Proof: It is obvious that a realizability certificate can be verified in polynomial time, therefore the realizability problem is in NP. Further, the reduction from 3-SAT to a realizability instance above is carried out in polynomial time. From Theorems 4.3 and 4.4 it is clear that this instance is realizable if and only if the 3-SAT instance is feasible so the realizability problem must be NP-complete. □

Theorem 4.6
It is NP-complete to decide if a family of SP-graphs is realizable when splitting is not allowed.

Proof: We only need to prove Theorem 4.4 for the single path case.

From Remark 4.4 it is clear that the SP-arcs to a destination induce a tree which yields shortest paths for the affected part of the SP-graph. There is no splitting in the construction in the proof of Theorem 4.4 for these specified shortest paths. There could be multiple shortest paths from a node where an SP-arc have not been specified. However, the binary pertubation technique used in the proof of Proposition 5.4 on page 74 in [9] can be used to construct weights without splitting if necessary. Therefore, a boolean assignment can be used to construct a single shortest path system in Theorem 4.4. That is, if a canonical X3SAT instance, $I = (X, \mathcal{C})$, is feasible, then the family of SP-graphs, $\mathcal{A}(I)$, is realizable as a single path system. □

4.4.1 Discussion

From a theoretical perspective it is important that the complexity issue has been settled. Let us consider the practical consequences of the unclosed gap between partial realizability and realizability.

Recall that the gap is due to disconnected SP-graphs since this implies that it is not guaranteed that the node potentials are tight. In the above examples, the "feasible" solutions obtained from (IPSPR-PC) and (IPSPR-PR) were not necessarily tight, but satisfied the SPR constraints.

Note that even though the realizability problem is NP-complete, there are still properties such that a family is realizable if and only if it is partially realizable, e.g. the simplest being that all SP-graphs are spanning. It is also desirable to develop strong and easily verifiable necessary properties that can be used to prove that a family is infeasible and sufficient properties that yield equivalence between realizability and partial realizability.

In Lemma 4.1 a few trivial necessary conditions for realizability were given. They are all easy to verify in linear time. Another trivial necessary condition is that there must be a reversely spanning l-arborescence among the arcs $A \setminus \bar{A}_l$ for all $l \in L$. Some further requirements can be derived from the examples above.

Deriving more requirements could be important since all necessary conditions can be used to derive valid inequalities for a master design problem. Such inequalities can be used to strengthen the formulation; this avoids that some IPSPR problems are solved and also that nodes further down an enumeration tree are unnecessarily explored.

Concerning sufficient conditions for a family of partial SP-graphs to be realizable if and only if it is partially realizable, we have already mentioned the criteria that all SP-graphs are spanning. The key property is to force the required node potentials to be tight. A related, less restrictive condition is that all partial SP-graphs are connected to their respective destination and the non-SP-arcs are contained in the set of nodes spanned by the respective SP-arcs. This yields that all node potentials are tight w.r.t. the nodes spanned by the SP-arcs and that there are no further restrictions on remaining arcs that can cause infeasibility.

Sufficiency properties can be used to guarantee that no unnecessary, additional computations are performed. Note that each time an IPSPR problem is "solved" with model (IPSPR-PC) or (IPSPR-PR) and the solution obtained is not tight we have not really solved the realizability problem. In an enumeration scheme, this implies that it is not certain if a node should be explored further or not.

Computational experiments have to be used to indicate how to handle the above issues in practice. It is not clear how much time that should be spent to determine if a family of partial SP-graphs is realizable. Perhaps it is better to just settle with less restrictive necessary conditions. In particular, partial realizability. It can actually also be the case that partial realizability and partial compatibility are to restrictive. If so, algorithms tailored to special classes of solutions, e.g. valid cycles, could be considered to efficiently give a certificate that a node should not be explored further witouh adding a valid cut. The final call must not be made until comprehensive computational experiments have been performed.

5 Infeasible Routing Patterns

The inverse partial shortest path routing problem (IPSPR) was introduced in the previous chapter. Mathematical models for the problem of finding administrative weights were given for several settings. Some realizability related concepts for partial SP-graph families were defined along with two necessary conditions. They are, in increasing order of strength: to be partially compatible and to be partially realizable. In this chapter the feasibility issue is studied further. Especially, we characterize routing patterns that yield infeasibility or force some unspecified part to be in a certain way.

As indicated earlier, IPSPR will be used as a subproblem when shortest path routing design (SPRD) problems. These problems are considered further in Chapter 8 of this thesis. When this administrative weight finding subproblem is infeasible, the part of the routing pattern that is not realizable should be prohibited, e.g. by constructing a valid inequality to a master design problem. Hence, it is important to study IPSPR to get a profound understanding of the source of (potential) infeasibility. Some of the models from the previous chapter are examined thoroughly below.

An outline is as follows. The IPSPR problems are formulated and model in Section 5.1 along with their respective Farkas systems. The solutions to the Farkas system of model (IPSPR-PR) yields a characterization of infeasible routing patterns in Section 5.2. The extremal structure of the cone induced by this Farkas system is considered in Section 5.3. A thorough analysis of the infeasible structures that involve at most two SP-Graphs is performed 5.4. Finally, the relation between the classes of structures that has been considered is outlined in Section 5.5.

5.1 Problem Formulation

Several formulations of administrative weight finding problems can be found in previous chapter and in the litterature, e.g. [27, 80, 9, 6, 42]. The Farkas systems of some of these models have been analyzed to find classes of infeasible routing patterns in [25, 23]. Other methods, sometimes of more pragmatic character, can also be used to derive similar structures and subclasses of infeasible patterns, e.g. in [96, 40, 9].

In this chapter a very large class of potentially infeasible structures is derived. Each element (infeasible structure) in this class either implies that a family of partial SP-graphs is not realizable or forces some part of the pattern to be in a certain way. This class is derived from the Farkas systems of the linear models in the previous chapters.

Recall the two necessary conditions for realizability for SP-graph families given in Section 4.1: an SP-graph family must be partially compatible and partially realizable. The Farkas systems of the associated linear models, (IPSPR-PC) and (IPSPR-PR), from Section 4.2 are very useful. They yield sufficient conditions to verify that a family of partial SP-graphs is not realizable or forces the fixation of a unspecified part of some SP-graphs.

Since partial realizability is not always a sufficient condition for realizability, the conditions derived from the Farkas system are not necessarily sufficient. Therefore, the class of structures derived from the Farkas systems can not be exhaustive. However, recall from Remark 4.1 that we defined a routing conflict to be a conflict that can be deduced solely from the the SP-arcs and non-SP-arcs. This implies that the class is exhaustive in the sense that it covers all routing conflicts. Let us now consider the partial compatibility and realizability models and their Farkas systems to derive the class of structures.

5.1.1 The Partial Compatibility Model

Let $G = (N, A)$ be a directed graph and $\mathcal{A}_L$ a family of partial SP-graphs for the set of destinations $L \subseteq N$. Each partial SP-graph, $(A_l, \bar{A}_l) \in \mathcal{A}_L$, 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 partial SP-graph, $(A_l, \bar{A}_l)$, are unrestricted and we do not care if they are in a shortest path to l or not. Denote this set of arcs by $U_l = A \setminus \big(A_l \cup \bar{A}_l\big)$. This yields a partitioning of the arcs, $A = A_l \cup \bar{A}_l \cup U_l$, for all $l \in L$ since it is assumed that $A_l \cap \bar{A}_l = \emptyset$.

The model for partial compatibility is repeated here for convenience and ease of reference. Recall that there is a link weight, w_{ij}, for each arc $(i, j) \in A$ and a node potential, π_i^l, for each node $i \in N$ and each destination $l \in L$. From Section 4.2 in the previous chapter we know that $\mathcal{A}_L$ is partially compatible if and only if the following system has a feasible solution.

$$\begin{array}{rcll} w_{ij} + \pi_i^l - \pi_j^l & = & 0 & (i,j) \in A_l,\ l \in L \\ w_{ij} + \pi_i^l - \pi_j^l & \geq & 1 & (i,j) \in \bar{A}_l,\ l \in L \\ w_{ij} + \pi_i^l - \pi_j^l & \geq & 0 & (i,j) \in U_l,\ l \in L \\ w_{ij} & \geq & 1 & (i,j) \in A. \end{array} \tag{PC}$$

If the variable substitution $w := w - 1$ is applied then Farkas' lemma can be used to conclude that $\mathcal{A}_L$ is not partially compatible if and only if the following system has a feasible solution.

$$\begin{array}{rcll}
\sum_{l\in L}\sum_{(i,j)\in A\setminus\bar{A}_l}\theta^l_{ij} & < & 0 & \\
\sum_{j:(i,j)\in A}\theta^l_{ij} - \sum_{j:(j,i)\in A}\theta^l_{ji} & = & 0 & i\in N,\ l\in L \\
\sum_{l\in L}\theta^l_{ij} & \le & 0 & (i,j)\in A \\
\theta^l_{ij} & \ge & 0 & (i,j)\in A\setminus\bar{A}_l\cup U_l,\ l\in L.
\end{array} \tag{PC-Farkas}$$

Farkas lemma and the results in the previous chapter yields the following. The family of partial SP-graphs $\mathcal{A}_L$ is not realizable if (PC-Farkas) has a feasible solution. If (PC-Farkas) does not have a feasible solution, then $\mathcal{A}_L$ is realizable if the arc sets in each partial SP-graph induce a reversely spanning arborescence rooted at the destination and *may be realizable* otherwise.

Remark 5.1. The Farkas approach used above have earlier been used for models that are very similar to (PC), e.g. in [25, 26, 83]. For instance, in [25] all SP-graphs are spanning, so $U_l = \emptyset$ for all $l \in L$ and $\mathcal{A}_L$ is realizable if and only if (PC) is infeasible.

5.1.2 The Partial Realizability Model

Recall that a family of partial SP-graphs is partially realizable if the model obtained when the distance constraints in Proposition 4.2 are added to (PC) is feasible. Using the same notation as above and adding a distance variable, d_{ij}, for each $i, j \in N$ and the distance inequalities to (PC) yields model (IPSPR-PR) in Section 4.2.

In Proposition 4.4 it was shown that some distance constraints are redundant. To determine if $\mathcal{A}_L$ is partially realizable it is necessary and sufficient to add the inequalities involving d_{sl} only when no arc emanate from s in A_l. Denote the set of required node pairs by $R = \bigcup_{l\in L} R_l$, where

$$R_l = \left\{(i,l) : i \neq l,\ \delta^+(i)\cap A_l = \emptyset\right\}. \tag{5.1}$$

Hence, $\mathcal{A}_L$ is partially realizable if and only if the following system has a feasible solution.

$$\begin{array}{rcll}
w_{ij} + \pi^l_i - \pi^l_j & = & 0 & (i,j)\in A_l,\ l\in L \\
w_{ij} + \pi^l_i - \pi^l_j & \ge & 1 & (i,j)\in \bar{A}_l,\ l\in L \\
w_{ij} + \pi^l_i - \pi^l_j & \ge & 0 & (i,j)\in U_l,\ l\in L \\
d_{il} + \pi^l_i - \pi^l_l & = & 0 & (i,l)\in R_l,\ l\in L \\
d_{ij} + \pi^l_i - \pi^l_j & \ge & 0 & (i,j)\in R_l,\ l\in L \\
w_{ij} & \ge & 1 & (i,j)\in A \\
d_{ij} & \ge & 1 & i\neq j,\ i,j\in N.
\end{array} \tag{PR-D}$$

The stucture of the constraints in this model reveals that there is essentially no difference between the weight variables, w, and the distance variables, d. It is possible to "clean up"

(PR-D) by considering the problem on a multigraph instead of an ordinary, simple graph as follows.

Let $\tilde{G} = (N, \tilde{A})$ be a directed multigraph. That is, directed parallel arcs are allowed in $\tilde{G}$ and $\tilde{A}$ is a multiset where elements may appear several times. The multigraph $\tilde{G}$ is induced by the (ordinary) directed graph $G = (N, A)$ and the SP-graph family $\mathcal{A}_L$ as described below.

The arc set, $\tilde{A}$, of $\tilde{G}$ consists of the ordinary arcs A and a set of destination arcs, $\tilde{D} = \bigcup_{l \in L} \tilde{D}_l$, where $\tilde{D}_l \supseteq R_l$. This implies that A is the ordinary set induced by $\tilde{A}$ without repetitions and that no arc appears in $\tilde{A}$ more than twice. With some abuse of notation we write $\tilde{A} = A \cup \tilde{D}$ and $\tilde{D} = \tilde{A} \setminus A$.

When multigraphs are considered the partial SP-graphs can be generalized by augmenting appropriate sets of destination arcs. Let $\mathcal{A}_L$ be a family of partial SP-graphs for the set of destinations $L \subseteq N$ where each partial SP-graph, $(A_l, \bar{A}_l) \in \mathcal{A}_L$, describes shortest paths to its destination. The corresponding family of generalized SP-graphs, $\widetilde{\mathcal{A}}_L$, for the set of destinations $L \subseteq N$ is defined from $\mathcal{A}_L$ by

$$\widetilde{\mathcal{A}}_L = \left\{ (A_l \cup \tilde{D}_l, \bar{A}_l) \right\}_{l \in L}. \tag{5.2}$$

The arcs in $\tilde{G}$ outside a generalized SP-graph, $\left(A_l \cup \tilde{D}_l \cup \bar{A}_l\right)$, are unrestricted and denoted by $\tilde{U}_l = \tilde{A} \setminus \left(A_l \cup \tilde{D}_l \cup \bar{A}_l\right)$.

To model partial realizability with multigraphs and generalized SP-graphs the variables have to be modified somewhat. There is now a link weight, $\tilde{w}_{ij}$, for each arc $(i,j) \in \tilde{A}$ in the multigraph and a node potential, π_i^l, for each node $i \in N$ and each destination $l \in L$. The new link weight, $\tilde{w}_{ij}$, corresponds to the variable d_{ij} in (PR-D) if it corresponds to a destination arc, $(i,j) \in \tilde{D}$, and to the old link weight variable w_{ij} in (PR-D) when it corresponds to an original arc, $(i,j) \in A$.

When all required destination arcs are included, $\tilde{D} \supseteq R$, the family of generalized SP-graphs, $\widetilde{\mathcal{A}}_L$, is partially realizable if and only if the following system has a feasible solution.

$$\begin{array}{rcll} \tilde{w}_{ij} + \pi_i^l - \pi_j^l & = & 0 & (i,j) \in A_l \cup \tilde{D}_l,\ l \in L \\ \tilde{w}_{ij} + \pi_i^l - \pi_j^l & \geq & 1 & (i,j) \in \bar{A}_l,\ l \in L \\ \tilde{w}_{ij} + \pi_i^l - \pi_j^l & \geq & 0 & (i,j) \in \tilde{U}_l,\ l \in L \\ \tilde{w}_{ij} & \geq & 1 & (i,j) \in \tilde{A}. \end{array} \tag{PR}$$

Again, the variable substitution $\tilde{w} = \tilde{w} - 1$ and Farkas' lemma yield that $\widetilde{\mathcal{A}}_L$ is not partially realizable if and only if the following system has a feasible solution.

$$\begin{array}{rcll} \sum_{l\in L}\sum_{(i,j)\in\tilde{A}\setminus\bar{A}_l}\theta^l_{ij} & < & 0 & \\ \sum_{j:(i,j)\in A}\theta^l_{ij}-\sum_{j:(j,i)\in A}\theta^l_{ji} & = & 0 & i\in N,\ l\in L \\ \sum_{l\in L}\theta^l_{ij} & \leq & 0 & (i,j)\in\tilde{A} \\ \theta^l_{ij} & \geq & 0 & (i,j)\in\tilde{U}_l,\ l\in L. \end{array} \qquad \text{(PR-Farkas)}$$

The conclusion is that the family of generalized SP-graphs, $\widetilde{\mathcal{A}}_L$, is not realizable if (PR-Farkas) has a feasible solution and that $\widetilde{\mathcal{A}}_L$ *may be realizable* otherwise.

Before the set of feasible solutions to (PR-Farkas) is analyzed to characterize forcing and infeasible routing patterns, a couple of remarks are in place.

Remark 5.2. It is often feasible to assume that $A = \bigcup_{l\in L} A_l$ since an arc outside $\bigcup_{l\in L} A_l$ is not on any shortest path and the link weight may be set to some large number (recall that only feasibility is considered). In (PC) it is not in general feasible to assume that $A = \bigcup_{l\in L} A_l$, but it is for example feasible in the spanning case. In (PR) it is always feasible to set the link weights of arcs not on any specified shortest path to some large number since there is an artificial path via a destination arc.

Also consider the above in terms of the Farkas system and duality. In (PR-Farkas), the flow on an original arc outside $\bigcup_{l\in L} A_l$ is nonnegative for all commodities and the aggregated flow must be nonpositive, hence the flow on the arc is zero for each commodity. The interpretation in terms of LP-duality is that redundant constraints (positive reduced cost) yield zero valued dual variables (zero flow).

Remark 5.3. A necessary condition for realizability is that (PR-Farkas) is infeasible. Note that any choice of $\tilde{D}_l$ yield a necessary condition. When $\tilde{D}_l \supseteq R_l$ for all $l \in L$ the condition corresponds to partial realizability and when $\tilde{D} = \emptyset$ it corresponds to partial compatibility. (To obtain (PC) or (PC-Farkas) from (PR) or (PR-Farkas), simply use $\tilde{D} = \emptyset$.)

Remark 5.4. The arc set $A_l \cup \tilde{D}_l$ contains a reversely spanning l-arborescence when $\tilde{D}_l \supseteq R_l$. Further, all arcs in R_l are included in all spanning l-arborescences contained in $A_l \cup R_l$ since including all arcs in R_l is the only way to connect the nodes with outdegree 0 in A_l to node l.

Motivated by the remarks above and Proposition 4.4 we will in the following assume that $\tilde{D}_l$ and $\tilde{D}$ are defined as

$$\tilde{D}_l = \big\{(i,l) : i \neq l,\ \delta^+(i)\cap A_l = \emptyset\big\} \tag{5.3}$$

and

$$\tilde{D} = \bigcup_{l\in L}\tilde{D}_l. \tag{5.4}$$

Proposition 5.1 below is related to Remark 5.2; it states that it is sufficient to consider only the union of the SP-arcs and only one arc in each pair of parallell arcs. Let $\widetilde{\mathcal{A}}_L$ be a family of generalized SP-graphs where $\tilde{D}_l$ is defined as in (5.3) and $\tilde{G}$ the associated multigraph. Then, the underlying ordinary graph, H say, induced by $\tilde{G}$ and the associated family of SP-graphs, $\mathcal{H}_L$ say, on H induced by $\widetilde{\mathcal{A}}_L$ is partially realizable if and only if $\mathcal{H}_L$ is. Formally, let $A^L = \bigcup_{l \in L} A_l$ and $H = (N^H, A^H)$, where

$$N^H = N \quad \text{and} \quad A^H = A^L \cup \tilde{D}, \tag{5.5}$$

and $\mathcal{H}_L = \left(A_l^H, \bar{A}_l^H\right)_{l \in L}$, where

$$A_l^H = A_l \cup D_l \quad \text{and} \quad \bar{A}_l^H = \bar{A}_l \cap A^L. \tag{5.6}$$

Note that $A^H = \bigcup_{l \in L} A_l^H$ and that all A_l^H induce spanning ingraphs. Further, the non-SP-arcs in $\widetilde{\mathcal{A}}_L$ are not necessarily preserved as non-SP-arcs in $\mathcal{H}_L$; it is required that the actual non-SP-arc is also an SP-arc (most likely for some other destination). In summary, H is formed from $\mathcal{H}_L$ by keeping all arcs that is an SP-arc for some destination and then it is augmented with the necessary destination arcs.

Proposition 5.1
Let $\widetilde{\mathcal{A}}_L$ be a family of generalized SP-graphs and $\mathcal{H}_L$ the induced family defined as above by (5.5) *and* (5.6)*, then $\widetilde{\mathcal{A}}_L$ is partially realizable if and only if $\mathcal{H}_L$ is partially realizable.*

Proof: First, let (w, d, π) be a solution that verifies that $\widetilde{\mathcal{A}}_L$ is partially realizable. Then, it is easy to verify that $\mathcal{H}_L$ is partially realizable by defining the weights on the A^H arcs from (w, d, π).

The case where $\widetilde{\mathcal{A}}_L$ is not partially realizable remains. Consider a conflict in $\widetilde{\mathcal{A}}_L$ that verifies this and let θ be a corresponding solution to (PR-Farkas).

Consider an arc $(i,j) \notin A^L$ such that $(i,j) \in \bar{A}_{l'}$. Since $\theta_{ij}^l \geq 0$ for all $l \in L$, the capacity constraint on the arc forces $\theta_{ij}^l = 0$ and the arc (i,j) can not be involved in the conflict. Therefore the conflict only involves arcs in A^L and $\tilde{D}$.

We now show how to form a solution $\bar{\theta}$ to (PR-Farkas) for $\mathcal{H}_L$ from θ. Let (i,j) be an arc in A^H. If $(i,j) \in A^L$ or $(i,j) \in \tilde{D}$ and $(i,j) \notin A^L \cap \tilde{D}$ just set $\bar{\theta}_{ij}^l = \theta_{ij}^l$, note that there is no ambiguity here since (i,j) does not correspond to a parallell arc. Therefore, without loss of generality assume that the conflict in $\widetilde{\mathcal{A}}_L$ involve an arc $(i,j) \in A^L \cap \tilde{D}$. For such an arc, simply combine the flows on the parallell arcs in A_l and $\tilde{D}_l$, that is, let $\bar{\theta}_{ij}^l = \theta_{ij}^l + \tilde{\theta}_{ij}^l$, where θ_{ij}^l is the flow on the A_l arc and $\tilde{\theta}_{ij}^l$ is the flow on the $\tilde{D}_l$ arc. □

This proposition implies that a partial realizability problem can be solved on an ordinary simple graph (that is, without parallell arcs) that only contains the SP-arcs and destination arcs. To obtain such a graph the preprocessing step mentioned in Section 4.3 in the previous chapter can be used. Note that this preprocessing may actually yield a graph that is even smaller than the graph H defined above since it may remove more unnecessary destination arcs. Also note that the weights obtained from a solution on the reduced graph must be used with great care; a weight on an arc corresponding to a destination arc have to be interpreted as a destination, not as an arc weight, cf. Remark 5.5.

It is clear from above that (PR) and (PR-Farkas) 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 with generalized SP-graphs in the following.

Remark 5.5. 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 generalized SP-graphs strongly affects which conclusions one is able to draw from the (in)feasibility of the models. For example, for spanning SP-graphs partial compatibility is equivalent to realizability, but in other cases a partially realizable family of generalized SP-graphs may not be realizable. Another example were just mentioned above in Proposition 5.1. When the underlying ordinary graph is used instead of the multigraph this implies that the weight on an arc is not necessarily the actual weight, but a destination. However, all these properties and requirements on the SP-graphs are not at all revealed by the models.

Let us now consider one of the main issue of this chapter that we have been striving for: to characterize routing patterns that are not realizable or forcing. Due to the complexity of realizability we (have to) settle with partial realizability. This is a fair compromise since (PR) and (PR-Farkas) have a lot of structure and seem to yield fairly strong necessary conditions in practice. To find infeasible and forcing structures the set of feasible solutions to (PR-Farkas) is examined.

5.2 Classes of Infeasible and Forcing Structures

It has been mention before that there is a very close connection between the Farkas system of (PR) and an ordinary multicommodity flow problem. To utilize and also emphasize this, the constraints in (PR-Farkas) 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 (PR-Farkas) is examined, it is clear that a solution is a multicommodity circulation since all node balances are zero. There are however 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 a new model in Chapter 7.

Throughout this chapter, the set of feasible solutions to (PR-Farkas) is denoted by Θ. The closure of Θ, denoted by cl Θ is the set of solutions that are feasible in (PR-Farkas) 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,

$$\Theta^0 = (\text{cl}\,\Theta) \setminus \Theta. \tag{5.7}$$

That is, circulations that satisfy the capacity and commodity specific flow bound constraints but where the objective value is 0. This set is referred to as the set of non-improving solutions.

These observations about the multicommodity structure of (PR-Farkas) should be used when the source of infeasibility is analyzed. Our aim is to explain infeasibility by the presence of some combinatorial structure among the generalized 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 in the latter part of this chapter. However, during the initial description of the different structures we want to focus solely on the definitions of the structures. All examples are postponed until the examination of the comprehensible and important special classes treated in Sections 5.4 below and in Section 6.1 in Chapter 6. Some more complicated examples are given in when the relation between structures is discussed and proved in Chapter 6.

5.2.1 The General Structure

Consider a point in the closure of the set of feasible solutions to (PR-Farkas). That is, $\theta \in \text{cl}\,\Theta$, so the objective value is allowed to be 0. From the discussion above it is clear that θ is a special kind of multicommodity circulation where the individual commodity specific flow bounds are satisfied and the aggregated flow on each arc is at most 0.

Any circulation can be decomposed into flows in undirected cycles. Each such cycle consist 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 labelings makes the cycle directed. 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.

Since the closure of the feasible set to (PR-Farkas) is considered, the origin is feasible. Any other point in the closure must however have some negative flow on some arc to fulfill the objective and capacity constraints. The commodity specific nonnegative flow constraints imply that all arcs with negative flow must belong to their corresponding generalized SP-graph.

This yields that the circulation for a given commodity, l, can be decomposed into a collection of K_l cycles,

$$\left\{ C_k^l \mid C_k^l = F_k^l \cup B_k^l,\ k = 1, \ldots, K_l \right\}, \tag{5.8}$$

with flow, x^l_k. In this decomposition, each cycle, C^l_k, consist of a set of forward arcs, F^l_k, and backward arcs, B^l_k. Further, both these arc sets consist of the arcs in P_{kl} path segments,

$$F^l_k = \bigcup_{p=1}^{P_{kl}} \overrightarrow{P}^l_{kp} \quad \text{and} \quad B^l_k = \bigcup_{p=1}^{P_{kl}} \overleftarrow{P}^l_{kp}. \tag{5.9}$$

In these decompositions, the path segments are alternating. That is,

$$C^l_k = \left(\overrightarrow{P}^l_{k1}\ \overleftarrow{P}^l_{k1} \cdots \overrightarrow{P}^l_{kP_{kl}}\ \overleftarrow{P}^l_{kP_{kl}}\right). \tag{5.10}$$

Because of the commodity specific flow bounds, all backward arcs must belong to the SP-arcs in the generalized SP-graph. That is, $B^l_k \subseteq A_l \cup \tilde{D}_l$ for all k.

When this cycle decomposition is applied to all commodities it is clear that a multicommodity circulation, θ, can be representated by a family of cycles and flows as follows.

$$\mathcal{C} = \left\{\mathcal{C}^l\right\}_{l\in L} \quad \text{and} \quad x = \left\{x^l\right\}_{l\in L}, \tag{5.11}$$

where

$$\mathcal{C}^l = \left\{C^l_k = F^l_k \cup B^l_k \mid B^l_k \subseteq A_l \cup \tilde{D}_l,\ k = 1,\dots,K_l\right\} \quad \text{and} \quad x^l \in \mathbb{R}^{|K_l|}_+. \tag{5.12}$$

This yields that θ is obtained from a family of cycles and the associated flow, $\mathcal{C}$ and x, via the relation

$$\theta^l_{ij} = \sum_{k:(i,j)\in F^l_k} x^l_k - \sum_{k:(i,j)\in B^l_k} x^l_k. \tag{5.13}$$

Remark 5.6. There is not a one-to-one correspondence between the multicommodity circulation θ and the cycle family-flow pair, $\mathcal{C}$ and x. Obviously, $\mathcal{C}$ and x yield a single θ, but other sets of cycles may yield the same θ. It is possible to define a canonical form for the family of cycles such that the relation becomes one-to-one, one such canonical form is considered in Section 5.2.3. However, this canonical form is not of significant practical importance, but merely used as a tool in some proofs.

By construction, any θ obtained from $\mathcal{C}$ and an x via (5.13) is feasible w.r.t. the node balance and the commodity specific flow bounds. If θ is also feasible w.r.t the capacity constraints, then $\mathcal{C}$ is feasible. If θ is also strictly feasible w.r.t. the objective constraint, $\mathcal{C}$ is improving, otherwise it is non-improving. This yields the following definitions.

Definition 5.1
A family of cycles, $\mathcal{C}$, is feasible if there exist an $x = \left\{x^l\right\}_{l\in L} \geq 0$ such that $x \neq 0$ and the θ induced by (5.13) *is feasible, i.e. $\theta \in cl\ \Theta$.*

Definition 5.2
A family of cycles, $\mathcal{C}$, is improving if there exist an $x = \left\{x^l\right\}_{l\in L} \geq 0$ such that the θ induced by (5.13) *is feasible and has a strictly negative objective value, i.e. $\theta \in \Theta$.*

Definition 5.3
A family of cycles, $\mathcal{C}$, is non-improving if for all $x = \{x^l\}_{l\in L} \geq 0$ such that the θ induced by (5.13) *is feasible the objective value is 0, i.e. $\theta \in \Theta^0$.*

Note that the objective is always nonpositive since

$$\sum_{l\in L}\sum_{(i,j)\in\tilde{A}\setminus\bar{A}_l}\theta^l_{ij} \leq \sum_{l\in L}\left(\sum_{(i,j)\in\tilde{A}\setminus\bar{A}_l}\theta^l_{ij} + \sum_{(i,j)\in\bar{A}_l}\theta^l_{ij}\right) = \sum_{(i,j)\in\tilde{A}}\sum_{l\in L}\theta^l_{ij} \leq 0. \quad (5.14)$$

These definitions yield the following theorem.

Theorem 5.1
System (PR-Farkas) *is feasible if and only there is a family of oriented cycles that is feasible and improving.*

Before any specific classes of solutions are considered a distinction between solutions that satisfy the capacity constraint with equality and those who do not is made. Let

$$\Theta^= = \left\{\theta \in \mathrm{cl}\,\Theta \mid \sum_{l\in L}\theta^l_{ij} = 0 \text{ for all } l \in L\right\} \quad (5.15)$$

and

$$\Theta^< = \left\{\theta \in \mathrm{cl}\,\Theta \mid \sum_{l\in L}\theta^l_{ij} < 0 \text{ for some } l \in L\right\}. \quad (5.16)$$

A solution to (PR-Farkas) is saturating if it satisfies all capacity constraints with equality and non-saturating if it satisfies some capacity constraint with strict inequality. Formally, this yields the following two definitions.

Definition 5.4
A solution to (PR-Farkas)*, say θ, is saturating if $\theta \in \Theta^=$.*

Definition 5.5
A solution to (PR-Farkas)*, say θ, is non-saturating if $\theta \in \Theta^<$.*

Remark 5.7. To the best of our knowledge, the class of non-saturating solutions have not been discussed earlier in the litterature. It is mentioned in [24], cf. Theorem 2 on page 512, that it is necessary that not all constraints in a model equivalent to (PC-Farkas) for spanning SP-graphs are satisfied with equality if a solution should be improving, cf. the objective relation in (5.14). However, no concrete example was given there, nor elsewhere. Examples of these solutions with two SP-graphs are given in Section 5.4, cf. the template in Figure 5.8 on page 97. It is straightforward to turn an example of a saturating solution into an example that contains a non-saturating solution, cf. Chapter 7.

The following two propositions are obtained directly from the definitions and (5.14). They are more general then Theorem 2 in [24] since they are adapted to a more general model, (PR-Farkas). 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 5.2
A non-saturating solution is improving.

Proof: If θ is non-saturating, the last inequality in (5.14) is strict, hence it is improving. □

Proposition 5.3
A saturating solution, $\theta \in \Theta^=$, is improving if and only if there is a commodity l and an arc (i,j) such that $(i,j) \in \bar{A}_l$ and $\theta_{ij}^l > 0$.

Proof: Since θ is saturating it is clear that

$$\sum_{l\in L} \sum_{(i,j)\in \tilde{A}\setminus \bar{A}_l} \theta_{ij}^l \leq \sum_{(i,j)\in \tilde{A}} \sum_{l\in L} \theta_{ij}^l = \sum_{l\in L} \left(\sum_{(i,j)\in \tilde{A}\setminus \bar{A}_l} \theta_{ij}^l + \sum_{(i,j)\in \bar{A}_l} \theta_{ij}^l \right) = 0. \quad (5.17)$$

Note that $\theta_{ij}^l \geq 0$ for all $(i,j) \in \bar{A}_l$, hence

$$\sum_{l\in L} \sum_{(i,j)\in \bar{A}_l} \theta_{ij}^l = 0 \iff \theta_{ij}^l = 0 \qquad \text{for all } (i,j) \in \bar{A}_l \text{ and all } l \in L \quad (5.18)$$

and

$$\sum_{l\in L} \sum_{(i,j)\in \bar{A}_l} \theta_{ij}^l > 0 \iff \theta_{ij}^l > 0 \text{ for some } (i,j) \in \bar{A}_l \text{ and some } l \in L. \quad (5.19)$$

Since $\sum_{l\in L} \sum_{(i,j)\in \bar{A}_l} \theta_{ij}^l$ can be seen as the slack in the objective constraint the claim follows. □

Corollary 5.1
All non-improving solutions are saturating, i.e. $\Theta^0 \subseteq \Theta^=$.

In Chapter 7 the relation between saturating and non-saturating solutions will be explained further. In particular it will be proved that there exists a saturating solution (not necessarily improving), whenever there exists a non-saturating solution. Further, it will turn out that a saturating solution obtained from a non-saturating solution is under rather general assumptions actually improving, that is $\Theta \setminus \Theta^= \neq \emptyset \Rightarrow \Theta^= \cap \Theta \neq \emptyset$. This is further explained in Section 7.3.1 in Chapter 7.

Let us now turn to non-improving solutions. A family of cycles, $\mathcal{C}$, that is feasible but not improving also yields important information about the associated partial SP-graphs, $\widetilde{\mathcal{A}}_L$. In any expansion of $\widetilde{\mathcal{A}}_L$ 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 is forcing.

The above conclusion can be derived with a complementary slackness argument from the theory of LP-duality. The flow, θ_{ij}^l, on an arc in a cycle in $\mathcal{C}$ is nonzero, hence the corresponding constraint in the dual, that is, (PR), must be active. But this constraint is $w_{ij} + \pi_i^l - \pi_j^l \geq 0$, therefore the arc must have reduced cost 0 and is on a shortest path to l, so (i, j) must belong to A_l. This gives the following theorem.

Theorem 5.2
Let $\widetilde{\mathcal{A}}_L$ be a family of generalized SP-graphs that is realizable and $\mathcal{C} = \left\{\mathcal{C}^l\right\}_{l \in L}$ non-improving family of cycles, where

$$\mathcal{C}^l = \left\{C_k^l = F_k^l \cup B_k^l \mid B_k^l \subseteq A_l \cup \tilde{D}_l,\ k = 1, \dots, K_l\right\}. \tag{5.20}$$

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

$$B_k^l \subseteq \mathcal{I}_l(w), \quad k = 1, \dots, K_l,\ l \in L. \tag{5.21}$$

Remark 5.8. In the unique shortest path (USP) case, no splitting is allowed and all partial SP-graphs are rooted forests. So if $(i, j) \in A_l$, then all other arcs emanating from i are in $\bar{A}_l$. This implies that almost all solutions are improving. The following kind of solutions with two cycles with destination, j and l, respectively is the only exception

$$\left\{\{C_1^l = F_1^l \cup B_1^l\}, \{C_1^j = F_1^j \cup B_1^j\}\right\}, \tag{5.22}$$

where $B_1^l = F_1^j$ is a path from i to j and $F_1^l = B_1^j$ is the destination arc $(i, j) \in \tilde{D}_j$. This yields as a special case that all solutions are improving in the USP case when all SP-graphs are spanning, cf. Lemma 11 in Paper IV in [27] or [26].

If some restrictions on the families of cycles are considered it is possible to obtain more structure on the classes which often yields that they are 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.

5.2.2 The Binary, Unitary and Simplicial Structures

Let θ be a solution to (PR-Farkas) and let $\mathcal{C}$ and x be an associated family of cycles and cycle flow, respectively. Since the feasible region of (PR-Farkas) is a polyhedral cone, any solution may be scaled such that θ 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. That is, the solution is integral and in a natural sense also minimal. It is now natural to define the binary and unitary solutions.

Definition 5.6
Let θ be a solution to (PR-Farkas) *and $\mathcal{C}$ and $x_k^l \in \mathbb{Z}_+$, an associated family of cycles and cycle flow, respectively. Then θ and $\mathcal{C}$ are called a binary solution and binary cycle family, respectively, if $x_k^l \in \mathbb{B}$ for all k and l.*

Definition 5.7
If θ is a solution to (PR-Farkas)*, then θ is called unitary if $\theta_{ij}^l \in \{-1, 0, 1\}$ for $(i, j) \in \tilde{A}$ and $l \in L$.*

A family of cycles that corresponds to a unitary solution is also called a unitary family of cycles. It is fairly obvious that a unitary solution is also binary.

Proposition 5.4
If θ is a unitary solution to (PR-Farkas)*, then it is a binary solution to* (PR-Farkas).

Proof: Consider a fixed $l \in L$. Since all $\theta^l_{ij} \in \{-1, 0, 1\}$ for $(i, j) \in \tilde{A}$ it is clear that the induced simple directed graph is Eulerian. Any decomposition of this graph into simple cycles yields a feasible collection $\mathcal{C}^l$ in (5.11). 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, $\mathcal{C}$, and a flow solution, x, in the representation in (5.11), it suffices to provide only the cycles where x^l_k is 1, therefore x is superflous and will in the following be omitted for all binary solutions. The binary solution θ will only be represented by

$$\mathcal{C}^l = \left\{ C^l_k = F^l_k \cup B^l_k \mid B^l_k \subseteq A_l \cup \tilde{D}_l,\ k = 1, \ldots, K_l \right\}. \tag{5.23}$$

Given this cycle family, (5.13) may be specialized and θ can be obtained from $\mathcal{C}$ via the relation

$$\theta^l_{ij} = |F^l_{ij}| - |B^l_{ij}|, \tag{5.24}$$

where

$$F^l_{ij} = \{k : (i, j) \in F^l_k\} \quad \text{and} \quad B^l_{ij} = \{k : (i, j) \in B^l_k\}. \tag{5.25}$$

Let us now focus on the unitary case. Since all nonzero flow have the same absolute value it suffices to use a single (not necessarily simple) cycle to represent θ^l for each $l \in L$. This yields the following specialization of (5.11).

$$\mathcal{C} = \left\{ C^l \mid C^l = F^l \cup B^l,\ B^l \subseteq A_l \right\}. \tag{5.26}$$

With this representation, θ is obtained from $\mathcal{C}$ via the relation

$$\theta^l_{ij} = \begin{cases} 1 & \text{if } (i, j) \in F^l \\ -1 & \text{if } (i, j) \in B^l \\ 0 & \text{otherwise.} \end{cases} \tag{5.27}$$

Remark 5.9. Note that (5.27) can be obtained directly from (5.24) since the sets F^k_{ij} and B^k_{ij} 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* (that is, an undirected arc) is at most two. These solutions are called simplicial because of their close connection to simplicial 2-complexes, cf. [5] for an introduction to topology and simplicial complexes.

Consider a unitary solution to (PR-Farkas) represented by

$$\mathcal{C} = \left\{ C^l \mid C^l = F^l \cup B^l,\ B^l \subseteq A_l \right\}. \tag{5.28}$$

and define the destination index sets, F_{ij} and B_{ij}, for each arc $(i,j) \in \tilde{A}$ as

$$F_{ij} = \{l : (i,j) \in F^l\} \quad \text{and} \quad B_{ij} = \{l : (i,j) \in B^l\}. \tag{5.29}$$

This yields the following definition of simplicial solutions.

Definition 5.8
Let θ be a unitary solution to (PR-Farkas) *represented by $\mathcal{C}$ and let F_{ij} and B_{ij} be defined as in* (5.29)*. Then, θ (and $\mathcal{C}$) is simplicial if*

$$|F_{ij} \cup F_{ji}| + |B_{ij} \cup B_{ji}| \leq 2 \quad \textit{for each arc } (i,j) \in \tilde{A}. \tag{5.30}$$

Remark 5.10. The capacity constraint in (PR-Farkas) also yields that $|F_{ij}| \leq |B_{ij}| \leq 1$. Equality in (5.30) is obtained in the saturating case for all arcs in $\mathcal{C}$. It holds that $|F_{ij}| = |B_{ij}| = 1$ and $|F_{ji}| = |B_{ji}| = 0$, for (i,j) or (j,i). In the non-saturating case equality is obtained as above for saturated arcs and also for pairs of non-saturated arcs where $|F_{ij}| = |F_{ji}| = 0$ and $|B_{ij}| = |B_{ji}| = 1$.

Clearly, the relation in (5.27) is sufficient also for simplicial solutions. Further, the interpretation of (PR-Farkas) in θ becomes

$$\sum_{l \in L} \left(|\theta^l_{ij}| + |\theta^l_{ji}| \right) \leq 2 \quad \text{for each } i < j,\ (i,j) \in N \times N. \tag{5.31}$$

A very important subclass of the simplicial solutions is the class of solutions that involve at most two destinations. The size restriction makes this a very comprehensible class. This subclass is considered in [25, 26] where the associated infeasible structure is called a valid cycle. A further restriction of the valid cycles yields an even smaller subclass arising from subpath inconsistency that has been considered by several authors, for instance in [6, 9, 40, 96], cf. the relation between generalized saturating valid cycles and other infeasible structures presented in the litterature on page 95. We will examine these solutions further in Section 5.4. The simplicial solutions are also analyzed more thoroughly in Section 6.1 in Chapter 6.

5.2.3 A Canonical Circuit Decomposition

It was mentioned in Remark 5.6 that there is not a one-to-one correspondence between multicommodity circulations cycle family-flow pairs. In practice, this is seldom an issue because it is often easy to choose a "natural" decomposition. However, it is from a theoretical perspective a bit of a flaw and it may be an obstacle in some proofs. To overcome this, a canonical form for solutions to (PR-Farkas) and cycle family-flow pairs is defined here.

Definition 5.9
A solution $\theta \in cl\,\Theta$ is canonical if it is integral and minimal. That is, there exist no $0 < t < 1$ such that $t\theta$ is integral and

$$\max_{ijl} |t\theta^l_{ij}| \leq \max_{ijl} |\theta^l_{ij}|. \tag{5.32}$$

Given a canonical solution, $\theta \in \text{cl}\,\Theta$, it is possible to define a canonical circuit decomposition from a fundamental cycle basis. The procedure is outlined here and a thorough treatment of fundamental cycle bases is given in Chapter 7. The foundation of fundamental cycle bases and our canonical form is to choose a spanning tree and then use the uniquely defined circuits induced by the arcs not in the tree. The procedure is as follows.

Let θ^l be the part of a canonical solution $\theta \in \text{cl}\,\Theta$ that corresponds to the variables associated with destination l. Form the tree, T_l, by first including all arcs in $\tilde{D}_l \cup A_l$. As long as T_l is not a tree, iteratively apply the following: select the highest numbered node with outdegree at least 2 and then remove the emanating arc with the highest numbered end node. When this is complete, T_l is a spanning intree to node l. Given T_l and θ^l, a circuit family, $\mathcal{C}^l$ and a flow vector x^l is uniquely determined as follows. If $\theta^l_{ij} > 0$ and $(i,j) \notin T_l$, add the unique circuit induced by (i,j) to $\mathcal{C}^l$ and set the corresponding position in x^l to θ^l_{ij}. If $\theta^l_{ij} < 0$ and $(i,j) \notin T_l$, then add the reverse of unique circuit induced by (i,j) to $\mathcal{C}^l$ and set the corresponding position in x^l to $-\theta^l_{ij}$. When this procedure is applied to all destination a a canonical circuit decomposition of a canonical solution is obtained.

We say that a cycle family flow pair is canonical if it is obtained as above. This clearly yields a one-to-one correspondence between canonical solution and canonical cycle family flow pairs.

An advantage with the above construction is its simplicity. There is however also a significant drawback with the canonical form, namely, arcs that carries no flow frequently appears in circuits. It is possible to overcome this by defining a circuit combining procedure, but we omit this discussion here. In practice, the canonical form is not so important since one often starts with a cycle family flow pair and then calculate the induced solution θ. When a cycle family flow pair is sought from a θ, it is usually only the support that is of interest. The reason for the above construction is that canonicity must be used in some proofs later on.

Let us now briefly discuss the extremal structure of the cones related to (PR-Farkas).

5.3 Extreme Rays and Generators

Recall that the set of feasible solutions to (PR-Farkas) is denoted by Θ. It is often not necessary to consider all solutions in the closure of Θ. In general, it suffices to consider the (canonical) extremal solutions to describe cl Θ. However, there may actually be some undesirable properties of the extreme rays of cl Θ when the cone is not pointed since a projection onto the orthogonal complement of the lineality space has to be carried out, cf. Section 5.3.1 and Equation (5.37). This may in a sense destroy the structure of an extremal solution, cf. Example 5.1.

The solutions to (PR-Farkas) will be used to form valid inequalities that prohibit infeasible routing patterns. Because of this, a closely related concept that we call a generator is defined, where the projection is omitted, to keep "extremal" solutions small. It will be sufficient to only consider generators. In fact, it is sufficent to only consider a subset of the generators, namely the the ones that are irreducible. These solutions esentially correspond to the generators that are inclusion-wise minimal.

All these concepts are formally introduced below; to describe generators and irreducible solutions to cl Θ, some definitions and results about the representation of polyhedral cones are required.

5.3.1 Representation of Polyhedral Cones

The treatment of this subject here is very brief. The reader is referred to the litterature on polyhedral theory for a more thorough presentations of this subject, e.g. the classical reference books [85] and [86] or Section 1 in the paper [78].

Let $C \subseteq \mathbb{R}^n$ be a polyhedral cone described as the intersection of a finite set of halfspaces

$$C = \{x \in \mathbb{R}^n \mid Ax \geq 0\} \tag{5.33}$$

with lineality space

$$L = \text{lin.hull}\, C = -C \cap C = \{x \in \mathbb{R}^n \mid Ax = 0\}. \tag{5.34}$$

If C is pointed, that is $L = \{0\}$, then C has a unique minimal representation (up to multiplication by positive scalars) as

$$C = \{0\} + \text{cone}\{y^{(1)}, \ldots, y^{(t)}\}, \tag{5.35}$$

where $y^{(1)}, \ldots, y^{(t)}$ are the exreme rays of C. When C is not pointed, the pointed cone $C_0 = C \cap L^{\perp}$ may be used to decompose C into the orthogonal sum

$$C = L + C_0. \tag{5.36}$$

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 from [28] (in german, translated to english in [78]) gives a characterization of the extreme rays of $C \cap L^{\perp}$.

Theorem 5.3 ([78], [28])
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 $C \cap L^{\perp}$ 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, λ say, in C, the corresponding ray, λ^1 say, in C_0 is obtained via a projection. If B is a basis of the lineality space L, then

$$\lambda^1 = \big(I - B'(BB')^{-1}B\big)\,\lambda. \tag{5.37}$$

This implies that if λ induces $n - d - 1$ linearly independent rows a^i of A such that $a^i\lambda = 0$, then λ^1 is an extreme ray of C, cf. [78].

Let us now return to the examination of model (PR-Farkas) and in particular the set of solutions, Θ.

5.3.2 Extreme Rays and Generators of the Closure of Θ

Clearly, cl Θ is a polyhedral cone since (PR-Farkas) is an inequality description with right hand side 0. Let Θ^L denote the lineality space of cl Θ. It is straightforward to verify that $\Theta^L \subseteq \Theta^0$, equality holds in some special cases, e.g. when the SP-graphs are complete. Now define

$$\Theta^\perp = (\mathrm{cl}\,\Theta) \cap \left(\Theta^L\right)^\perp, \tag{5.38}$$

which yields the orthogonal decomposition

$$\mathrm{cl}\,\Theta = \Theta^L + \Theta^\perp. \tag{5.39}$$

That is, the cone is (orthogonally) decomposed into a non-improving affine subspace and an improving cone.

The following shows that the ordinary extreme rays of cl Θ may be inadequate in the SPRD context. In that context, solutions are used to create valid inequalities that prohibit parts of infeasible routing patterns. Intuitively, solutions whose support is of small cardinality is to be preferred, since this often yield stronger valid inequalities.

Example 5.1

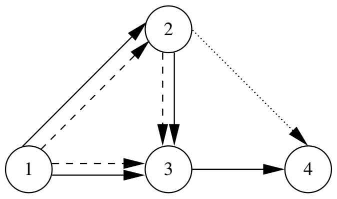


Figure 5.1: *An instance where it is more natural to represent cl Θ by generators that are not extreme rays of cl Θ. The solid, dashed and dotted arcs describe the SP-arcs for the SP-graph with destination 4, 3 and 1 respectively.*

Consider the generalized SP-graphs in Figure 5.1. It is easily verified that the lineality space, Θ^L, is generated by the vector $\bar{\theta}$ with nonzero components

$$\begin{array}{rcr} \bar{\theta}^3_{12} = \bar{\theta}^3_{23} = \bar{\theta}^4_{13} & = & 1, \\ \bar{\theta}^4_{12} = \bar{\theta}^4_{23} = \bar{\theta}^3_{13} & = & -1. \end{array} \tag{5.40}$$

To span the cone cl Θ, another solution, not in the lineality space, is required. One such solution is $\tilde{\theta}$ given by the nonzero components

$$\begin{array}{rcr} \tilde{\theta}^1_{23} = \tilde{\theta}^4_{24} = \tilde{\theta}^1_{34} & = & 1, \\ \tilde{\theta}^4_{23} = \tilde{\theta}^1_{24} = \tilde{\theta}^4_{34} & = & -1. \end{array} \tag{5.41}$$

It is straightforward to verify that the solutions in cl Θ are generated by nonnegative combinations of $\pm\bar{\theta}$ and $\tilde{\theta}$. An alternative to using $\tilde{\theta}$ as a generator is to use

$$\tilde{\theta} + \lambda\bar{\theta} \tag{5.42}$$

for some $\lambda \in \mathbb{R}$. The two "natural" choices for λ here are 0 and -1 since then all nonzero components in $\tilde{\theta} + \lambda\bar{\theta}$ have absolute value 1. These two generator candidates are depicted in Figure 5.2.

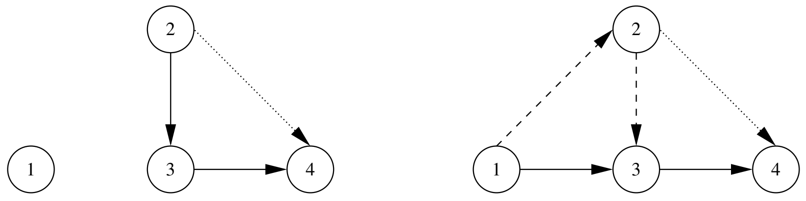


Figure 5.2: *The two (natural) candidates of the additional generator of cl* Θ*. The solution on the left corresponds to* $\lambda = 0$ *in* (5.42) *and the one on the right to* $\lambda = -1$.

Out of these candidates, the one on the left that corresponds to $\lambda = 0$ has the additional advantage that it minimizes the support. This may be of significant importance in the design context, cf. Chapter 8. However, it is easy to show that the valid inequalities induced by the right conflict are not dominated by the valid inequalities induced by the left conflict in this case.

Now consider the orthogonal decomposition of cl Θ into Θ^L and $\Theta^\perp$. It is clear from above that $\pm\bar{\theta}$ generates Θ^L. However, none of the "natural" solutions, $\tilde{\theta}$ or $\tilde{\theta} - \bar{\theta}$, are in $\Theta^\perp$. To obtain the remaining generator, that is, the extreme ray of $\Theta^\perp$, a projection onto $\Theta^\perp$ can be used. This yields $\lambda = 1/6$ and that the orthogonal decomposition becomes

$$\text{cl}\,\Theta = \left\{ \theta \mid \theta = a\bar{\theta} + b\left(\tilde{\theta} + \frac{1}{6}\bar{\theta}\right), \text{ for some } a \in \mathbb{R} \text{ and } b \in \mathbb{R}_+ \right\}. \tag{5.43}$$

Since the support of the solutions in Figure 5.2 are strictly contained in the support of this extreme ray it is easy to show that the induced valid inequalities are dominated. When the design context is considered, this example clearly shows that other decompositions of cl Θ than the orthogonal decomposition with extreme rays in (5.39) should be used when the lineality space is non-empty.

Motivated by Example 5.1 we will not use the ordinary extreme rays of cl Θ. Instead, the closely related concept of a generator of cl Θ is defined as follows.

Definition 5.10
Let θ *be a solution in cl* $\Theta = \Theta^L + \Theta^\perp$ *and* $\theta^{(1)}, \ldots, \theta^{(t)}$ *the extreme rays of* $\Theta^\perp$*. Then* θ *is a generator of cl* Θ *if*

$$\theta = \theta^L + s\theta^{(i)}, \tag{5.44}$$

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

Clearly, any generator generates a whole (equivalence) class of generators. In the following, no distinction between different representants for the class is made, but we do emphasize that a solution of minimal support is to be preferred whenever it is not hard to compute.

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

Proposition 5.5
Let B be a basis of Θ^L then θ is generator of cl Θ if and only if

$$\theta^p = \big(I - B'(BB')^{-1}B\big)\,\theta \tag{5.45}$$

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 can not 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 5.3.

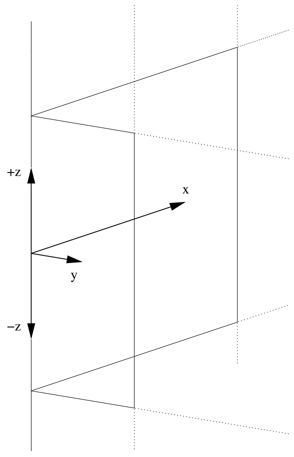


Figure 5.3: *Illustration of our definition of a generator versus the definition of an extreme ray. The cone C under consideration is generated by positive linear combinations of the vectors x, y and $\pm z$. The lineality space is spanned by the vectors $+z$ and $-z$ and $C^\perp$ is spanned by x and y. This yields that x and y are the "ordinary" extreme rays. However, with our definition any solution in one of the two (hyper)planes sketched is a generator, e.g. $x + z$ or y*

The benefit with using generators is that it allows us to essentially ignore the lineality space and the projection in (5.37). 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 definitons coincide. However, since generators are used, we got our back covered, so to speak and do not have to worry about cases such as Example 5.1.

5.3.3 Irreducible Solutions of the Closure of Θ

Clearly, any generator (or extreme ray) corresponds to a routing conflict. What is not that obvious is that such a conflict is not necessarily minimal (w.r.t. inclusion) even when the representant of the generator class is of minimal support. Intuitively, a conflict is irreducible if the involved SP-arcs can not be used to form a smaller conflict. That is, a conflict that involves a subset of these SP-arcs. Consider the following example.

Example 5.2

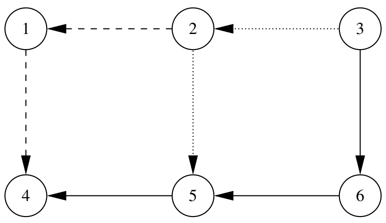


Figure 5.4: *A (reducible) conflict that corresponds to a generator but also contains a smaller conflict. The SP-arcs of three generalized SP-graphs are induced the by solid, dashed and dotted arcs, respectively. Together, these three SP-graphs induce a conflict. However, the SP-graphs corresponding to the solid and dotted arcs alone induce a smaller conflict.*

Consider the solution induced in Figure 5.4. The generalized 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 θ with the following nonzero components is a generator

$$\begin{array}{rcr} \theta'_{14} = \theta'_{21} = \theta'_{32} = \theta''_{25} = \theta''_{54} = \theta'''_{36} = \theta'''_{65} & = & 1 \\ \theta'_{36} = \theta'_{54} = \theta'_{65} = \theta''_{14} = \theta''_{21} = \theta'''_{25} = \theta'''_{32} & = & -1. \end{array} \tag{5.46}$$

However, the following solution, $\bar{\theta}$, with the following nonzero components is another, smaller generator

$$\begin{array}{rcr} \bar{\theta}'_{25} = \bar{\theta}'_{32} = \bar{\theta}'''_{36} = \bar{\theta}'''_{65} & = & 1 \\ \bar{\theta}'_{36} = \bar{\theta}'_{65} = \bar{\theta}'''_{25} = \bar{\theta}'''_{32} & = & -1. \end{array} \tag{5.47}$$

Note that the support of θ is not contained in the support of $\bar{\theta}$, or vice versa. However, the support of the negative components in $\bar{\theta}$ is contained in the support of the negative components in θ. We say that θ is reducible.

Example 5.2 shows that the subset of generators that are minimal, or irreducible, is of particular importance. To describe the set of complete and feasible routing patterns it is sufficient to forbid the subpatterns arising from irreducible generators. In terms of the conflict hypergraph, the "irreducible conflicts essentially correspond to its circuits", cf. Section 8.2.4 in Chapter 8.

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

$$\operatorname{supp}([\theta]^-) = \{((i,j),l) \in A \times L \mid \theta_{ij}^l < 0\} \tag{5.48}$$

denote the set of indices for the negative valued variables in θ.

Definition 5.11
Let θ be a solution in cl Θ. Then θ is an irreducible solution of cl Θ if there does not exist another solution $\bar{\theta}$ such that

$$\operatorname{supp}([\bar{\theta}]^-) \subset \operatorname{supp}([\theta]^-). \tag{5.49}$$

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

Proposition 5.6
The set of irreducible solutions is a subset of the set of generators.

Example 5.2: continued

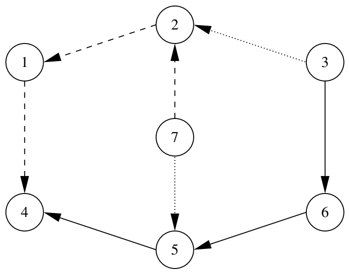


Figure 5.5: *An irreducible conflict obtained by a minor adjustment of the conflict in Figure 5.4 in Example 5.2.*

Again, consider the conflict in Figure 5.4 in Example 5.2. If the arc $(2, 5)$ is subdiveded as in Figure 5.5, the conflict now becomes irreducible. This subdivision procedure is in general applicable to construct an irreducible conflict from a reducible generator, cf. Section 6.1 and in particular Example 6.4 in Chapter 6.

Irreducible solutions are considered in [9] for *path sets in the USP case.* Analyzing the cone is easier when there is no splitting since the lineality space becomes $\{0\}$, so cl $\Theta = \Theta$ and all generators are improving extreme rays.

5.4 Structures Involving at Most Two SP-Graphs

The case with at most two SP-graphs has been considered earlier in the litterature for the compatibility version of IPSPR, that is, model (PC), e.g. in [25, 26, 96, 40, 9]. This restricted case seems to be *much easier* to solve and analyze. In fact, a compact and coherent description of all generators to (PR-Farkas) that involves at most two SP-graphs can be given. It is also possible to develop very efficient (linear time) graph search algorithms to find these solutions. First, consider this introductory example.

Example 5.3

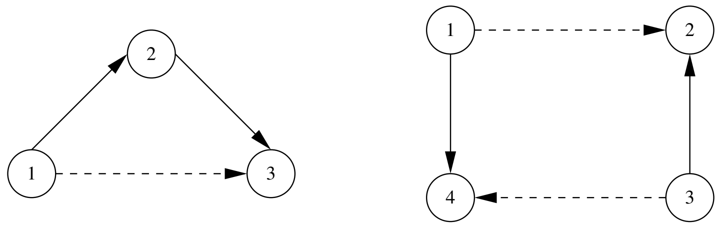


Figure 5.6: *The two SP-graphs indicated by solid and dashed arcs on the left induce a potentially infeasible structure due to subpath inconsistency. The two SP-graphs on the right induce a potentially infeasible structure called a valid cycle.*

One of the most simple potential conflicts imaginable involves exactly two SP-graphs and is given on the left in Figure 5.6. The structure is induced by only three SP-arcs that form an undirected cycle consisting of two arc disjoint paths. The interpretation of these paths is that they should simultaneously be shortest paths (from 1 to 3). This implies that both paths must be shortest paths, the structure is forcing. If some arc in the pattern is a non-SP-arc, then the structure induce a routing conflict (often called subpath inconsistency or violation of the Bellman property). The actual family of cycles for this instance is

$$\mathcal{C} = \left\{ C^{l'} = F^{l'} \cup B^{l'},\ C^{l''} = F^{l''} \cup B^{l''} \right\}, \tag{5.50}$$

where

$$F^{l'} = B^{l''} = \{(1,2),\ (2,3)\} \quad \text{and} \quad F^{l''} = B^{l'} = \{(1,3)\}. \tag{5.51}$$

A slightly more complicated, but still very simple (compared to what is to come) potential conflict that involves exactly two SP-graphs is given on the right in Figure 5.6. The structure induced by the four SP-arcs that form an undirected cycle is called a feasible or a valid cycle. This time, there is not a simple interpretation or explanation of potential infeasible as above. The family of cycles for this instance is

$$\mathcal{C} = \left\{ C^{l'} = F^{l'} \cup B^{l'},\ C^{l''} = F^{l''} \cup B^{l''} \right\}, \tag{5.52}$$

where

$$F^{l'} = B^{l''} = \{(1,2),\ (3,4)\} \quad \text{and} \quad F^{l''} = B^{l'} = \{(1,4),\ (3,2)\}. \tag{5.53}$$

Note that the circulating flow above use the same undirected cycle twice, forwards and backwards. This is one reason that it is easy to analyze the case with only two SP-graphs, compared to the general case; in principle only one cycle has to be considered.

It seems that the case with two SP-graphs is a very important special case in practice; in [25] computational experiments showed that most infeasible instances contain an infeasible routing pattern that involves two SP-graphs. Actually, 99$\%$ of their infeasible test instances turned out to contain such an infeasible structure. How instances are genereated obviously affect this percentage drastically. But since their instances are constructed to be "almost feasible" this indicates that it is probable that the majority of infeasible instances can be explained by a routing pattern that involves only two SP-graphs.

Motivated by the clarity and practical importance of this class of structures it is examined more thoroughly. First, the routing patterns that involve exactly one generalized SP-graph, and then patterns with two SP-graphs are considered.

An infeasible routing pattern with a single generalized SP-graph, $(A_l \cup \tilde{D}_l, \bar{A}_l)$ say, must be a cycle where the aggregated flow on all arcs are nonpositive. This yields only two cases: either the cycle is a directed cycle in $A_l \cup \tilde{D}_l$ or it is formed by a path consisting only of arcs in $A_l \cap \bar{A}_l$. Since these arcs are parallell, the path can also be seen as a cycle. In the former case with the directed cycle, the flow on all arcs in the cycle is negative and the solution is non-saturating. Obviously, the directed cycle must be simple to be a generator. In the latter case, the flow on arc (i, j) is negative when it is considered as a dual variable associated with A_l and positive when associated with $\bar{A}_l$. This kind of solution must involve a single arc that is used in both directions to be a generator. Note that no destination arc can be involved in any of the two cases above unless an arc in A_l emanates from l.

From the above paragraph, it is clear that all solutions involving exactly one generalized SP-graph are in some sense absurd. If this class is considered anyway, we may without loss of generality assume that a generator is represented by a simple undirected cycle C that is the union of its forward arcs, F, and its backward arcs, B, that is $C = F \cup B$. Further, the set of forward arcs, F, is empty when the solution corresponds to a directed cycle and contains a single arc otherwise. The set of backward arcs, B, is either a simple directed cycle or contains a single arc.

Now consider a simplicial solution, $\theta \in \text{cl}\ \Theta$, involving two generalized SP-graphs where the number of cycles for each SP-graph is 1. It can be represented by

$$\mathcal{C} = \left\{ C^{l'} = F^{l'} \cup B^{l'},\ C^{l''} = F^{l''} \cup B^{l''} \right\}. \tag{5.54}$$

where $C^{l'}$ and $C^{l''}$ are simple undirected cycles and $B^{l'} \subseteq A_{l'}$ and $B^{l''} \subseteq A_{l''}$. Due to the capacity constraint, the arc sets must satisfy

$$F^{l'} \subseteq B^{l''} \quad \text{and} \quad F^{l''} \subseteq B^{l'}. \tag{5.55}$$

Further, equality holds if and only if θ is saturating. Hence, when θ is saturating it can be represented by the simple undirected cycle, $C = F \cup B$, where $F = F^{l'} = B^{l''}$ and $B = F^{l''} = B^{l'}$.

In a non-saturating solution the backward arcs not covered by forward arcs induce a directed cycle, E. That is,

$$E = \left(B^{l''} \setminus F^{l'}\right) \cup \left(B^{l'} \setminus F^{l''}\right). \tag{5.56}$$

If this directed cycle is decomposed into simple directed cycles, each simple cycle consists of exactly two arc disjoint path segments with the same terminal nodes, one segment for the forward and one for the backward arcs. To represent a non-saturating it suffices to provide the arcs that are used backwards by one of the two SP-graphs. That is, $C = F \cup B$, where $F = B^{l''}$ and $B = B^{l'}$.

The above description implies that it is possible to represent some routing patterns that are either infeasible or forcing and involve exactly two generalized SP-graphs by the union of its forward and backward arcs, F and B, just as in the case with a single generalized SP-graph.

Obviously, the set of solutions considered above is a subset of the feasible solutions with two generalized SP-graphs. What may be less obvious is that all generators that involves exactly two generalized SP-graphs can be represented like this.

Theorem 5.4
If θ is a generator of cl Θ where the SP-graph indices of all nonzero variables are either l' or l'', then there is a family of simple cycles,

$$\mathcal{C} = \left\{C^{l'} = F^{l'} \cup B^{l'},\ C^{l''} = F^{l''} \cup B^{l''}\right\} \tag{5.57}$$

and a $t > 0$, such that

$$\theta_{ij}^{l'} = \begin{cases} t & \textit{if } (i,j) \in F^{l'} \\ -t & \textit{if } (i,j) \in B^{l'} \\ 0 & \textit{otherwise} \end{cases} \tag{5.58}$$

and

$$\theta_{ij}^{l''} = \begin{cases} -t & \textit{if } (i,j) \in F^{l''} \\ t & \textit{if } (i,j) \in B^{l''} \\ 0 & \textit{otherwise.} \end{cases} \tag{5.59}$$

Further, $F^{l'} \subseteq B^{l''} \subseteq A_{l''}$ and $F^{l''} \subseteq B^{l'} \subseteq A_{l'}$ and equality holds if and only if θ is saturating. Also, the arcs in the union $E = \left(B^{l''} \setminus F^{l'}\right) \cup \left(B^{l'} \setminus F^{l''}\right)$ forms a, not necessarily simple, directed cycle.

Proof: Without loss of generality assume that $t = 1$ and that the lineality space is empty. The latter assumption implies that generators are extreme rays. Write the incidence vector of aggregated arc flows as

$$\theta' + \theta'' + s = 0, \tag{5.60}$$

where θ' and θ'' are the arc flow parts of $\theta^{l'}$ and $\theta^{l''}$, respectively, and s the slack. Clearly, θ', θ'' and s correspond to oriented cycles.

Assume that $|\theta| \notin \mathbb{B}^A$, that is that (5.58) and (5.59) do not hold. Now decompose the cycle associated with $\theta' = -\theta'' - s$ into $\theta'_1 + \theta'_2 = -\theta''_1 - \theta''_2 - s_1 - s_2$, where $|\theta'_2|, |\theta''_2|, |s_2| \in \{0, t'\}^A$ for some $0 < t' < 1$. This can be achieved by taking the arcs in the cycle associated with θ' that maximize $|\theta'|$. Clearly, θ'_2, θ''_2 and s_2 induce a solution that verifies that θ is not extremal which yields a contradiction.

Since $|\theta| \in \mathbb{B}^A$, it is clear that the induced cycles must be simple. For an arc (i, j) where $\theta^{l'}_{ij} + \theta^{l''}_{ij} = 0$, it clearly holds that one is the reversal of the other. That is, $F^{l'} \subseteq B^{l''} \subseteq A_{l''}$ and $F^{l''} \subseteq B^{l'} \subseteq A_{l'}$.

If θ is saturating, all arcs satisfy $\theta^{l'}_{ij} + \theta^{l''}_{ij} = 0$ and equality holds. In the non-saturating case $\theta' = -\theta'' - s$ and $s \geq 0$ with strict inequality for some arcs. For such an arc, (i, j), it must be the case that $s_{ij} = 1$ and exactly one of $\theta^{l'}_{ij}$ and $\theta^{l''}_{ij}$ is -1. This yields either $(i, j) \in B^{l'}$ and $(i, j) \notin F^{l''}$ or $(i, j) \in B^{l''}$ and $(i, j) \notin F^{l'}$.

Finally consider the arcs in E associated with non-saturated slack arcs. Using (5.60) it is clear that s is a circulation, since sums of circulations yields circulations and θ', θ'' and 0 are circulations. Since $s \geq 0$ the associated cycle must be directed. □

This implies that all generators of the closure of the set of solutions to (PR-Farkas) that involve at most two generalized SP-graph are simplicial solutions and that the number of simple cycles for each SP-graph is 1.

Definition 5.12
The subset of solutions in cl Θ that involve at most two generalized SP-graphs is denoted by Θ^2.

Let us now examine these solutions in detail. The case with a single SP-graph has been covered above. First, the saturating solutions involving two SP-graphs are considered and then non-saturating solutions with two SP-graphs are discussed.

5.4.1 Saturating Solutions with Two SP-Graphs

Let θ be a generator of cl Θ that is saturating and involves two generalized SP-graphs with destination l' and l'', that is, $\theta \in \Theta^2$. Without loss of generality assume that θ has been scaled so that all components are -1, 0 or 1, then it may be represented by

$$\mathcal{C} = \left\{ C^{l'} = F^{l'} \cup B^{l'},\ C^{l''} = F^{l''} \cup B^{l''} \right\}. \tag{5.61}$$

Since θ is a saturating generator the forward arcs for one destination are the backward arcs for the other, that is, $F^{l'} = B^{l''}$ and $F^{l''} = B^{l'}$. Hence, the family $\mathcal{C}$ is determined by the simple undirected cycle, $C = F \cup B$, where

$$F = F^{l'} = B^{l''} \subseteq A_{l''} \text{ and } B = F^{l''} = B^{l'} \subseteq A_{l'}. \tag{5.62}$$

This yields that the relation to θ is given by

$$\theta_{ij}^{l'} = -\theta_{ij}^{l''} = \begin{cases} 1 & \text{if } (i,j) \in F \\ -1 & \text{if } (i,j) \in B \\ 0 & \text{otherwise.} \end{cases} \tag{5.63}$$

Note that this representation also includes the case with a single generalized SP-graph. A directed cycle can be obtained if $F = \emptyset$ and a solution from $A_l \cap \bar{A}_l \neq \emptyset$ is obtained by setting F to the non-SP-arcs and B to the SP-arcs.

By construction, the family of cycles induced by the cycle $C = F \cup B$ is feasible. Now consider when it is improving; Proposition 5.3 yields the following.

Proposition 5.7
Let θ be a saturating generator of Θ^2 and $\mathcal{C}$ the corresponding family of cycles, i.e.

$$\mathcal{C} = \left\{ C^{l'} = F^{l'} \cup B^{l'},\ C^{l''} = F^{l''} \cup B^{l''} \right\}. \tag{5.64}$$

When $C = F \cup B$ is given by equation (5.62)*, the solution θ is improving if and only if $F \cap \bar{A}_{l'} \neq \emptyset$ or $B \cap \bar{A}_{l''} \neq \emptyset$.*

Proof: The relation in (5.63) yields

$$\theta_{ij}^{l'} > 0 \quad \text{if } (i,j) \in F \quad \text{and} \quad \theta_{ij}^{l''} > 0 \quad \text{if } (i,j) \in B. \tag{5.65}$$

Now the result follows immediately from Proposition 5.3: $\theta \in \Theta^{=}$ is improving if and only if there is a commodity l and an arc (i,j) such that $(i,j) \in \bar{A}_l$ and $\theta_{ij}^l > 0$. □

Improving cycles are called valid since the above is a generalization of the valid cycles defined in [25, 26] to the non-connected case.

Definition 5.13
A feasible and improving cycle, $C = F \cup B$, is called a generalized saturating valid cycle.

This definition yields the following necessary condition for realizability.

Proposition 5.8
Let $\mathcal{A}_L$ be a family of partial SP-graphs. If L contains two destinations, l' and l'' say, such that $A_{l'} \cup \tilde{D}_{l'}$ and $A_{l''} \cup \tilde{D}_{l''}$ induce a generalized saturating valid cycle, then $\mathcal{A}_L$ is not partially realizable.

The structure of the feasible cycles may be decribed 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

$$F = \bigcup_{p=1}^{K} \overrightarrow{P}_p \quad \text{and} \quad B = \bigcup_{p=1}^{K} \overleftarrow{P}_p. \tag{5.66}$$

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

$$C = \overrightarrow{P}_1 \overleftarrow{P}_1 \cdots \overrightarrow{P}_K \overleftarrow{P}_K. \tag{5.67}$$

If it is assumed that there is no emanating arc from l' in $A_{l'}$ or from l'' in $A_{l''}$ it is clear that a potential destination arc must be the last arc in a path segment and that this segment ends at the destination node. This decomposition yields that any saturating generator obtained from two generalized SP-graphs can be described by the template in Figure 5.7. The interpretation of the figure is as follows. The solid and dashed arrows are associated with the generalized SP-graph with destination l' and l'' respectively. A curly arrow represent a path segment with arbitrarily many arcs (including 0 and 1) and the ordinary arrows are destination arcs. So if $C = F \cup B$, then all arcs in the solid and dashed path segments are in B and F, respectively.

In Figure 5.7 there are $K = 3$ path segments for each SP-graph but it is obvious that this construction can be generalized to any K. Also, any path segment can be removed. That is, $\overrightarrow{P}_i = \emptyset$ or $\overleftarrow{P}_i = \emptyset$, if the corresponding terminal nodes are merged together.

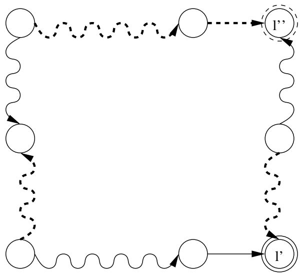


Figure 5.7: *A template for generalized saturating valid cycles. A solid (dashed) arrow is associated with the generalized SP-graph with destination l' (l''). A curly (straight) arrow represent a path segment (destination arc). For the generalized saturating valid cycle $C = F \cup B$, all solid (dashed) arcs are in B (F).*

An account of the relation between the generalized saturating valid cycles and other classes of solutions discussed in the litterature 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: subpath optimality in [6], the Bellman property in [9] and subpath consistency in [26]. A routing pattern that fails to satisfy the above conditions yields a solution to (PR-Farkas) associated with two cycles that consist of two disjoint path segments. That is, the simplest case of a generalized saturating valid cycle. In [25, 26], the valid cycle can contain any number of path segments and corresponds to a our generalized saturating valid cycles without destination arcs.

Infeasible patterns involving destination arcs that induce paths to the destinations have not, as far as we know, been described as a class of solutions. However, in [96], valid inequalities are based on this class of solutions, but they only handle the case with one path segment. Similar inequalities can also be found in [9] and [40]. None of the above papers handle the case with two or more path segments.

5.4.2 Non-Saturating Solutions with Two SP-Graphs

Let $\theta \in \Theta^2$ be a non-saturating generator that involves the generalized SP-graphs with destination l' and l''. From Proposition 5.2 it is clear that a non-saturating solution is improving, hence $\theta \in \Theta$. Again, assume that θ has been scaled so that all components are -1, 0 or 1, and is represented by the family of cycles

$$\mathcal{C} = \left\{ C^{l'} = F^{l'} \cup B^{l'},\, C^{l''} = F^{l''} \cup B^{l''} \right\}, \tag{5.68}$$

where

$$E = E' \cup E'' = \left(B^{l'} \setminus F^{l''} \right) \cup \left(B^{l''} \setminus F^{l'} \right) \tag{5.69}$$

induces a directed cycle when the orientation induced by the labelings is taken into account.

In this case, it is not possible to describe θ by a simple undirected cycle as above since $F^{l'} \neq B^{l''}$ and $F^{l''} \neq B^{l'}$. It is however still possible to represent it as $C = F \cup B$, only including the arcs used backwards. That is,

$$F = B^{l''} \quad \text{and} \quad B = B^{l'}. \tag{5.70}$$

To obtain the cycles $C^{l'}$ and $C^{l''}$ from C the arcs in $E = E' \cup E''$ can be used as follows.

$$C^{l'} = F^{l'} \cup B^{l'} = (F \cup B) \setminus E'' = C \setminus E'' \tag{5.71}$$

and

$$C^{l''} = F^{l''} \cup B^{l''} = (F \cup B) \setminus E' = C \setminus E'. \tag{5.72}$$

Hence the relation to θ is described by

$$\theta^{l'}_{ij} = \begin{cases} 1 & \text{if } (i,j) \in F \setminus E'' \\ -1 & \text{if } (i,j) \in B \\ 0 & \text{otherwise,} \end{cases} \tag{5.73}$$

$$\theta^{l''}_{ij} = \begin{cases} -1 & \text{if } (i,j) \in F \\ 1 & \text{if } (i,j) \in B \setminus E' \\ 0 & \text{otherwise.} \end{cases} \tag{5.74}$$

This structure, $C = F \cup B$ (sometimes along with E' and E''), is called a generalized non-saturating valid cycle. It is non-saturating and therefore also improving, since it is so similar to the generalized saturating valid cycles above, it is justified to call it a valid cycle. As above, the absence of this structure is a necessary condition for realizability.

Proposition 5.9
Let $\mathcal{A}_L$ be a family of partial SP-graphs. If L contains two destinations, l' and l'' say, such that $A_{l'} \cup \tilde{D}_{l'}$ and $A_{l''} \cup \tilde{D}_{l''}$ induce a generalized non-saturating valid cycle, then $\mathcal{A}_L$ is not partially realizable.

Let us again apply the decomposition of F and B into path segments. When the number of segments is K, the decomposition becomes

$$F = \bigcup_{p=1}^{K} \overrightarrow{P}_p \quad \text{and} \quad B = \bigcup_{p=1}^{K} \overleftarrow{P}_p, \tag{5.75}$$

where the path segments are again alternating. Thus, the generalized non-saturating valid cycle can also be described by

$$C = \overrightarrow{P}_1 \overleftarrow{P}_1 \cdots \overrightarrow{P}_K \overleftarrow{P}_K. \tag{5.76}$$

When there are no emanating arcs from l' in $A_{l'}$ or from l'' in $A_{l''}$ a potential destination arc is the last arc in the path segment that ends in the destination node.

A template can be given in a similar manner as for the the saturating case above. But we must not include optional arcs, otherwise an exponential number of templates have to be considered. The template for a generalized non-saturating valid cycle is given on the left in Figure 5.8. Here it is optional to include one or both of the path arcs between a pair of nodes. Further, all path segment 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, cf. the explanation of Figure 5.7 on page 95. A realization of the template is also given on the right in Figure 5.7.

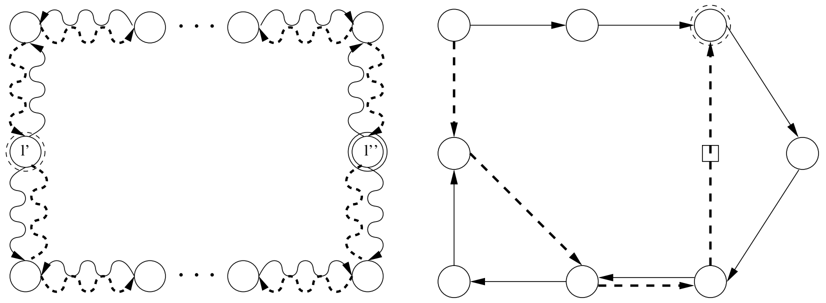


Figure 5.8: *(Left) The template for generalized non-saturating valid cycles. A dashed (solid) arrow is associated with the generalized SP-graph with destination l' (l''). A curly arrow represents a path segment. For the generalized non-saturating valid cycle $C = F \cup B$, all solid (dashed) arcs are in B (F). (Right) A realization of the template on the left, the straight arrows now represents actual arcs and the square marked arrow is a destination arc. This realization stems from a template with 6 nodes and 5 path arcs. Three of these path arcs have been used in both directions which yielded the two 3-cycles and the single 2-cycle, one path arc was used to create the two upper ordinary arcs and the remaining path arc was used to create a corresponding ordinary arc.*

Remark 5.11. Three path arcs were used in both directions in the realization on the right in Figure 5.7. Note that there exist $8 = 2 \cdot 2 \cdot 2$ saturating solutions that use a subset of these arcs for this realization, cf. Corollaries 5.2 and 5.3 and Section 7.3 in Chapter 7, in particular Theorems 7.2 and 7.3. This clearly shows that the non-saturating valid cycle above is not irreducible.

The class of generalized non-saturating valid cycles has not been considered earlier in the litterature and no related class of solutions have been presented. However, as mentioned above the class is connected to the generalized saturating valid cycles which in turn are related to valid cycles.

The following observation is actually a corollary to Theorem 7.2 in Chapter 7.

Corollary 5.2
If there exists a generalized non-saturating valid cycle, then there exists a feasible family of cycles that is saturating and involves two generalized SP-graphs.

Recall that a saturating solution obtained from a non-saturating is not necessarily improving. But, the completion of a family of generalized SP-graphs always yields an improving solution.

Corollary 5.3
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. $A = A_l \cup \tilde{D}_l \cup \bar{A}_l$ and $A_l \cup \tilde{D}_l$ is acyclic for $l \in \{l', l''\}$. Then, if these SP-graphs form a generalized non-saturating valid cycle they also form a generalized saturating valid cycle.

Proof: Let $C = F \cup B$ and $E = E' \cup E''$ describe the generalized non-saturating valid cycle. Consider the directed cycle E, by assumption E is not contained in $A_{l'} \cup \tilde{D}_{l'}$ or $A_{l''} \cup \tilde{D}_{l''}$ and since the SP-graphs have been completed at least one arc in every directed cycle must be a non-SP-arc. By symmetry, assume that it is in F. Now consider the undirected cycle $\bar{C} = \bar{F} \cup \bar{B}$ where $\bar{F} = F$ and $\bar{B} = B \setminus E'$. It is clear that $\bar{C}$ is feasible. Since $\bar{F} \cap \bar{A}_{l'} \neq \emptyset$, Proposition 5.7 yields that $\bar{C}$ is improving. $\square$

The above results are similar to Theorem 5 in [24] (page 518) that states that there is no (improving) solution with two commodities unless there is a valid cycle. However, our statements are more specific and general since a more general case is considered and since non-saturating solutions are taken into account. Not considering non-saturating solutions is actually a flaw in their derivation of the theorem.

The coherency of the case with two generalized SP-graphs has allowed us to describe and analyze *all generators*. This class of solution will be very important when valid inequalities are derived for SPRD problems. A crucial task is to separate such inequalities. The foundation of a separation algorithm is to establish if a pair of generalized SP-graphs induce (non-) saturating valid cycles. This problem is solved in the next section.

5.4.3 Algorithms to Find Generalized Valid Cycles

From Corollary 5.2 it is clear that there exists a generalized non-saturating valid cycle only if there also exists a saturating feasible family of cycles. Therefore, one should always begin with a search for saturating feasible cycles. If such cycles are found their validity can be checked and afterwards an algorithm may be applied to search for a non-saturating valid cycle, if required.

Consider the problem of finding a saturating feasible cycle formed by two generalized SP-graphs, $(A_{l'} \cup \tilde{D}_{l'}, \bar{A}_{l'})$ and $(A_{l''} \cup \tilde{D}_{l''}, \bar{A}_{l''})$ say. A couple of simple observations are the foundation of our algorithm which is closely related to the algorithms in [26]. Basically, the search for a feasible cycle has simply been adapted to the case with ordinary and destination arcs.

Let $C = F \cup B$ represent a saturating solution. Recall that C becomes directed if the orientation induced by the labelings is taken into account. Therefore, construct the graph $\bar{G}$ from the empty graph as follows. Add all arcs in $A_{l'} \cup \tilde{D}_{l'}$ and the reverse of all arc in $A_{l''} \cup \tilde{D}_{l''}$ to $\bar{G}$. This construction yields following lemma.

Lemma 5.1
A feasible cycle $C = F \cup B$ yields a directed cycle in $\bar{G}$.

This implies that it is sufficient to consider directed cycles in $\bar{G}$. A simple directed cycle necessarily lies within a non-trivial strongly connected component with at least 2 nodes. Therefore it is sufficient to find and examine the strongly connected components of $\bar{G}$ to find a feasible cycle.

Remark 5.12. Any simple directed cycle lies within a strongly connected component and all strongly connected component with at least 3 nodes correspond to a feasible cycle. However, not all strongly connected component with exactly 2 nodes correspond to a feasible cycle. A component consisting of the nodes i and j is induced if $(i,j) \in (A_{l'} \cup \tilde{D}_{l'}) \cap (A_{l''} \cup \tilde{D}_{l''})$ or $(i,j) \in (A_l \cap \bar{A}_l$ for some $l \in \{l', l''\}$. Considering the possible cases induced by the above yields that a strongly connected component containing only the nodes i and j correspond to a feasible cycle unless $(i,j) \in A_{l'} \cap A_{l''}$ and $(i,j) \notin A_l \cap \bar{A}_l$ for both $l \in \{l', l''\}$.

To determine if there is a generalized saturating valid cycle, the arc sets $\bar{A}_{l'}$ and $\bar{A}_{l''}$ matters. From Proposition 5.7 it is clear that a directed cycle $\bar{C}$ in $\bar{G}$ corresponds to a generalized saturating valid cycle if and only if $(i,j) \in \bar{A}_{l'}$ and $(j,i) \in \bar{C}$ or $(i,j) \in \bar{A}_{l''}$ and $(i,j) \in \bar{C}$. This yields the following theorem which is a generalization of a main theorem in [26].

Theorem 5.5
The generalized SP-graphs $(A_{l'} \cup \tilde{D}_{l'}, \bar{A}_{l'})$ and $(A_{l''} \cup \tilde{D}_{l''}, \bar{A}_{l''})$ induce a generalized saturating valid cycle if and only if a strongly connected component of $\bar{G}$ contains an arc (i,j) such that $(j,i) \in \bar{A}_{l'}$ or $(i,j) \in \bar{A}_{l''}$.

Note that the case where a strongly connected component of $\bar{G}$ contains exactly two nodes is also covered by the theorem. If there is no strongly connected component of $\bar{G}$ with at least 3 nodes then there is no family of feasible cycles with at least 3 nodes and no use in searching for a non-saturating valid cycle. Otherwise, the following observation can be used as the foundation of an algorithm. A simplest non-saturating valid cycle contains a simple directed cycle and is therefore an Euler tour that contains as a subgraph the directed cycle formed by two directed path segments, $E = E' \cup E''$. The directed paths, E' and E'', only contain SP-arcs from the generalized SP-graphs with destination l' and l'', respectively. Clearly, there is a pair of nodes, i and j, in a strongly connected component of $\bar{G}$ such that E' is an $i-j$-path and E'' a $j-i$-path. This observation is used below to find a non-saturating valid cycle in a strongly connected component of $\bar{G}$, if one exists.

For an arbitrary graph G, let $T(G)$ be the matrix that represents the transitive closure of G. That is, $T(G)_{ij} = 1$ if there is a path from i to j in G and 0 otherwise. Consider the matrix R that is the sum of the transitive closure of $A_{l'} \cup \tilde{D}_{l'}$ and the transitive closure of the reverse of $A_{l''} \cup \tilde{D}_{l''}$. That is

$$R = T(A_{l'} \cup \tilde{D}_{l'}) + T(A_{l''} \cup \tilde{D}_{l''})'. \tag{5.77}$$

For a node pair, (i,j), the matrix entry $R_{ij} = 2$ if and only if j is reachable from i in $A_{l'} \cup \tilde{D}_{l'}$ and i reachable from j in the reverse of $A_{l''} \cup \tilde{D}_{l''}$.

Hence, if i and j is the node pair and E the directed cycle above, then $R_{ij} = 2$ which yields a generalized non-saturating valid cycle. If there is no node pair i and j in the same connected component of $\bar{G}$ such that $R_{ij} = 2$, then the SP-graphs do not induce a generalized non-saturating valid cycle. This yields the following.

Theorem 5.6
The generalized SP-graphs $(A_{l'} \cup \tilde{D}_{l'}, \bar{A}_{l'})$ and $(A_{l''} \cup \tilde{D}_{l''}, \bar{A}_{l''})$ induce a generalized non-saturating valid cycle if and only if a strongly connected component of $\bar{G}$ with at least 3 nodes contains a node pair, (i,j), such that $i \neq j$ and $R_{ij} = 2$, where R is the matrix defined in (5.77).

Two algorithms have implicitly outlined in the theorems above. From Theorem 5.5 the following algorithm is obtained to find a generalized valid cycle among a pair of generalized SP-graphs.

Algorithm 5.4.1. Given two generalized SP-graphs $(A_{l'} \cup \tilde{D}_{l'}, \bar{A}_{l'})$ and $(A_{l''} \cup \tilde{D}_{l''}, \bar{A}_{l''})$, find an induced generalized saturating valid cycle if one exists.

1. Form the graph $\bar{G}$ as follows.
   - For each arc $(i,j) \in A_{l'} \cup \tilde{D}_{l'}$, add (i,j) to $\bar{G}$.
   - For each arc $(i,j) \in A_{l''} \cup \tilde{D}_{l''}$, add (j,i) to $\bar{G}$.
2. Find the strongly connected components, $C_1, \ldots, C_K$ of $\bar{G}$.
3. For each strongly connected component, C_k, such that $|C_k| = 2$ do the following.
   - If possible, choose an arc $(i,j) \in C_k$ such that $(j,i) \in \bar{A}_{l'}$ or $(i,j) \in \bar{A}_{l''}$ form the generalized saturating valid cycle from (i,j) as follows.
     (a) If $(i,j) \in \bar{A}_{l''}$ set $C = (i,j) \cup (j,i)$.
     (b) If $(j,i) \in \bar{A}_{l'}$ set $C = (j,i) \cup (i,j)$.

   (c) Return the **generalized valid cycle,** C.

4. For each strongly connected component, C_k, such that $|C_k| \geq 3$ do the following.

   - If possible, choose an arc $(i,j) \in C_k$ such that $(j,i) \in \bar{A}_{l'}$ or $(i,j) \in \bar{A}_{l''}$ then find a path from node j to node i as follows.
     (a) Do a breadth first search from node j until node i is reached. Store the parent indices.
     (b) Unravel the path from j to i by the parent indices, starting from i.
        i. For each arc (s,t) in the path where $(s,t) \in A_{l'} \cup \tilde{D}_{l'}$, add (s,t) to F.
        ii. For each arc (s,t) in the path where $(t,s) \in A_{l''} \cup \tilde{D}_{l''}$, add (t,s) to B.
        iii. (Optional) If an arc was added to F and its reversal to B, then remove it from F (or its reversal from B).
     (c) Form the generalized saturating valid cycle from the path and the arc (i,j).
        i. If $(i,j) \in \bar{A}_{l''}$ add (i,j) to F.
        ii. If $(j,i) \in \bar{A}_{l'}$ add (j,i) to B.
        iii. (Optional) If (i,j) was added to F and (j,i) to B, then remove (i,j) from F (or (j,i) from B).
        iv. Let

$$C = F \cup B.$$

     (d) Return the **generalized valid cycle,** C.

5. Return **there exist no generalized valid cycle.**

Remark 5.13. If the optional steps are omitted a generalized non-saturating valid cycle may be found instead of a saturating. In such a non-saturating valid cycle the directed cycle E will consist of exactly two arcs and nodes.

In step 2, the strongly connected components can be found by some standard linear time algorithm, e.g. Tarjan [94], Kosaraju (cf. [90]) or Gabow [53]. Clearly, the algorithm runs in linear time, $\mathcal{O}(\tilde{m}+n)$, since the strongly connected components are found in $\mathcal{O}(\tilde{m}+n)$ and then the arcs are considered at most once again.

Remark 5.14. There may be two benefits with using Gabows algorithm in step 2: it is path-based and also computes the condensation of $\bar{G}$ as a byproduct. The former fact might be exploitable in our case since ingraphs are used, the latter may be useful if a dynamic version of the problem is considered where arcs are added to $\bar{G}$ (via $A_{l'}$ or $A_{l''}$).

When an "improving" arc is found in step 4 that makes the component contain an improving cycle there are different ways to find the cycle. The suggested method finds a shortest (w.r.t. number of arcs) cycle containing the given arc with a breadth first search.

Other methods that should be taken into account are the following. Define a metric w where the length of an arc, (s,t), is 1 if $(t,s) \in \bar{A}_{l'}$ or $(s,t) \in \bar{A}_{l''}$ and 0 otherwise. Find a longest (simple) path from j to i. Alternatively, but not equivalently, use the metric

$1-w$ and find a shortest path from j to i. The idea behind both these approaches is to find generalized valid cycles that contain several improving arcs. Note that it is NP-hard to find the longest path, but this is not likely to be a problem in practice where the strongly connected components are usually small and sparse.

All the above methods seem reasonable and have their advantages. Note that all of them are in some sense "local", e.g. the breadth first search finds a shortest cycle *containing the given arc*. Instead of returning the first cycle found, it may be a good idea to iterate through all improving arcs to find a "globally" best generalized valid cycle, e.g. finding *the* shortest cycle.

Let us now turn to the slightly more complicated problem of finding a generalized non-saturating valid cycle. Theorem 5.6 yields the following algorithm.

Algorithm 5.4.2. Given two generalized SP-graphs $(A_{l'} \cup \tilde{D}_{l'}, \bar{A}_{l'})$ and $(A_{l''} \cup \tilde{D}_{l''}, \bar{A}_{l''})$, find an induced generalized non-saturating valid cycle if one exists.

1. Form the multigraph $\bar{G}$ as follows.
   - For each arc $(i,j) \in A_{l'} \cup \tilde{D}_{l'}$, add (i,j) to $\bar{G}$.
   - For each arc $(i,j) \in A_{l''} \cup \tilde{D}_{l''}$, add (j,i) to $\bar{G}$.
2. Find the strongly connected components, $C_1, \ldots, C_K$ of $\bar{G}$.
3. For each strongly connected component, C_k, such that $|C_k| \geq 3$ do the following.
4. For each arc $(i,j) \in C_k$,
   - If $|\delta^+(i) \cap C_k| \geq 2$ and $|\delta^-(j) \cap C_k| \geq 2$ then determine if $R_{ij} = 2$, where R is the matrix defined in (5.77) as follows.
     (a) Perform a breadth first search in $A_{l'} \cup \tilde{D}_{l'}$ from node i (stop if node j is reached). Store the parent indices.
     (b) If node j was not reached, continue with the next arc in the component.
     (c) Perform a breadth first search in the reverse of $A_{l''} \cup \tilde{D}_{l''}$ from node j (stop if node i is reached). Store the parent indices.
     (d) If node i was not reached, continue with the next arc in the component.
     (e) A generalized non-saturating valid cycle has been found, extract it as follows.
        i. Perform a breadth first search in $\bar{G}$ from node j until node i is reached. Store the parent indices.
        ii. Unravel the path from j to i by the parent indices. Determine the forward arcs, F, and the backward arcs, B, from the $j-i$-path.
        iii. Use the parent indices from step 4a to unravel the path from i to j in $A_{l'} \cup \tilde{D}_{l'}$. Add all arcs in the path to E'.
        iv. Use the parent indices from step 4c to unravel the path from j to i in the reverse of $A_{l''} \cup \tilde{D}_{l''}$. Add all arcs in the path to E''.

(f) Form the generalized non-saturating valid cycle from the arc sets above.

$$C = (F \cup E') \cup (B \cup E'') .$$

(g) return the **generalized non-saturating valid cycle,** C.

5. return **there exist no generalized non-saturating valid cycle.**

It is clear that the algorithm above is not as efficient as the one that finds saturating valid cycles. In the worst case, the additional breadth first searches have to be performed for all node pairs. This yields the time complexity $\mathcal{O}\left(\tilde{m}\left(\tilde{m}+n\right)\right)$. In practice it is doubtful that the strongly connected components are large and that many node pairs have large degrees within the components.

Clearly, the comments and suggested modifications of Algorithm 5.4.1 also applies to Algorithm 5.4.2. A further modification that may be considered is to calculate the transitive closures of the generalized SP-graphs in advance. Note that this may definitely be preferred if all pairs of SP-graphs are tested for generalized valid cycles. This modification significantly improves the teoretical time complexity and the calculation of the transitive closures likely becomes the most costly operation. If a straightforward implementation is used to calculate the transitive closures the overall time complexity becomes $\mathcal{O}\left(n\tilde{m}\right)$. Other, more suitable and efficient alternatives can also be considered, e.g. since the generalized SP-graphs are acyclic the chain decomposition approach in [91] is applicable. It is unclear how this modification affect the practical efficiency.

5.5 On the Relation Between Classes of Structures

In total, five classes of solutions have been considered in this chapter. They are, in decreasing order of generality, referred to as: general, binary, unitary, simplicial and generalized valid cycle solutions. It must of course be proved that all these subclasses are proper subclasses.

The following notation will be used. The collections of all cycle families that correspond to an irreducible generator are denoted by:

- $\mathcal{S}^2$ for simplicial solutions that involve at most two destinations,
- $\mathcal{S}$ for simplicial solutions,
- $\mathcal{U}$ for unitary solutions,
- $\mathcal{B}$ for binary solutions,
- $\mathcal{G}$ for general solutions.

Similarily, 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. This yields the following theorems.

Theorem 5.7
The relation between the collections of all cycle families that correspond to an irreducible generator of a certain type is as follows,

$$\mathcal{S}^2 \subset \mathcal{S} \subset \mathcal{U} \subset \mathcal{B} \subset \mathcal{G}. \tag{5.78}$$

Proof: Inclusion holds trivially from the definitions in Section 5.2. Examples in Chapter 6 will show that all inclusions are strict. □

Equivalently for collections of instances, we have the following.

Theorem 5.8
The relation between the collection of instances where the least complicated solution is of a certain type is as follows,

$$\mathcal{S}_I^2 \subset \mathcal{S}_I \subset \mathcal{U}_I \subset \mathcal{B}_I \subset \mathcal{G}_I. \tag{5.79}$$

We continue our analysis of the class of simplicial solutions in the next chapter.

6

Simplicial Cycle Families

A study of infeasible shortest path routing (SPR) patterns was initiated in the previous chapter. This investigation is continued here for the more complex class of solutions that we call simplicial solutions. The aim of the chapter is threefold. We want to get a deeper understanding of the simplicial structure, prove the relation theorems stated at the end of the previous chapter and characterize generators and irreducible solutions.

The first goal is, to a large extent, achieved by considering the correspondence between the simplicial structures and graph embeddings. This yields powerful tools for constructing interesting example instances which is very important to get a profund understanding of infeasibility. It is also a first step towards our second goal to prove Theorems 5.7 and 5.8. For this, it is sufficient to provide examples of instances where a least complicated solution belongs to the infeasible structures considered in Section 5.2 of Chapter 5. As a tool, we will also define a dependency concept based on a graph closely related to the dual graph. This can be used to easily explain if a simplicial solution is a generator or an irreducible generator. Hence, our third goal to characterize generators and irreducible solutions can be fulfilled for the class of simplicial solutions.

The chapter outline is as follows. The simplicial structure is reconsidered in Section 6.1 and a correspondence with graph embeddings is shown. In Section 6.2, the dependency graph is defined and used to characterize when simplicial cycle families are generators and irreducible generators. Finally, the relation theorems (Theorems 5.7 and 5.8) from Chapter 5 are proved in Section 6.3.

6.1 A Characterization of the Simplicial Structure

Recall that the inverse partial shortest path routing problem (IPSPR) is considered and that we seek infeasible solutions by analyzing the Farkas system of model (PR) on page 72 in Chapter 5. This Farkas system, model (PR-Farkas), can be found on page 73.

Consider a saturating feasible solution, θ, to (PR-Farkas) that is canonical and simplicial, cf. Definition 5.4 on page 78, Definition 5.8 on page 82 and Definition 5.9 on page 82. Recall from Chapter 5 that θ can be represented by the cycle family

$$\mathcal{C} = \{C^l \mid C^l = F^l \cup B^l,\ B^l \subseteq A_l\} \tag{6.1}$$

or the destination index sets, F_{ij} and B_{ij}, where

$$F_{ij} = \{l : (i,j) \in F^l\} \quad \text{and} \quad B_{ij} = \{l : (i,j) \in B^l\} \tag{6.2}$$

for each arc $(i,j) \in \tilde{A}$. By assumption, it holds that

$$\theta^l_{ij} = \begin{cases} 1 & \text{if } l \in F_{ij}, \text{or equivalently, if } (i,j) \in F^l \\ -1 & \text{if } l \in B_{ij}, \text{or equivalently, if } (i,j) \in B^l \\ 0 & \text{otherwise.} \end{cases} \tag{6.3}$$

At the moment, it may not be apparent from above and Definition 5.8 in Chapter 5 that the class of simplicial structures is actually a very comprehensible class. To characterize and analyze this class we will show in Sections 6.1.1 and 6.1.2 that a simplicial cycle family corresponds to an embedding of a graph. But first, an introductory example is given.

In [23] an infeasible instance with a simplicial solution and without a valid cycle was given. Other authors have given other examples, e.g. the conflict specified by paths on page 77 in [9] induces SP-graphs that yield a unitary solution but no valid cycle and the conflict on page 79 in [9] induces SP-graphs that yield a simplicial solution but no valid cycle. One of these examples are considered in terms of SP-graphs in Example 6.11.

A smaller example of a simplicial solution without valid cycles than in [23] will be given here (w.r.t the number of nodes and arcs in the SP-graphs). Actually, our example is a minimal example with a simplicial solution and no valid cycle. Since there should be no valid cycle there must be at least 3 SP-graphs. It is straightforward to rule out all cases with at most 4 nodes. Since our ingraphs are trees, minimality is obtained.

Example 6.1

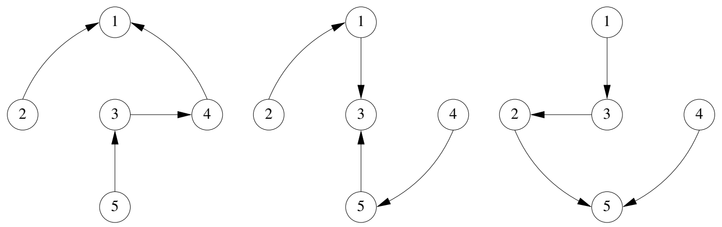


Figure 6.1: *The SP-graphs $(A_1, \bar{A}_1)$, $(A_3, \bar{A}_3)$ and $(A_5, \bar{A}_5)$, respectively. The figure should be interpreted as follows. A drawn arc represents that the arc is an SP-arc in the associated SP-graph. Arcs not drawn are non-SP-arcs.*

Consider the three intrees in Figure 6.1 and the graph induced by 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 Algorithm 5.4.1 in Section 5.4.3.

There is however a feasible solution to (PR-Farkas). Consider the family of cycles in Figure 6.2 defined by

$$\mathcal{C} = \left\{C^1 = F^1 \cup B^1,\ C^3 = F^3 \cup B^3,\ C^5 = F^5 \cup B^5\right\}, \tag{6.4}$$

where

$$\begin{aligned} F^1 &= \{(2,1),\ (3,2)\}, & B^1 &= \{(3,4),\ (4,1)\},\\ F^3 &= \{(2,5),\ (4,1)\}, & B^3 &= \{(2,1),\ (4,5)\},\\ F^5 &= \{(3,4),\ (4,5)\}, & B^5 &= \{(2,5),\ (3,2)\}. \end{aligned} \tag{6.5}$$

This $\mathcal{C}$ yields a θ-solution via (5.24) that is a feasible solution to (PR-Farkas). Actually, all solutions to (PR-Farkas) lie on the extreme ray generated by this θ.

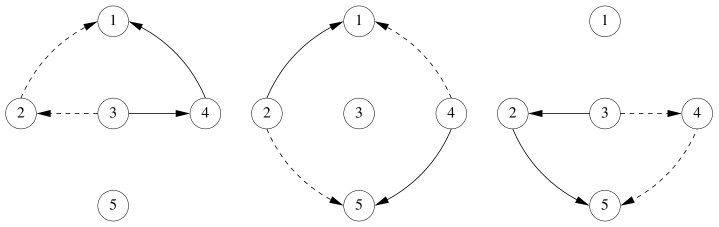


Figure 6.2: *The cycles $C^1 = F^1 \cup B^1$, $C^3 = F^3 \cup B^3$ and $C^5 = F^5 \cup B^5$, respectively. A dashed arc (i, j) is a forward arc, i.e. $(i, j) \in F^l$ and $\theta^l_{ij} > 0$. A solid arc (i, j) is a backward arc, i.e. $(i, j) \in B^l$ and $\theta^l_{ij} < 0$.*

The example in [9] has exactly 4 nodes, 8 arcs and 4 SP-graphs, it also contains a more "complicated" solution, cf. Example 6.11.

6.1.1 Graph Embeddings Yield Simplicial Cycle Families

The starting point in this section is that an instance that contains a simplicial solution is to be constructed.

First consider the problem to create an instance with a simplicial solution that involves at most two destinations, that is a valid cycle. This is easily achieved by taking an arbitrary undirected cycle and assign forward and backward labels to the edges. To make the valid cycle correspond to a subpath inconsistency conflict, all forward (and hence, backward) arcs must be sequential.

A less trivial task is to create an instance with a simplicial solution that involves more than two destinations. Preferably also as a least complicated solution. The solution of this problem accentuate the close relation between cellular graph embeddings, oriented cycle double covers and simplicial cycle families. This also yields a connection to the fundamental (oriented) cycle double cover conjecture and the (strong) cellular embedding conjecture in topological graph theory, cf. the book [99] or the survey articles [65, 34] for a thorough treatment of cycle double cover theorems and related conjectures.

Definition 6.1
For an undirected graph G, an oriented cycle double cover is a collection of directed cycles, such that their union covers each edge in G exactly twice, once in each direction.

An oriented cycle double cover easily yields a simplicial solution by orienting all edges, or equivalently, by assigning each edge to an adjacent cycle. Given an orientation of all arcs, the procedure is as follows. First associate a directed cycle with a (possibly fictious) destination. Then partition the arcs in the directed cycle into forward and backward arcs such that they all comply with the orientation of the arc and cycle.

Conjecture 6.1 (Oriented cycle double cover conjecture, [66])
Every bridgeless graph has an oriented cycle double cover.

An earlier unoriented version of this conjecture can be found in [93, 89]. A strongly related conjecture is the cellular embedding conjecture. An embedding of a graph G on a (oriented) surface is a drawing of G on the surface without edge crossings. The embedding is cellular if every face boundary is a (directed) cycle of the graph.

Conjecture 6.2 (Cellular embedding conjecture, [61, 65])
Every 2-connected graph has a cellular embedding in some (orientable) surface.

It is well known that this conjecture implies the (oriented) cycle double cover conjecture since the face boundaries of a cellular embedding constitute a (oriented) cycle double cover for the original graph. Thus, we have all the pieces to construct a simplicial solution. Starting from a cellular embedding of an arbitrary undirected bridgeless graph an oriented cycle double cover is constructed which then yields a simplicial cycle family.

This general procedure is now demonstrated for two graphs. In the first, somewhat more simple example, a planar graph is used. That is, the graph is embedded into the sphere. Then, a little more complex example is considered where the Petersen graph is embedded into a torus.

Example 6.2

The planar drawing of the graph on the left in Figure 6.3 is also called a cellular embedding into the sphere. An oriented cycle double cover is obtained by the 2-cells/2-complexes/faces in the cellular embedding and a direction. If the orientations of the inner faces are choosen to be counter-clockwise the oriented circuits on the right in Figure 6.3, also given below in Equation (6.6), is as an oriented cycle double cover.

$$\begin{array}{rcl}
C_1 & = & (1\,4\,5\,2) \\
C_2 & = & (2\,5\,6\,3) \\
C_3 & = & (4\,7\,8\,5) \\
C_4 & = & (5\,8\,9\,6) \\
C_0 & = & (1\,2\,3\,6\,9\,8\,7\,4).
\end{array} \tag{6.6}$$

Note that the outer face cycle, C_0, is the reverse of the sum of all inner cycles. That is,

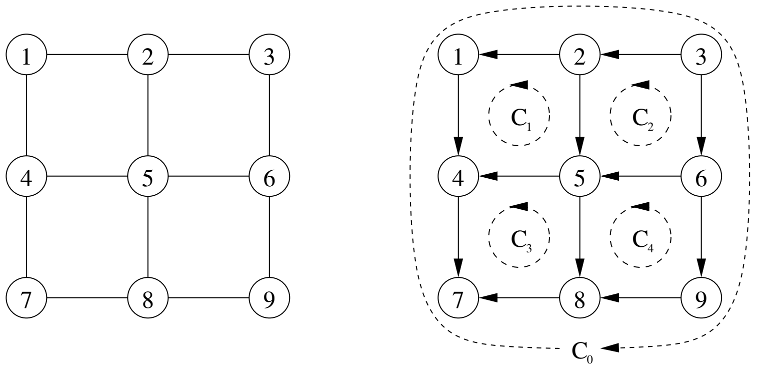


Figure 6.3: A planar embedding of a graph (left) and the oriented cycle double cover induced by the embedding (right). Each face in the embedding yields a cycle that can be directed since the surface (the sphere) is orientable. A simplicial family of cycles can be obtained from the arc orientation on the right.

$$-C_0 = C_1 + C_2 + C_3 + C_4. \tag{6.7}$$

The collection of circuits in (6.6) is now turned into a simplicial family of cycles by the procedure outlined above. First, each edge in the graph is assigned (in some arbitrary manner) an orientation. A simple orientation is to point all arcs to the south-west. Consider an inner directed cycle. To comply with this 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 following five cycles,

$$\begin{aligned}
C^a &= \{(1,4),\ (2,1)\} \cup \{(2,5),\ (5,4)\}\\
C^b &= \{(2,5),\ (3,2)\} \cup \{(3,6),\ (6,5)\}\\
C^c &= \{(4,7),\ (5,4)\} \cup \{(5,8),\ (8,7)\}\\
C^d &= \{(5,8),\ (6,5)\} \cup \{(6,9),\ (9,8)\}\\
C^e &= \{(3,6),\ (6,9),\ (8,7),\ (9,8)\} \cup \{(1,4),\ (2,1),\ (3,2),\ (4,7)\},
\end{aligned} \tag{6.8}$$

where the left and right arc sets in the union are the forward and backward arcs, respectively. Note that we do not consider the actual destination associated with the cycles, cf. Remark 6.8.

Remark 6.1. Often, an important issue when constructing examples is that the conflicts are irreducible. The orientation used in Example 6.2 was choosen a bit carelessly and therefore the cycle family became reducible. How to cope with this issue is adressed further below, cf. Example 6.4 and also Sections 6.2 and 6.2.4.

Remark 6.2. 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 (6.8) in Example 6.2. Re-orient the edges between 2 and 5 and also 4 and 5 to point right and up, respectively. This yields the arcs $(4,5)$ and $(5,2)$ and the following non-saturating cycle family

$$\begin{aligned} C^b &= \{(3,2)\} \cup \{(3,6),\ (5,2),\ (6,5)\} \\ C^c &= \{(4,7)\} \cup \{(4,5),\ (5,8),\ (8,7)\} \\ C^d &= \{(5,8),\ (6,5)\} \cup \{(6,9),\ (9,8)\} \\ C^e &= \{(3,6),\ (6,9),\ (8,7),\ (9,8)\} \cup \{(1,4),\ (2,1),\ (3,2),\ (4,7)\}, \end{aligned} \tag{6.9}$$

Note that the cycle C^a has been omitted. It now corresponds to the slack cycle. The above simplicial cycle family is indeed non-saturating. It is however not irreducible since the cycles C^b and C^c induce valid cycles. Our orientation was "badly" choosen, cf. Remark 6.1.

Example 6.3

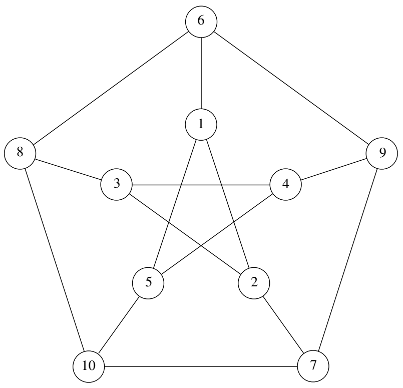


Figure 6.4: *A drawing of the famous Petersen graph in the plane with crossings.*

Consider the Petersen graph in Figure 6.4. To obtain an oriented cycle double cover it can be cellularly embedded into an orientable surface. It is well known that the Petersen graph is toroidal. That is, it can be embedded into the torus. A cellular embedding is given in Figure 6.5.

If we use this embedding and choose the orientation to be counter-clockwise, then the faces of the Petersen graph on the torus embedding in Figure 6.5 yields the following oriented circuits as an oriented cycle double cover.

$$\begin{aligned} C_1 &= (1\ 6\ 9\ 7\ 2) \\ C_2 &= (2\ 7\ 10\ 8\ 3) \\ C_3 &= (3\ 8\ 6\ 1\ 5\ 4) \\ C_4 &= (4\ 5\ 10\ 7\ 9) \\ C_5 &= (5\ 1\ 2\ 3\ 4\ 9\ 6\ 8\ 10). \end{aligned} \tag{6.10}$$

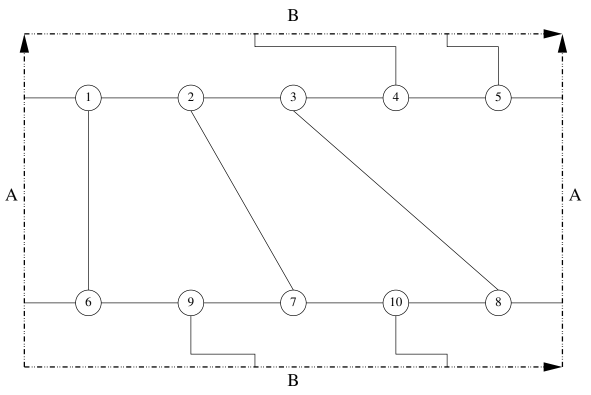


Figure 6.5: *An embedding of the Petersen graph into the torus without crossings. If the polygon is stretched and folded such that arrows on the sides with the same label (A and B) are matched and the graph edges along the two arrows with the same label are sequentially matched from the arrow head to its tail an embedding of the graph without crossing on the torus is obtained.*

To turn the circuits in (6.10) into a simplicial family of cycles each edge can be assigned an orientation. 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 6.6 and the simplicial cycle family consisting of the following five cycles

$$\begin{aligned}
C^1 &= \{(1,6),\ (6,9),\ (9,7)\} \cup \{(2,7),\ (1,2)\} \\
C^2 &= \{(2,7),\ (7,10),\ (10,8)\} \cup \{(3,8),\ (2,3)\} \\
C^3 &= \{(3,8),\ (8,6)\} \cup \{(1,6),\ (5,1),\ (4,5),\ (3,4)\} \\
C^4 &= \{(4,5),\ (9,4)\} \cup \{(10,5),\ (7,10),\ (9,7)\} \\
C^5 &= \{(5,1),\ (1,2),\ (2,3),\ (3,4),\ (10,5)\} \cup \{(9,4),\ (6,9),\ (8,6),\ (10,8)\}.
\end{aligned} \tag{6.11}$$

It has been illustrated above how cellular embeddings can be used to construct simplicial solutions. It is clear that 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, that depends on the orientation as pointed out in Remark 6.1. For the above examples it can easily be verified that the cycle family in Example 6.2 is reducible while (less easily) the cycle family in Example 6.3 is indeed irreducible.

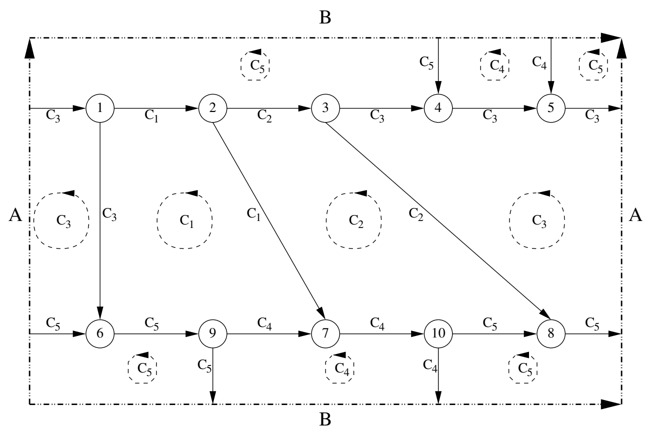


__Figure 6.6:__ The embedding of the Petersen graph on the torus where all arcs have been assigned to an adjacent face.

If the simplicial cycle family obtained from an embedding and orientation is not irreducible we have two measures to cope with this. First, another orientation can of course be tried. This resolves the problem in some, but not all, cases. The other alternative is to use a postprocessing step that subdivides some arcs. This is always effective, but not always desirable since arc subdivisions in a sense destroy the structure of the original graph. The effect of the subdivision is to force a dependency between the faces involved. Put in terms of the dependency graph, to be discussed in Section 6.2, we bi-direct the arc between the two faces.

The postprocessing step is illustrated below by an example. Actually, the procedure has already been carried out earlier when Example 5.2 was continued. In Figure 5.4 on page 88 the arc $(2, 5)$ was subdivided to obtain the graph in Figure 5.5 on page 89.

Example 6.4

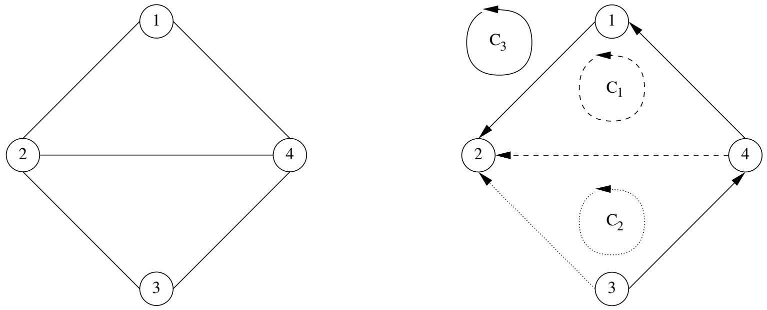


__Figure 6.7:__ (Left) A cellular embedding of a planar graph with four nodes. (Right) An assignment of adjacent faces to all edges.

Consider the planar graph on the left in Figure 6.7. It is easily verified that the simplicial cycle family induced by the orientation on the right in Figure 6.7 is reducible. In fact, it is straightforward to verify than there is no orientation such that the induced cycle family is irreducible since both circuits with three edges will always induce a smaller cycle family.

Let us therefore apply the following postprocessing step. Take an arbitrary arc and subdivide it into two new arcs so that these new arcs become associated with different faces. If this does not produce a cycle family that is irreducible, re-apply the procedure to an arc in the original assignment.

Suppose that arc $(4, 1)$ is first choosen and subdivided into $(1, a)$ and $(4, a)$. To be consistent with the orientation, arc $(1, a)$ must be assigned to C_1 and $(4, a)$ to C_3. Unfortunately, this operation does not produce an irreducible cycle family so the process is repeated. Choose arc $(3, 4)$ and subdivide it into $(b, 3)$ and $(b, 4)$; arc $(b, 3)$ is assigned to C_2 and $(b, 4)$ to C_3. Neither this operation produce an irreducible cycle family. Clearly, it is of no use to consider the arcs $(1, 2)$ or $(3, 2)$ since the inter-dependency between the adjacent circuits have already been forced by the previous two operations. This leaves only arc $(4, 2)$. A consistent subdivision is to assign $(c, 2)$ to C_1 and $(c, 4)$ to C_2. This yields the irreducible cycle family depicted on the left in Figure 6.8.

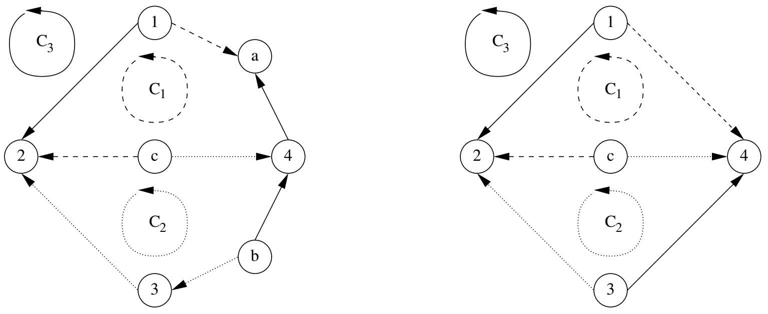


Figure 6.8: *(Left) After the postprocessing step has been applied three times, the final simplicial family of cycles is irreducible. (Right) If arc* $(4, 1)$ *is re-oriented and* $(4, 2)$ *is subdivided a smaller irreducible cycle family is obtained.*

It is not necessary to use three subdivisions for the cycle family induced at the right of Figure 6.7, only two is required. An alternative would be to re-orient arc $(4, 1)$, then it is sufficient to use a single subdivision to obtain an irreducible cycle family where only arc $(4, 2)$ is subdivided. The outcome of this alternative is on the right in Figure 6.8. The reasons for this will become clear in Section 6.2 which deals with the dependency graph of a simplicial cycle family, cf. Definition 6.2.

6.1.2 Simplicial Cycle Families Yield Graph Embeddings

It has been established above that cellular embeddings can be used to construct simplicial solutions. The following theorem concerns the other direction, namely, that a simplicial solution yields a cellular embedding in an orientable surface. This further illustrates the connection to oriented cycle double covers and cellular embeddings and motivates that topological graph theory may be important to get a deeper understanding of model (PR-Farkas) and IPSPR problems.

Theorem 6.1
Let θ be a saturating simplicial solution to (PR-Farkas) *represented by $\mathcal{C}$ and let F_{ij} and B_{ij} be defined as in* (6.2). *Further, let $\bar{\mathcal{C}}$ be a decomposition of $\mathcal{C}$ into circuits and $G\left(\bar{\mathcal{C}}\right) = (N(B), B)$ the graph induced by all backward arcs, i.e.*

$$B = \bigcup_{l \in L} B^l. \tag{6.12}$$

Then, $G\left(\bar{\mathcal{C}}\right)$ is embeddable in an orientable surface such that there is a one-to-one correspondence between the oriented circuits $\bar{\mathcal{C}}$ and the faces of the embedding.

Proof: Let $C_1, \ldots, C_q$ be an enumeration of the oriented circuits in $\bar{\mathcal{C}}$. Take the circuits $C_1, \ldots, C_q$, one at a time, and glue them together along a sequence of common arcs. When this process is finished, a fundamental polygon is obtained where each corner of the polygon is a node in the original graph and the nodes in the graph may occur several times as corners of the polygon. It is a fundamental fact in topology that a fundamental polygon corresponds to an oriented surface if this polygon has an even number of sides and each directed side, say A with corner i followed by corner j, in the polygon also occurs in reverse, that is, A^{-1} with corner j followed by corner i. By construction, this embedding obviously have the desired property that faces corresponds to oriented circuits.

Let us verify that the polygon has the required properties. Since $C_1, \ldots, C_q$ is an oriented cycle double cover each arc appears either 0 or 2 times on the polygon boundry, hence the number of sides in the polygon is even.

By construction, the boundary of the polygon is an oriented circuit. The direction of an arc in this circuit is determined by the oriented circuit that it belongs to. Clearly, since the side in the polygon that corresponds to arc (i, j) is used forwards by some circuit and backwards by some other circuit, each side on the boundary appears in pairs, A and A^{-1} say, where (i, j) is used forwards for side A and backwards for side A^{-1}, or vice versa. □

Remark 6.3. It is no restriction to assume that the solution in Theorem 6.1 is saturating since a slack cycle without backward arcs can be introduced, cf. Remark 6.2.

Let us illustrate the contents in this theorem and the construction in the proof by an example.

Example 6.5

The components in Example 6.3 serve as a good starting point. The embedding of the Petersen graph in Figure 6.5 yields the oriented circuits in (6.10). These circuits can be turned into the simplicial cycle family with the cycles in (6.11).

By the procedure in the proof of Theorem 6.1 let us take these oriented circuits in (6.10) and glue them together. This yields the fundamental polygon in Figure 6.9. This polygon is associated with an orientable surface of genus no more than 8 since there are 16 sides and when the boundary is traversed each side is encountered twice, once forwards, and once backwards. Clearly, the genus can be reduced by merging some sides. In fact, since there are 5 internal nodes and 10/2 nodes on the boundary, 7 internal arcs and 16/2 arcs on the boundary and 5 faces, the Euler characteristic is

$$\chi = n - m + f = (5+5) - (7+8) - 5 = 0, \tag{6.13}$$

and since $\chi = 2 - 2g$ the genus must be 1.

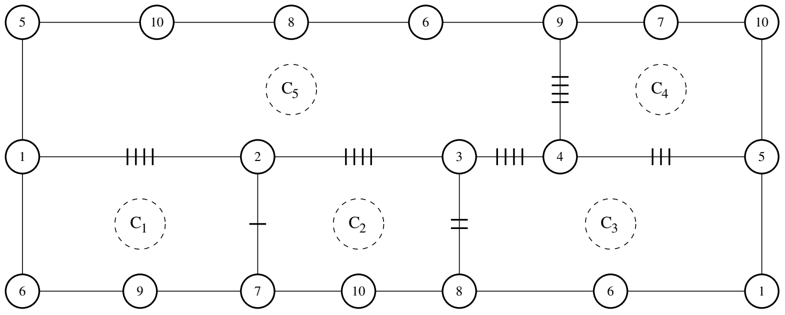


Figure 6.9: The final fundamental polygon obtained after gluing together the matching edges of the circuits in (6.10). The circuits name of the associated circuit is inscribed in the dashed circle and edges that have been glued together have been marked. The glueing ordering is indicated by the number of marks on the edges.

We have now verified the correspondence between the simplicial structures and graph embeddings. This may yield important insights about a very large class of infeasible ISPR structures. It also provides us with some powerful tools for constructing interesting example instances with some sought properties. The construction of examples is actually an important task; it often develops a much deeper understanding of the problem at hand. Further, it may also be the case that one gets sufficiently aquainted with the problem and can formalize procedures to construct examples. These procedures can then be used to characterize and analyze the subclasses of solutions. This was the case above for the class of simplicial solutions.

6.2 The Simplicial Dependency Graph

To define our dependency graph some preliminaries about the concept of graph duality are first given. The dual graph associated with a graph is well known for planar graphs, but perhaps not for non-planar graphs. Given an undirected graph, $G = (V, E)$, and an embedding, I, of G there is a dual graph, $G_I^* = (V^*, E^*)$. Each face in the embedding of G yields a vertex in G_I^* and two vertices in G_I^* are connected if and only if their correponding faces are adjacent in the embedding I. As an example, the dual of the Petersen graph and the embedding in Figure 6.5 is given in Figure 6.10.

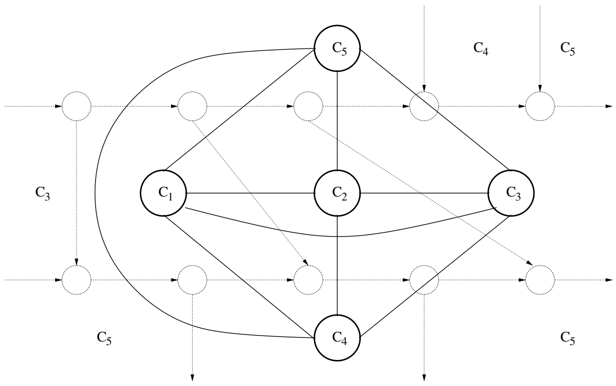


Figure 6.10: *The dual graph associated with the Petersen graph and the embedding in Figure 6.5.*

Remark 6.4. Some remarks on the duality associated with graphs are given here. (1) the dual graph may contain multiple arcs, (2) a graph may be associated with several different dual graphs depending on the embedding, and (3) there is an embedding such that $(G^*)^* = G$.

The dual graph naturally encodes information on the adjacency of cycles in a simplicial cycle family given an embedding, which in a sense corresponds to a dependency relation between the cycles.

6.2.1 A Simplicial Cycle Family Induces a Dependency Graph

An inadequacy with the dual graph is that it does not encode the fundamental information about which arcs are used backwards by which cycles. This may be crucial since it specifies exactly how cycles depend on each other, and not just that there is a dependency. This issue is resolved by introducing the dependency graph induced by a simplicial cycle family. Formally, we have the following definition.

Definition 6.2
Let $\mathcal{C}$ be a simplicial cycle family decomposed into a collection of oriented circuits, $\bar{\mathcal{C}}$. The

dependency graph, $G^*\left(\bar{\mathcal{C}}\right) = (N^*, A^*)$, associated with $\bar{\mathcal{C}}$ is the directed simple graph with node set

$$N^* = \bar{\mathcal{C}} \tag{6.14}$$

and arc set

$$A^* = \{(C_i, C_j) \in N^* \times N^* \mid (u,v) \in C_i \cap F \textit{ and } (v,u) \in C_i \cap B\}. \tag{6.15}$$

In words, the dependency graph, $G^*\left(\bar{\mathcal{C}}\right)$, associated with the oriented circuit decomposition $\bar{\mathcal{C}}$ is the graph with a node for each circuit and a directed arc (i,j) if and only if the circuit corresponding to node i uses an arc in the circuit corresponding to node j in the forward direction. That is, the existance of the *arc (i,j) means that circuit i depends on circuit j*, so to speak.

The definition of $G^*\left(\bar{\mathcal{C}}\right)$ is clearly closely connected to the dual graph induced by an embedding into a surface. One could say that our construction is simply the projection of the dual graph onto the underlying simple graph where also arcs have been directed in accordance with the dependence between the circuits. That is, our dependency graph is a directed simple version of the dual that does not contain multiple edges or loops.

The following examples clarify how the dependency graph can be constructed. The second example also illustrates how the postprocessing step affect the dependency graph.

Example 6.6

If there is an embedding available it is easy to construct the dependency graph from the associated dual graph. Simply remove multiple edges and then direct the edges to comply with the dependency structure. Using the planar embedding in Figure 6.3 in Example 6.2 yields the dependency graph in Figure 6.11. Note that the arcs from C_1 to C_0 and from C_0 to C_4 corresponds to two edges each in the ordinary dual graph.
The embedding of the Petersen graph in Example 6.3 yields the dual graph in Figure 6.10. Using the orientation in Figure 6.6 we simply have to orient these edges which gives the dependency graph in Figure 6.12.

Example 6.7

The circuit families in Example 6.4 will be used. First, take the circuit family induced by the cellular embedding and assignment on the right in Figure 6.7. This family, call it $\mathcal{C}^{(1)}$, consists of the three circuits

$$\begin{aligned} C^1 &= F^1 \cup B^1 = \{(4,1),\ (1,2)\} \cup \{(4,2)\} \\ C^2 &= F^2 \cup B^2 = \{(3,4),\ (4,2)\} \cup \{(3,2)\} \\ C^3 &= F^3 \cup B^3 = \{(3,2)\} \cup \{(3,4),\ (4,1),\ (1,2)\}. \end{aligned} \tag{6.16}$$

These circuits yield the dependency graph $G^*\left(\mathcal{C}^{(1)}\right)$ that is given on the left in Figure 6.13. Recall the postprocessing step from Example 6.4 and that its purpose is to make

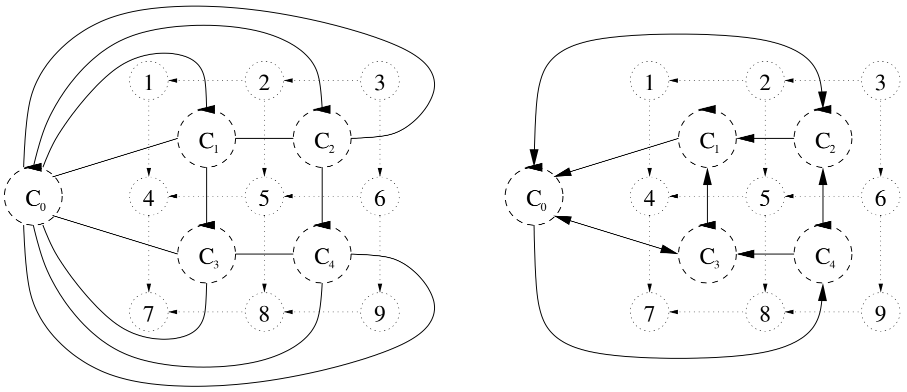


Figure 6.11: *The dual graph (left) and the dependency graph (right) associated with the cycle family in Example 6.2.*

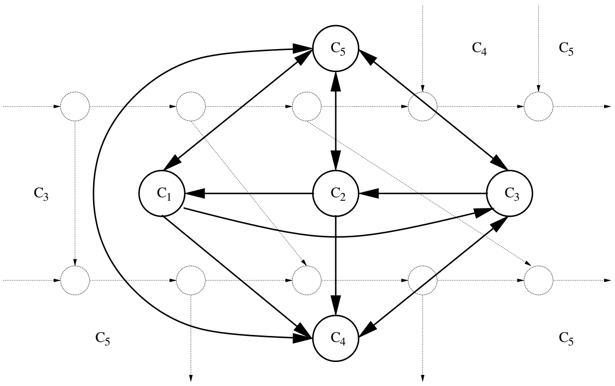


Figure 6.12: *The dependency graph obtained from the cycle family in Example 6.3.*

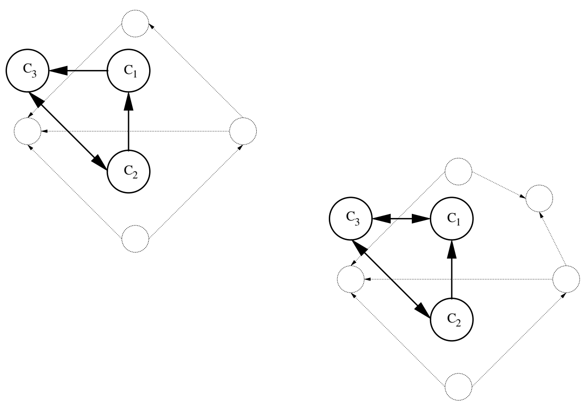


Figure 6.13: *The dependency graphs at intermediate steps of the postprocessing step that make reducible structures irreducible by the subdivision operation. (Left) The dependency graph of the original structure. (Middle) The dependency graph the first (and second) subdivision. (Right) The dual after three subdivisions. Note that a subdivision makes an arc in the dependency graph bi-directed.*

the sudivided arcs correspond to bi-directed arcs in the dependency graph. After the first step, the first circuit, was changed into

$$C^1 = F^1 \cup B^1 = \{(4,a),\ (1,2)\} \cup \{(1,a),\ (4,2)\}. \tag{6.17}$$

The dual of this new circuit family, call it $\mathcal{C}^{(2)}$, is given in the middle in Figure 6.13, where the arc between C_1 and C_2 is now bi-directed, as desired. Next consider, the second postprocessing step from Example 6.4 which yielded a change of the second circuit into

$$C^2 = F^2 \cup B^2 = \{(b,4),\ (4,2)\} \cup \{(b,3),\ (3,2)\}. \tag{6.18}$$

Note that the dual of this new circuit family, $\mathcal{C}^{(3)}$ say, is the same as for $\mathcal{C}^{(2)}$ since the arc between C_2 and C_3 was already bi-directed.

Finally, it is easy to verify that the dual of the circuit family induced be the left graph in Figure 6.8 that arose from the final preprocessing step is the dependency graph given on the right in Figure 6.13.

Remark 6.5. When the embedding is available, the procedure in Example 6.6 may be preferred since it easily can be used to obtain a drawing on a surface without crossings, if desired. Without the embedding, it may be harder to draw the dependency graph "nicely", but it is of course no problem to construct the graph.

There are two applications where the dependency graph is useful. First, the dependency relation can be used when examples are created. More importantly, it turns out that the dependency graph yields a nice characterization of generators and irreducibility for simplicial cycle families, cf Section 6.2.4. Let us now consider the related "dual" problem to construct a simplicial cycle family with a given dependency graph.

6.2.2 A Dependency Graph Induces Simplicial Cycle Family

An operation that transforms a simplicial cycle family to a dependency graph has just been described above. Given an embedding of a connected directed graph, D, a family of oriented circuits from D can be constructed by an inverse operation. That is, an operation that brings nodes back to circuits and assigns forward and backward directions to arcs in the circuits. We formalize this using the following definitions.

Definition 6.3
In a cellular embedding of a graph into an orientable surface, an arc (i, j) that is shared by an ordered pair of faces, (u, v), is consistent w.r.t. (u, v) if it is traversed clockwise by the boundary of u and counter-clockwise by the boundary of v.

Note that an arc is consistent with exactly one of the two ordered pair of adjacent faces and no other pair of faces.

Definition 6.4
Let $D = (N, A)$ be a simple connected directed graph and I a cellular embedding of D into an orientable surface. Denote by R the set of faces induced by I. Then, the family of oriented circuits, $\mathcal{C}_I^* (D)$, associated with D is defined by

$$\mathcal{C}_I^* (D) = \{C_i = F_i \cup B_i\}_{i \in N} \tag{6.19}$$

where

$$F_i = \{(u, v) \in R \times R \mid \textit{Some arc } (i, j) \in A \textit{ is consistent w.r.t } (u, v)\}, \tag{6.20}$$

and

$$B_i = \{(u, v) \in R \times R \mid \textit{Some arc } (j, i) \in A \textit{ is consistent w.r.t } (u, v)\}. \tag{6.21}$$

Intuitively, $D = (N, A)$ is a dependency graph, $G^* \left(\bar{\mathcal{C}}\right)$, obtained from some collection of oriented circuits, $\bar{\mathcal{C}}$ say, in a graph $G = (R, A_{\mathcal{C}})$. To obtain D as a dependency graph, there must be a circuit C_i in $\bar{\mathcal{C}}$ for each node $i \in N$. An arc, (i, j) in D means that C_i depends on C_j. That is, there is an arc, $(u, v) \in A_{\mathcal{C}}$ say, such that $(u, v) \in F_i$ and $(u, v) \in B_j$.

There are essentially only three cases that have to be dealt with in this definition when the forward and backward arcs are to be decided for a given circuit. They are all covered by node 1 in the template of the partial embedding in Figure 6.14.

First considered arc $(1, 2) \in A$. For the north and east faces, the arc $(1, 2)$ is consistent with the face pair $(E, N) \in R \times R$ since it traverses the E face clockwise and the N face counter-clockwise. Therefore, $(E, N) \in F_1$ according to (6.20) (and also $(E, N) \in B_2$ according to (6.21)). Next, arc $(3, 1) \in A$ is traversed clockwise by the east face and counter-clockwise by the south face, therefore $(E, S) \in B_1$ (and $(E, S) \in F_3$). Finally, the arcs $(1, 4)$ and $(4, 1)$ similarily yield that $(N, W) \in F_1$ and $(S, W) \in B_1$.

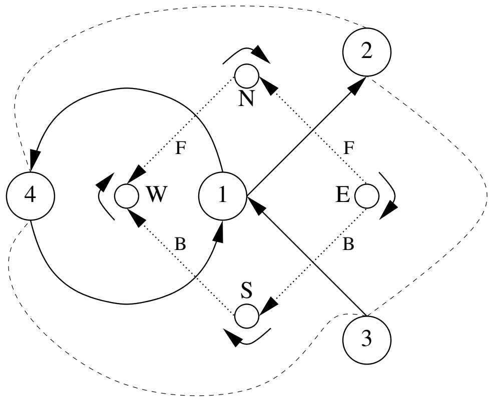


Figure 6.14: *Part of a cellular embedding of a graph and the associated part of its dual family of oriented circuits. The large nodes are the nodes of the graph and the solid arrows its arcs. The small circles correspond to the faces of the embedding and the dotted arcs denote the resulting arcs in the associated family of circuits. The markings F and B indicate whether it is a forward or a backward arc.*

In practice, the easiest way to create a cycle family from an embedding of a directed graph, D, is to first construct its dual. Then, for each node i in D, traverse the corresponding face in the dual clockwise. When an arc (i, j) in D is encountered, that is, it intersects the face edge (u, v) in the dual, 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 6.8

Consider the right dependency graph in Figure 6.13 and apply the procedure associated with Definition 6.4 to obtain a cycle family. At first, one may expect to obtain the family of cycles denoted by $\mathcal{C}^{(3)}$ in Example 6.7 since we are esentially applying the dual to the dual, so to speak.

There is always an ambiguity w.r.t. the embedding if there are bidirected arcs in the dependency graph. Suppose that we choose the embedding on the left in Figure 6.15 where also the dual has been sketched. Applying the procedure above to this embedding, using the dual, yields the graph and orientation on the right in Figure 6.15 which actually coincides with the cycle family induced by the right graph in Figure 6.8.

Let us also see what happens if another emmbedding is used. Assume that the arcs (C_1, C_2) and(C_2, C_1) are reversed. This does not change the dual graph, but it does alter which arcs that become forward and backward arcs for the two cycles C_1 and C_2. It is easy to verify that the cycle family becomes as in Figure 6.16. It is not possible to

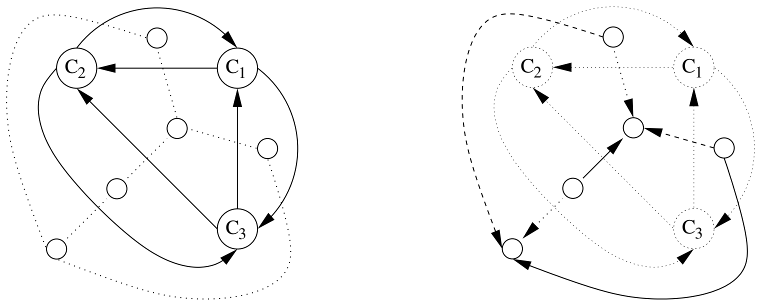


Figure 6.15: *(Left) An embedding of a directed graph that is the dependency graph from Example 6.7 and its dual graph. (Right) The associated cycle family obtained from the embedding.*

obtain a cycle family that is structurally different from the ones in Figure 6.8 and 6.16. Which one that is actually obtained depends on the embedding.

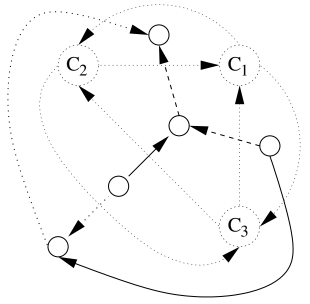


Figure 6.16: *A different embedding of the graph in Figure 6.15 yields another cycle family.*

From the above example it is clear that there are dependency graphs and embeddings such that the cycle family that yields the dependency graph is recovered. For instance, the first embedding considered was also the dependency graph of the cycle family that we obtained. It is however not certain that the same cycle family is retrieved. For some cycle families, this is simply not possible, because it contains some "structurally uninteresting" contents that is projected away in the process, as in the case with $\mathcal{C}^{(3)}$ above.

In conclusion, the procedures defined above are not really dual operations. Given a cycle family some information is lost (projected) when the dependency graph is constructed. It is not possible to retrieve ("unproject") it from this simple graph to the multigraph that is required to recover the original cycle family. However, we see no reason why this would even be desirable since the essential structure is left untouched.

Hopefully it is evident by now that the two operations introduced in this section yield powerful tools to construct simplicial structures with certain sought properties.

6.2.3 The Equivalence Class Induced by a Dependency Graph

Let us just elaborate some more on the duality aspect. A minor drawback with the "projection" onto the underlying simple directed graph was revealed above; namely, it may be the case that

$$\mathcal{C} \neq \mathcal{C}_I^* \left(G^* \left(\mathcal{C}\right)\right) \quad \text{or} \quad D \neq G^* \left(\mathcal{C}_I^* \left(D\right)\right). \tag{6.22}$$

This is a price we have to pay when multigraphs are not used.

It is in fact easy to see that all circuit families that yield the same dependency graph, D say, forms an equivalence class. Also, any embedding of D yields a circuit family whose dependency graph is D. Given a circuit family, $\mathcal{C}$, this yields

$$\mathcal{C} \equiv \mathcal{C}_I^* \left(G^* \left(\mathcal{C}\right)\right) \equiv \mathcal{C}_I^* \left(G^* \left(\mathcal{C}_I^* \left(G^* \left(\mathcal{C}\right)\right)\right)\right) \tag{6.23}$$

and the corresponding statement for the dependency graph

$$G^* \left(\mathcal{C}\right) = G^* \left(\mathcal{C}_I^* \left(G^* \left(\mathcal{C}\right)\right)\right). \tag{6.24}$$

Considering the equivalence class, or rather their common dependency graph, may be a good idea when subclasses of structures are classified and analyzed. It is for instance possible to develop specialized algorithms that find simplicial solutions for some equivalence classes where the dependency graphs have some special structure.

In particular, all valid cycles correspond to the simple directed cycle with only two nodes and two arcs. An algorithm to find these structures were discussed in Section 5.4.3. More complicated classes and associated algorithms can also be derived, e.g. in a recent project (cf. [73]) polynomial algorithms were derived to find any simplicial solution that yield a dependency graph that is a path or a tree, respectively, if a single node is removed. Other structured classes can also be found in polynomial time with similar techniques.

6.2.4 Characterization of Irreducible Generators

The most important property of the dependency graph is that it can be used to characterize generators and irreducible solutions. In this section it is assumed that the lineality space contains only the origin, a remark about the general case is made after the first proof.

Proposition 6.1
Assume that the lineality space is $\{0\}$ and let θ be a canonical saturating simplicial solution to (PR-Farkas) *represented by the canonical circuit decomposition $\mathcal{C}$. Then, θ is an extremal solution if and only if the associated dependency graph $G^* \left(\mathcal{C}\right)$ contains a single strongly connected component.*

Proof: Assume that the dependency graph $D = G^* (\mathcal{C})$ consists of two strongly connected components D_1 and D_2. Trivially, θ can be composed by two solutions θ_1 and θ_2 that yield D_1 and D_2 as dependency graphs.

Now assume that the dependency graph $D = G^* (\mathcal{C})$ is strongly connected. Clearly, θ is an extremal solution if and only if there exist no ordering of the circuits in $\mathcal{C}$ such that a strict subsequence induces a solution θ_1. This can be tested by "non-deterministically" removing one circuit at a time. If all circuits are removed (in all orderings), then θ must be an extremal solution.

Since circuits correspond to nodes in the dependency graph, this is equivalent to removing a single node at a time. If this process at some time encounters a situation where the reduced graph is strongly connected, a cycle family that corresponds to the solution θ_1 has been removed and θ is not extremal. Hence, for θ to be extremal, the graph must be strongly connected. □

Remark 6.6. When the lineality space is not the origin, the characterization above yields generators if one of the solutions associated with D_1 or D_2 is in the lineality space.

Undoubtly, in the design context the most imporant extremal solutions are the irreducible ones. We have the following characterization of irreducible extremal solutions.

Theorem 6.2
Let θ be a canonical saturating simplicial solution that is a generator of (PR-Farkas) *represented by the canonical circuit decomposition $\mathcal{C}$ and denote the associated dependency graph by $D = G^* (\mathcal{C})$. Denote by N_l the nodes in D that correspond to circuits associated with destination, l, and denote by D_l the graph induced by the removal of N_l from D. Then, θ is irreducible if and only if D_l is strongly connected for all l.*

Proof: Assume that D_l is not strongly connected. Let D_1 be a component of D_l that has no arc that leaves the component. Such a component exists since D_l is not strongly connected. Now consider the graph H induced by D_1 and N_l, that is, put back the nodes and appropriate arcs removed to obtaind D_l. By construction there is a θ_1 with an associated circuit family $\mathcal{C}_1$ such that the dependency graph $G^* (\mathcal{C}_1)$ is H. Further, θ_1 can be choosen such that all its associated backward arcs are backward arcs also for θ. That is, θ is reducible.

To prove the other direction we consider a non-deterministic algorithm approach similar to the one in the above proof. The following observation yields a characterization. If there for every l holds that there exist no ordering of the circuits in $\mathcal{C}$ such that a strict subsequence together with the backward arcs induced by l induces a solution, then θ is irreducible.

This is again tested by non-deterministically first selecting an l and removing all associated circuits and then removing one circuit at a time. If all circuits are removed (in all orderings), then θ is not reducible.

Since circuits correspond to nodes in the dependency graph, this is equivalent to first removing all nodes corresponding to an l. Then, a single node is removed at a time. If this process finds a graph that is strongly connected, then θ, is reducible; a cycle family can be created from this component with some backward arcs associated with l. Hence, for θ to be extremal, the graph D_l must be strongly connected for all l. □

Let us illustrate the latter theorem with some examples.

Example 6.9

Recall the cycle family constructed in Example 6.2. It was pointed out in Remark 6.1 that this cycle family is not irreducible. Using Theorem 6.2 this can easily be seen in the associated dependency graph in Figure 6.11 on page 118. If node C_0 is removed, the node C_1 becomes a strongly connected component 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

$$C = \{(1,4),\ (2,1)\} \cup \{(2,5),\ (5,4)\}. \tag{6.25}$$

Hence, the original cycle family was reducible as previously stated.

It is also straightforward to verify that the cycle family constructed from the Petersen graph in Example 6.2 is irreducible. We can also verify this using Theorem 6.2. The associated dependency graph was given in Figure 6.12 on page 118. Clearly, the removal of a single node always leaves a strongly connected graph in this case, and the irreducibility conclusion follows.

Example 6.10

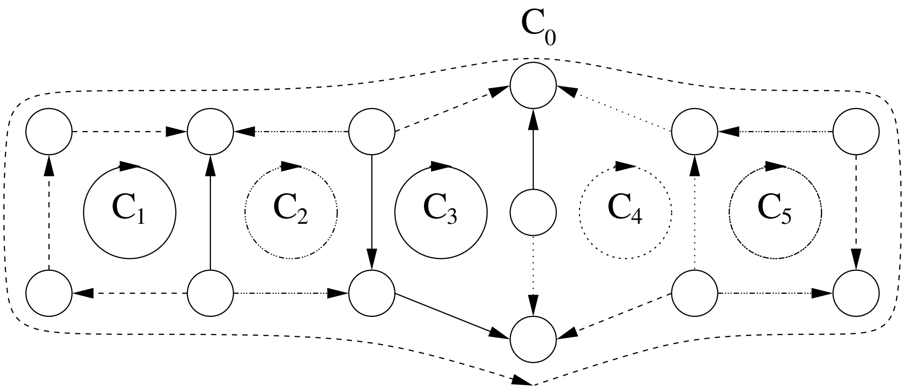


Figure 6.17: The cycle family induced by the orientation and embedding of this graph is not irreducible. The arcs in this instance can be used to form six conflicts in total. Out of these, three are irreducible.

Consider the cycle family induced by the orientation and embedding of the graph in Figure 6.17. 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 not irreducible. It is actually possible to find no less than 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 6.18.

First, consider the removal of the solid nodes associated with cycle 1 and 3. This leaves cycle 2 as a strongly connected component without emanating arcs. If the backward arcs from cycle 1 and 3 are combined with the backward arcs in cycle 2 valid cycle with arcs spanned by cycle 2 is obtained.

Now remove the node associated with cycle 0. This leaves the component with cycle 1 without emanating arcs. 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 component with cycles 3 and 4 is considered, which is feasible since it does not contain an emanating arc, another conflict is obtained. The backward arcs in cycle 3 and 4 combined with some arcs in cycle 0 yields an irreducible solution.

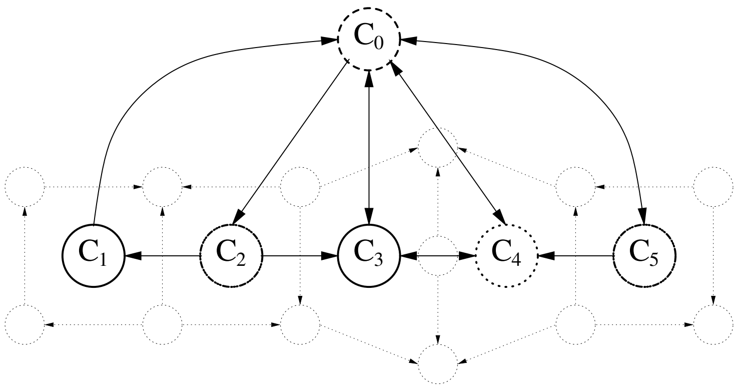


Figure 6.18: *The dependency graph associated with the cycle family in Figure 6.17.*

It is clear that the conditions in the two theorems above yields a good characterization since that can be verified efficiently. However, we have not provided an algorithm to extract some or all the irreducible conflicts. It should not be to hard to exactly characterize all irreducible conflicts contained in a reducible conflict using multigraphs. This would also be very desirable in the design context where a reducible conflict should preferably be dissected into irreducible parts, then a minimal conflict or a most violated irreducible conflict can be selected.

Remark 6.7. We have not used multigraphs in our presentation above. This may have been a mistake since it introduces some ambiguity and also complicates some tasks. It is easier to be clear and precise with multigraphs since an embedding of a multigraph naturally encode all necessary information. The advantage with the projection onto the simple graph is that it only captures the essential information about the structure. This is desirable for some tasks, such as classification.

Let us now turn to the issue of establishing that all solution classes defined in Section 5.2 are actually relevant. That is, there exist generators in each of these classes and that there is an instance where the least complicated irreducible generator belongs to each of these classes.

6.3 The Hierarchy of Infeasible Structures

Recall from Section 5.2 that we have considered 4 classes of solutions. They are, in decreasing order of generality, referred to as: general, binary, unitary and simplicial solutions.

It remains to prove that all these subclasses are proper subclasses. For each relation, it suffice to provide a single example of an instance where the least complicated conflict belongs to the class of solution. It may be easier follow the examples if they are presented in increasing order of complexity of the structure.

So far, several example instances with valid cycles and simplicial cycle families without valid cycles have been given. Therefore, it suffice to begin with a unitary solution that is not simplicial. This is done in Example 6.11. It becomes more complicated in Example 6.12 where a binary solution is described and no unitary solution exist. Finally, a general solution that is not binary is provided in the large Example 6.13.

Remark 6.8. In all of the examples below there is not a connection between the index of the SP-graph and the destination unless that 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 the examples larger and force them to contain arcs that do not really matter for the structures studied. Note that this impose no loss of generality since 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.

To show that all unitary solutions are not simplicial we use an example due to Bley, cf. page 77 in [9]. This example is in our opinion remarkably beautiful; it is very small, symmetric, planar, simple and yet rather complex.

Example 6.11

A straightforward translation of the path based example in Figure 5.4 on page 77 in [9] to SP-graphs yields the 4 SP-graphs in Figure 6.19. The SP-arcs for these graphs are

$$\begin{aligned} A^1 &= \{(4,3),\ (3,1)\}, & A^2 &= \{(3,4),\ (4,2)\}, \\ A^3 &= \{(2,1),\ (1,3)\}, & A^4 &= \{(1,2),\ (424)\}, \end{aligned} \tag{6.26}$$

which yields four different oriented circuits with the same underlying undirected cycle $(1\ 2\ 4\ 3)$. All four oriented circuits are constructed uniquely by choosing the associated SP-arcs as backward arcs and then completing the circuits by forward arcs from two other SP-graphs.

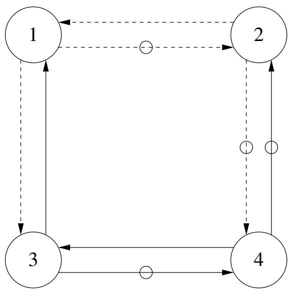


Figure 6.19: *The SP-arcs of four SP-graphs with destination 1 through 4. The solid arcs with and without a circle marking are associated with destination 1 and 2, respectively. A dashed arc is associated with destination 3 if it does not have a circle marking and with destination 4 if it does. The four uniquely induced circuits formed when both SP-arcs for a certain destination are used as backward arcs yields a feasible family of cycles.*

Let us now show that all binary solutions are not unitary. To accompish 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 6.12

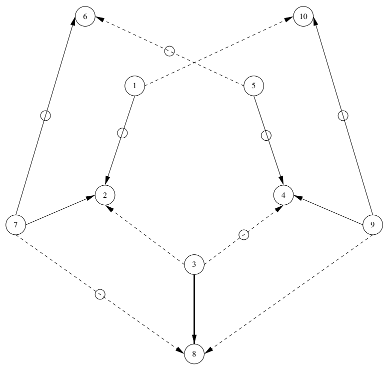


Figure 6.20: *A graph and a family of SP-graphs where the induced binary solution is not unitary. The SP-arcs associated with commodity 1 through 4 are given by solid arcs, solid circle marked arcs, dashed arcs and dashed circle marked arcs, respectively. A binary solution is obtained if all five feasible circuits are used. In this solution, the thickened arc* $(3, 8)$ *carries two units of flow in both direction.*

The graph in Figure 6.20 and the following four SP-graphs with SP-arc sets given by

$$\begin{aligned} A^1 &= \{(3,8),\ (7,2),\ (9,4)\}, \\ A^2 &= \{(1,2),\ (5,4),\ (7,6),\ (9,10)\}, \\ A^3 &= \{(1,10),\ (3,2),\ (9,8)\}, \\ A^4 &= \{(3,4),\ (5,6),\ (7,8)\}. \end{aligned} \tag{6.27}$$

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 very easy to deduce which cycles are possible for the different SP-graphs. In Figure 6.21, the graph have been embedded into the Möbius strip. From this drawing it is very easy to verify that the only circuits that obey the commodity specific flow bounds are the one sketched out. The five circuits are

$$\begin{aligned} C_1^1 &= B_1^1 \cup F_1^1 = \{(3,8),\ (7,2)\} \cup \{(3,2),\ (7,8)\}, \\ C_1^2 &= B_1^2 \cup F_1^2 = \{(3,8),\ (9,4)\} \cup \{(3,4),\ (9,8)\}, \\ C_2 &= B_2 \cup F_2 = A^2 \cup \{(7,2),\ (9,4),\ (1,10),\ (5,6)\}, \\ C_3 &= B_3 \cup F_3 = A^3 \cup \{(3,8),\ (9,10),\ (1,2)\}, \\ C_4 &= B_4 \cup F_4 = A^4 \cup \{(3,8),\ (7,6),\ (5,4)\}. \end{aligned} \tag{6.28}$$

Clearly, the family of cycles consisting of all these circuits is binary. Further, in the associated canonical solution, 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.

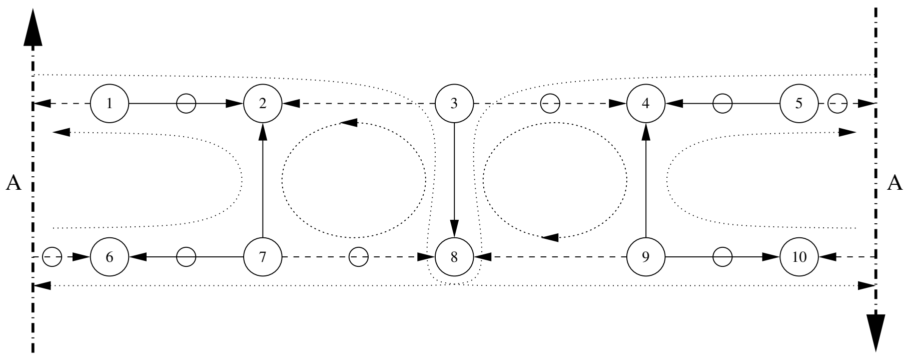


Figure 6.21: *An embedding of the graph in Figure 6.21 in the Möbius strip. To obtain the surface, the sides labeled with A are glued together so that the arrows overlap. All oriented circuits, C_1^1, C_1^2, C_2, C_3 and C_4 have been sketched out with a dotted circle with an orientation that is consistent with the associated commodity.*

We mentioned the similarity of the graph in Figure 6.21 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, more often referred to as the complete bipartite graph with three nodes in each partition, $K_{3,3}$. 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. Clearly, the cycle $(2,\ 7,\ 4,\ 9)$ will induce a valid cycle. To obtain an irreducible solution two subdivisions were made which yielded the above instance.

To complete our 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 (PR-Farkas) may in principle be arbitrarily complicated. To obtain our objective this time, the construction is essentially based on two Möbius strips that are glued together.

Example 6.13

In this "huge" example, the instance induced by the graph in Figure 6.22 is considered. There are six commodities of three kinds. The commodities associated with dotted, square marked dotted, dashed and square marked dashed arcs are similar and will be treated essentially equivalently. These commodities will be referred to as type-1 commodities. The square marked solid arcs is a bit different since it will be used in two circuits with equal flow, call the associated commodity a type-2 commodity. Finally, the commodity associated with purely solid arcs is what will make the solution in this example general instead of "just" binary. It will be associated with two circuits with *different flow amount*.

First, it will be verified that this instance indeed yields a general solution. This is rather straightforward. Then, we must also show that there is no other solution in the instance. This will, perhaps surprisingly, not be to complicated; we have actually created an "unnecessarily large" instance via some subdivisions to reduce the number of possible cycles. This implies that it is much easier to verify the uniqueness of the proposed solution.

From the graph it is easily verified that for each type-1 commodity there is a unique oriented circuit that uses all the commodity specific arcs backwards. The only possible cycle for the commodity associated with dashed arcs is given in Figure 6.23. The three other type-1 commodities are obtained by symmetry.

Equivalently, there is a unique oriented circuit that uses the upper square marked solid arcs backwards, it is sketched in Figure 6.23. The lower circuit is obtained by symmetry.

Only one commodity remains, the one associated with the solid arcs. In this case there are several 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 6.24.

It is now straightforward to verify 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.

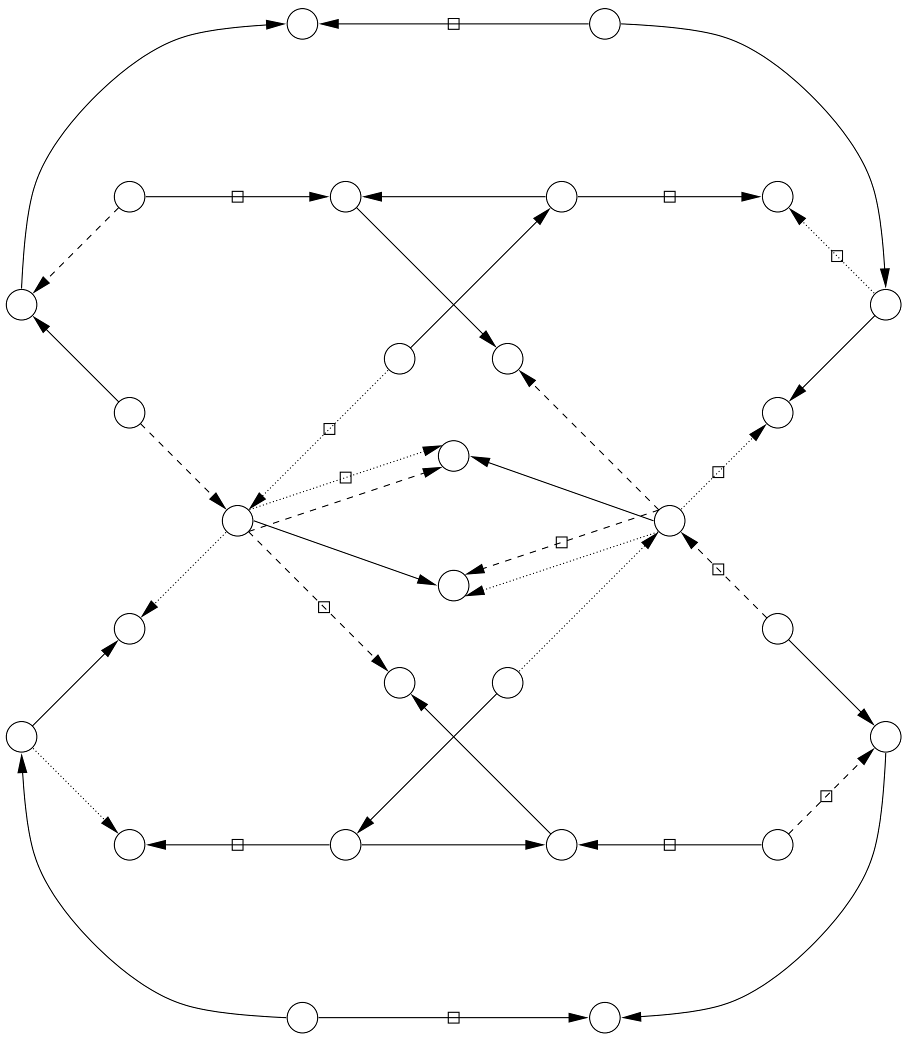


Figure 6.22: *An instance with six commodities that induce a unique cycle family that is not binary.*

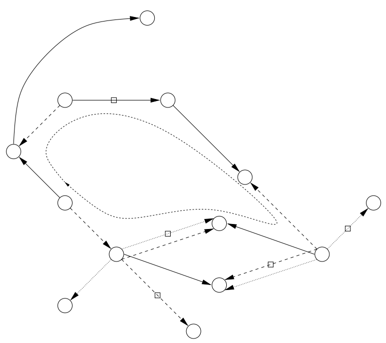
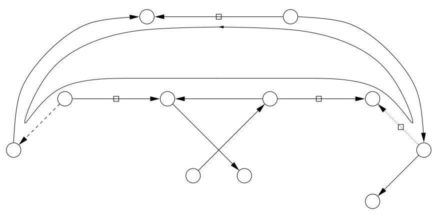



Figure 6.23: *(Left) The part of the graph where oriented circuits that use dashed arcs backwards could possibly exist. The unique oriented circuit is sketched. (Right) The upper part of the graph where oriented circuits that use square marked solid arcs backwards could possibly exist. The unique upper oriented circuit is sketched.*

By construction, all arcs not in the inner circuit carry one forward and one backward unit of flow. In the inner circuit, the type-1 commodities carry one flow unit backwards on its own arc and forwards on the associated 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. That is, it is all or nothing and the proposed solution is unique.

This enables us to draw the conclusions that we have been striving for. Recall the notation. The collections of all cycle families that correspond to an irreducible generator are denoted by:

- $\mathcal{S}^2$ for simplicial solutions that involve at most two destinations,
- $\mathcal{S}$ for simplicial solutions,
- $\mathcal{U}$ for unitary solutions,
- $\mathcal{B}$ for binary solutions,
- $\mathcal{G}$ for general solutions.

Also, the subscript I is used on these collections to denote the collection of instances where the least complicated solution belongs to the corresponding collection. The following theorems were proposed in Chapter 5.

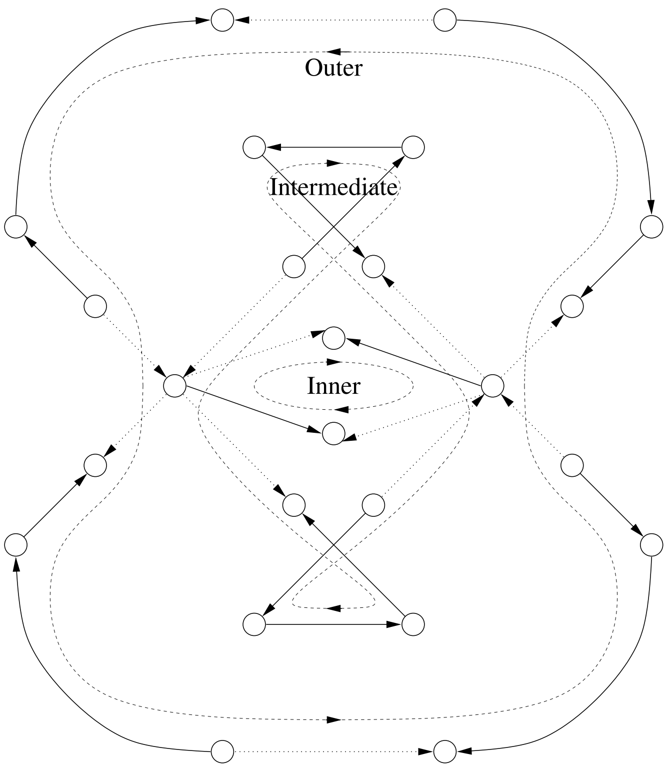


Figure 6.24: *Three dashed "basis" cycles that represents all possible oriented circuits that use solid arcs backwards.*

Theorem 6.3 (Theorem 5.7 in Chapter 5)
The relation between the collections of all cycle families that correspond to an irreducible generator of a certain type is as follows,

$$\mathcal{S}^2 \subset \mathcal{S} \subset \mathcal{U} \subset \mathcal{B} \subset \mathcal{G}. \tag{6.29}$$

Proof: Recall that inclusion holds trivially from the definitions in Section 5.2 in Chapter 5 and that it only remains to prove that the inclusions are strict. This is acheieved by Example 5.3 in Chapter 5 and Example 6.1 and 6.11 through 6.13 in this chapter. □

Equivalently for collections of instances, we have the following.

Theorem 6.4 (Theorem 5.8 in Chapter 5)
The relation between the collection of instances where the least complicated solution is of a certain type is as follows,

$$\mathcal{S}_I^2 \subset \mathcal{S}_I \subset \mathcal{U}_I \subset \mathcal{B}_I \subset \mathcal{G}_I. \tag{6.30}$$

Proof: Inclusion again holds trivially. By construction, all examples referred to in the above proof were created to have unique solutions. □

In [23], a class of solutions called 3-valid cycles is defined. This class is the subset of the general solutions where all x associated with the same destination have the same value. They asked whether the absence of 3-valid cycles is a sufficient condition for feasibility of (PR). The answer is no.

Corollary 6.1
Absence of 3-valid cycles is not in general a sufficient condition for the feasibility of (PR).

Proof: The instance in Example 6.13 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 solid unmarked commodity. □

This completes our analysis and characterization of infeasible routing patterns. As we have seen it is possible to specialize algorithms and results for the classes of solutions. This may be an interesting direction for further research, in particular, the class of simplicial solutions deserves more attention.

We will return to the above structures in Chapter 8 when valid inequalities and specialized separation algorithms are derived. To establish the validity of the inequalities one can interpret the set of feasible routing patterns via the conflict hypergraph.

7 Cycle Bases: Models and Methods

THE multicommodity circulation structure of (PR-Farkas) motivates a modelling approach based on circulations. A straightforward model can be obtained by enumerating all cycles that satisfy the commodity specific flow bounds. In such a model, the variables correspond to the amount of flow sent in the cycles and a solution is feasible if the induced flow fulfills the aggregated capacity constraints, cf. model P3 in [24]. An obvious drawback with this naive approach is that the number of cycles is in general exponential in the size of the graph, therefore such a formulation could be less attractive from a computational perspective. This may to some extent be resolved by column generation. A theoretical drawback with an enumeration model is that there is not a one-to-one correspondance between its solutions and the solutions to (PR-Farkas). In some cases, this may be severe. Our conclusion is that the enumeration approach does not serve our purpose. To overcome the drawbacks and maintain some of the structural advantages from a circulation based modelling approach fundamental cycle bases are used.

An outline of the chapter is as follows. Since it seem to be rather uncommon to model network flow and circulation problems with cycle bases (as far as we know) an introduction is given in Section 7.1. The terminology is explained and it is shown how fundamental cycle bases can be used to model minimum cost flow and multicommodity minimum cost flow problems. In Section 7.2 the same approach as in the multicommodity case is then used to develop cycle basis models for the multicommodity circulation Farkas systems (PR-Farkas) and (PC-Farkas). The cycle basis model of (PR-Farkas) is further analyzed in Section 7.3 where it is shown that some constraints are (under very general conditions) redundant. The Farkas system of the cycle basis model is derived in Section 7.4. It is naturally interpreted as a path based formulation of the inverse partial shortest path problem (IPSPR) with only exponentailly many path constraints. To illustrate how the cycle basis model is constructed in practice some numerical examples are given in Section 7.5. The chapter is concluded with Section 7.6 about computations with cycle basis matrices.

7.1 Modelling Circulation Problems with Cycle Bases

Almost all network flow problems can be modelled with circulations. Given an understanding of cycle bases and the relation between flow in cycles and flow on arcs it is straightforward to develop mathematical models for several common network flow problems. Some necessary definitions and standard results about the cycle space and fundamental cycle bases are introduced here. They may for instance be found in [71], the recent survey article [67] or the text books [7, 21].

7.1.1 Oriented Circuits, Circulations and Cycle Bases

Let $G = (N, A)$ be a strongly connected directed graph and M_G its incidence matrix. An oriented circuit, $C = F \cup B$, 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 yield 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

$$\gamma_{ij}^C = \begin{cases} 1 & (i,j) \in F \\ -1 & (i,j) \in B \\ 0 & (i,j) \notin C. \end{cases}$$

The cycle space, $\mathcal{C}_G \subset \mathbb{R}^A$, of G is the vector space generated by the incidence vectors of all oriented circuits of G and a circulation is a point in the cycle space. The cycle space can also be defined as the null space of the incidence matrix of G, which has rank $n-1$ when G is strongly connected. Hence, Sylvesters law of nullity yields the well known fact that the dimension of $\mathcal{C}_G$ equals the cyclomatic number of G. That is,

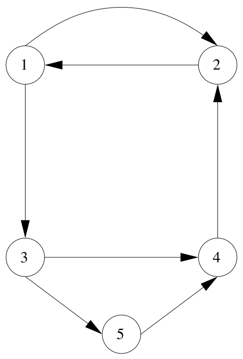
$$\dim \mathcal{C}_G = m - n + 1. \tag{7.1}$$

A cycle basis is a set of $m-n+1$ oriented circuits whose incidence vectors form a basis of $\mathcal{C}_G$ and the associated cycle matrix is the matrix formed by these incidence vectors. It is clear that the cycle space may also be generated by the enumeration approach above, but this is not desirable because of the ambiguity that is introduced. A small example should clarify the above definitions.

Example 7.1

The graph in Figure 7.1 has 5 nodes and 7 arcs, hence the dimension of its cycle space is $7-5+1=3$. Since the incidence vectors in the table at the right constitute 3 linearly independent circulations, they form a basis for the cycle space.

There are several classes of cycle bases, but for our pursposes it is sufficient to consider fundamental cycle bases. As early as 1847, Kirschoff presented a very elegant method to construct a basis for the cycle space in [68], cf. [8]. The simple idea is to use a spanning tree and the oriented circuits induced by the arcs not in the tree.



	C_1	C_2	C_3
12	1	1	
13		-1	1
21	1		1
34		-1	
35			1
42		-1	1
54			1

Figure 7.1: *A graph and a basis for its cycle space.*

7.1.2 Fundamental Cycle Bases

Let T be the arcs of a spanning tree in G. Given an arc $(s,t) \in A \setminus T$, the fundamental cycle, C^T_{st}, in G w.r.t. T is the unique cycle in the graph induced by $T \cup \{(s,t)\}$. The orientation of C^T_{st} is determined by the single arc not in the tree, (s,t). Denote the set of all fundamental cycles by $\mathcal{C}(T) = \{C^T_{(s,t)} : (s,t) \in A \setminus T\}$. It is called a fundamental cycle basis of G w.r.t. T. Clearly, a fundamental cycle basis is a cycle basis, cf. the cycle matrix partitioning in (7.2).

Denote the cycle matrix that is formed by the incidence vectors associated with the fundamental cycles w.r.t. T by Γ_T. The column associated with C^T_{st} in Γ_T is completely determined by the arc, (s,t), not in the tree. Further, the only nonzero entries of the column are the 1 in the row associated with (s,t) and the entries in the rows associated with the arcs in the tree. Hence, the matrix $\Gamma_T \in \mathbb{Z}^{A \times A\setminus T}$ forms a basis of the cycle space and can be partitioned as

$$\Gamma_T = \begin{pmatrix} I \\ \bar{\Gamma}_T \end{pmatrix}, \tag{7.2}$$

where the identity matrix is of order $m - n + 1$ and $\bar{\Gamma}_T \in \mathbb{Z}^{T \times A\setminus T}$. The rows that form the identity correspond to the arcs outside the tree and the rows that form $\bar{\Gamma}_T$ correspond to the arcs in the tree.

Since G is strongly connected its incidence matrix, M_G, has rank $n-1$. Therefore, a single row is removed from M_G and the remaing matrix is partitioned as

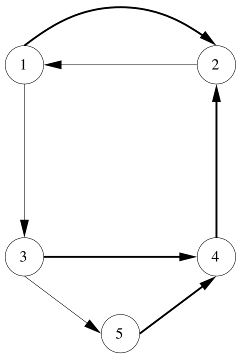
$$(M_T, M_N)\,. \tag{7.3}$$

The submatrices M_T and M_N contain the columns corresponding to the arcs in the tree and not in the tree, respectively. This partitioning yields that M_T is invertible and also that

$$\bar{\Gamma}_T = -M_T^{-1} M_N. \tag{7.4}$$

Example 7.2

The cycle basis in Example 7.1 is clearly not fundamental since the cycle matrix does not contain an identity matrix of order 3 as a submatrix. To obtain a fundamental cycle basis for the graph in Figure 7.1, choose a tree and use the induced fundamental cycles. The tree marked with thick arcs in Figure 7.2 yields a fundamental cycle basis whose cycle matrix is given on the right in Figure 7.2. Note that the rows associated with the non-tree arcs $(1,3)$, $(2,1)$ and $(3,5)$ forms an identity matrix of order 3.



	C_{13}	C_{21}	C_{35}
12	**-1**	**1**	
13	1		
21		1	
34	**1**		**-1**
35			1
42	**1**		
54			**1**

Figure 7.2: *A graph and a fundamental cycle basis for its cycle space.*

7.1.3 Modelling Circulations with Cycle Bases

Before any actual models are given it is illustrated how cycle bases can be used to model circulations. Denote the amount of flow sent in the fundamental cycle C^{st} by x_{st} and let $\theta \in \mathcal{C}_G$ be the circulation induced by $x \in \mathbb{R}^{A \setminus T}$. This yields the following bijective relation between x and θ,

$$\theta_{ij} = \sum_{(s,t) \in A \setminus T} \gamma^{st}_{C_{ij}} x_{st}, \quad (i,j) \in A, \tag{7.5}$$

or equivalently, using the cycle matrix,

$$\theta = \Gamma_T x. \tag{7.6}$$

It is clear that (7.5) and (7.6) defines a bijection since Γ_T has full column rank, cf. the partitioning (7.2). Also note that

$$\theta_{ij} = x_{ij}, \quad (i,j) \in A \setminus T, \tag{7.7}$$

because of the identity matrix in the partitioning in (7.2).

Given this 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.

7.1.4 The Minimum Cost Circulating Flow Problem

The minimum cost flow problem is a fundamental network flow problem. It is no restriction to consider the circulation version of the problem, it may in fact even be seen as a generalization; the sources and sinks can be handled by adding a super source and a super sink and a set of additional arcs. Then, the possibility to bound the flow on these new arcs also handles interval sources and sinks.

The problem formulation of the minimum cost flow problem is as follows. Given a strongly connected graph ($G = (N, A)$), arc costs (c_{ij}), lower (l_{ij}) and upper (u_{ij}) bounds, find a minimum cost circulation that satisfies the flow bounds. Let θ_{ij} be the amount of flow on arc $(i, j) \in A$, then the standard mathematical model is

$$\begin{array}{llll} \min & \sum_{(i,j)\in A_l} c_{ij}\theta_{ij} & & \\ s.t. & \sum_{j:(i,j)\in A} \theta_{ij} - \sum_{j:(j,i)\in A} \theta_{ij} = 0 & i \in N & \\ & l_{ij} \le \theta_{ij} \le u_{ij} & (i,j) \in A. & \end{array} \qquad \text{(MCCF)}$$

From this model and the relation in (7.6) it is easy to derive a cycle basis formulation. Let T be a spanning tree of G and add the fundamental cycle flow variables, x_{st}, obtained from T, where $(s, t) \in A \setminus T$. Now the relation $\theta = \Gamma_T x$ implies that all circulation constraints are automatically satisfied. The cost of sending one unit of flow along the fundamental cycle C_{st}^T is

$$\sum_{(i,j)\in A} \gamma_{ij}^{C_{st}} c_{ij}. \qquad (7.8)$$

To complete the model it is sufficient to model the flow bounds. This yields a cycle basis formulation of the minimum cost circulating flow problem.

$$\begin{array}{ll} \min & c'\Gamma_T x \\ s.t. & l \le \Gamma_T x \le u. \end{array} \qquad \text{(MCCF-CB)}$$

Let us also consider an alternative derivation of (MCCF-CB) from (MCCF). Divide the θ-variables into tree (basic) and non-tree (non-basic) variables, θ_B and θ_N. Then put the problem in matrix form

$$\begin{array}{ll} \min & z = (c'_B,\ c'_N) \begin{pmatrix} \theta_B \\ \theta_N \end{pmatrix} \\ s.t. & (M_B,\ M_N) \begin{pmatrix} \theta_B \\ \theta_N \end{pmatrix} = 0 \\ & l_T \le \theta_T \le u_T \\ & l_N \le \theta_N \le u_N. \end{array} \qquad \text{(MCCF-M)}$$

The node balance constraint and the relation (7.4) between the incidence and the cycle matrix yields an expression for θ_B in terms of θ_N,

$$0 = (M_B,\ M_N) \begin{pmatrix} \theta_B \\ \theta_N \end{pmatrix} = \left(I,\ M_B^{-1} M_N\right) \begin{pmatrix} \theta_B \\ \theta_N \end{pmatrix}. \tag{7.9}$$

Hence,

$$\theta_B = -M_B^{-1} M_N \theta_N = \bar{\Gamma}_T \theta_N. \tag{7.10}$$

This implies that the objective becomes

$$\begin{aligned} z = (c'_B,\ c'_N) \begin{pmatrix} \theta_B \\ \theta_N \end{pmatrix} &= (c'_B,\ c'_N) \begin{pmatrix} -M_B^{-1} M_N \theta_N \\ \theta_N \end{pmatrix} = \\ &= \left(c'_N - c'_B M_B^{-1} M_N\right) \theta_N = (c'_B,\ c'_N) \begin{pmatrix} \bar{\Gamma}_T \\ I \end{pmatrix} \theta_N = c' \Gamma_T \theta_N. \end{aligned} \tag{7.11}$$

Identifying θ_N and x yields that the cycle basis model, (MCCF-CB), is the representation of (MCCF-M), and (MCCF), in the non-basic, or independent, variables associated with the basis obtained from the spanning tree T. Also note in (7.11) that the objective coefficients, $c'\Gamma_T$, in the cycle basis model is just the the reduced costs w.r.t. the basis for this spanning tree.

Any choice of T is feasible in (MCCF-CB). A particularly interesting choice may be an optimal shortest path tree. Note that a basis corresponding to such an optimal tree is dual feasible in (MCCF-CB) since it is an optimal solution to a relaxation of the problem. Therefore it is a good candidate as a starting basis in the dual simplex method.

7.1.5 Multicommodity Minimum Cost Circulating Flow

The straightforward modelling approach used above can easily be generalized to the multicommodity case. Again it is no restriction to only consider circulations.

A strongly connected graph $G = (N, A)$ and a set of commodities $K \subset N \times N$ are given along with arc costs (c_{ij}^k), individual lower (l_{ij}^k) and upper (u_{ij}^k) bounds for each commodity $k \in K$ and aggregated lower (l_{ij}) and upper (u_{ij}) bounds. The problem is to find a circulation for each commodity that satisfies the individual and aggregated flow bounds such that the total cost is minimized. A standard mathematical model is

$$\begin{array}{llll} \min & \displaystyle\sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^l \theta_{ij}^k & & \\ s.t. & \displaystyle\sum_{j:(i,j) \in A} \theta_{ij}^k - \sum_{j:(j,i) \in A} \theta_{ij}^k \ = \ 0 & i \in N,\ k \in K & \\ & l_{ij} \le \displaystyle\sum_{k \in K} \theta_{ij}^k \le u_{ij} & (i,j) \in A & \qquad \text{(MMCCF)} \\ & l_{ij}^k \le \theta_{ij}^k \le u_{ij}^k & (i,j) \in A,\ k \in K. & \end{array}$$

Apply the above technique again. Let T_k be a spanning tree for each $k \in K$ and let $\theta^k = \Gamma_{T_k} x^k$. This implies that the circulation constraints are satisfied and that the cost of sending one unit of flow along the fundamental cycle induced by $k \in K$ and $(s,t) \in A \setminus T_k$ becomes

$$\sum_{(i,j)\in A} \gamma_{ij}^{C_{st,k}} c_{ij}^k. \tag{7.12}$$

In the multicommodity case, both the individual and aggregated flow bounds have to be modelled. The cycle basis formulation becomes

$$\begin{aligned} \min \quad & \sum_{k\in K} \left(\Gamma_{T_k} c^k\right)' x^k \\ & l \leq \sum_{k\in K} \Gamma_{T_k} x^k \leq u \\ & l_k \leq \Gamma_{T_k} x^k \leq u_k \qquad k \in K. \end{aligned} \tag{MMCCF-CB}$$

An alternative derivation of (MMCCF-CB) from (MMCCF) in the same manner as in the single commodity case is possible.

We are now acquainted with fundamental cycle bases and have seen two examples of how they can be used to model common network flow problems. This approach will now be applied to the multicommodity circulation structured Farkas systems considered in the previous chapter.

7.2 IPSPR Formulations Based on Cycle Bases

There are two cases to consider: partial compatibility and partial realizability. Recall that model (PR) can be used both to determine if a family of generalized SP-graphs is partially compatible and partially realizable. When the cycle basis formulation of the Farkas system of (PR) is derived these cases must however be treated separately. The reason is that there is a reversely spanning l-arborescence in the arc set $A_l \cup \tilde{D}_l$ (cf. Remark 5.4) in the partial realizability case. In the partial compatibility case, $\tilde{D}_l = \emptyset$, and it is no longer guaranteed that there is a reversely spanning l-arborescence. The existence of such an l-arborescence will turn out to be very important. The nice partial realizability case is considered first and then the case where there is not an l-arborescence is treated.

7.2.1 Modelling Partial Realizability with Cycle Bases

The Farkas system of the partial realizability model is repeated here. A family of generalized SP-graphs, $\widetilde{\mathcal{A}}_L$, and the directed multigraph $\tilde{G} = (N, \tilde{A})$ along with its corresponding ordinary graph $G = (N, A)$ is given. For $l \in L$, the generalized SP-graph with destination l is defined by its SP-arcs and non-SP-arcs, $(A_l \cup \tilde{D}_l, \bar{A}_l)$. Without loss of generality assume that $A = \bigcup_{l\in L}(A_l \cup \bar{A}_l)$ and $\tilde{D} = \bigcup_{l\in L} \tilde{D}_l$, where

$$\tilde{D}_l = \left\{(i,l) : i \neq l,\ \delta^+(i) = \emptyset\right\}, \tag{7.13}$$

and $A_l \cap \bar{A}_l = \emptyset$ for all $l \in L$. From the previous chapter we have that $\widetilde{\mathcal{A}}_L$ is not partially realizable if and only if the following model is feasible.

$$\begin{aligned} \sum_{l \in L} \sum_{(i,j) \in \tilde{A} \setminus \bar{A}_l} \theta^l_{ij} &< 0 \\ \sum_{j:(i,j) \in \tilde{A}} \theta^l_{ij} - \sum_{j:(j,i) \in \tilde{A}} \theta^l_{ji} &= 0 \qquad i \in N,\ l \in L \\ \sum_{l \in L} \theta^l_{ij} &\leq 0 \qquad (i,j) \in \tilde{A} \\ \theta^l_{ij} &\geq 0 \qquad (i,j) \in \tilde{A} \setminus \left(A_l \cup \tilde{D}_l\right). \end{aligned} \tag{PR-Farkas}$$

Recall that our aim is to derive a circulation based model for (PR-Farkas) without the drawbacks implied by complete cycle enumeration. This can be accomplished by using fundamental cycle bases. An advantage of the cycle basis approach is that some of the commodity specific flow bounds are handled implicitly by a special choice of trees.

For each $l \in L$, let $T_l \subseteq A_l \cup \tilde{D}_l$ be an arbitrary spanning intree to l such that all arcs in $\tilde{D}_l$ are in T_l, that is, $\tilde{D}_l \subseteq T_l$. By construction of $\tilde{D}_l$, an intree with this property exists since the arc set $A_l \cup \tilde{D}_l$ contains a reversely spanning l-arborescence and all arcs in $\tilde{D}_l$ enters node l, cf. Remark 5.4. Denote the cycle matrix of the associated fundamental cycle basis by Γ_l. Each column in Γ_l is an incidence vector associated with a fundamental cycle induced by an arc not in T_l. For each arc $(s,t) \notin T_l$, the associated fundamental cycle is denoted by $C^l_{st} = T_l \cup \{(s,t)\}$ and the column by $\gamma_{l,st}$. The entry in $\gamma_{l,st}$ associated with arc (i,j) is $\gamma^{ij}_{l,st}$.

Circulations are modelled by introducing flow variables for each non-tree arc and commodity. For $(s,t) \notin T_l$, let x^l_{st} be the amount of flow sent in the fundamental cycle C^l_{st}. This yields, cf. (7.5), that the flow of commodity l on arc (i,j) is

$$\theta^l_{ij} = \sum_{(s,t) \in \tilde{A} \setminus T_l} \gamma^{ij}_{l,st} x^l_{st}, \tag{7.14}$$

for all $(i,j) \in A$ and all $l \in L$. Recall that this relation is significantly simplified for arcs not in the tree. The partitioning in (7.2) yields

$$\theta^l_{ij} = x^l_{ij}, \quad (i,j) \notin T_l,\ l \in L. \tag{7.15}$$

By construction, all θ^l defined by (7.14) satisfy the node balance constraints. To fulfill the capacity constraints, consider the total amount of flow sent along an arc $(i,j) \in A$,

$$\sum_{l \in L} \theta^l_{ij} = \sum_{l \in L} \sum_{(s,t) \in \tilde{A} \setminus T_l} \gamma^{ij}_{l,st} x^l_{st}. \tag{7.16}$$

This amount must be nonpositive for all arcs. That is,

$$\sum_{l \in L} \sum_{(s,t) \in \tilde{A} \setminus T_l} \gamma^{ij}_{l,st} x^l_{st} \leq 0, \quad (i,j) \in A. \tag{7.17}$$

To guarantee that the commodity specific flow bounds are satisfied for an $l \in L$ the arcs are partitioned into three disjoint arc sets. 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. That is,

$$P_l = \tilde{A} \setminus \left(A_l \cup \tilde{D}_l\right) \quad \text{and} \quad N_l = \left(A_l \cup \tilde{D}_l\right) \setminus T_l. \tag{7.18}$$

This yields 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 $P_l \cup N_l$ is a partitioning of the arcs outside the tree T_l and recall that these are the arcs that yield fundamental cycles.

Clearly the nonnegativity constraints for arcs that are not SP-arcs or destination arcs require

$$x_{st}^l \geq 0, \quad (s,t) \in P_l,\ l \in L. \tag{7.19}$$

There is however no such constraint for an SP-arc. Hence, there is no sign restriction on x_{st}^l when $(s,t) \in N_l$ since all arcs in the associated fundamental cycle is an SP-arc or a destination arc.

Finally, denote the increment of the objective from sending one unit of flow along the cycle $C_{st}^l = F \cup B$ by c_{st}^l. From (PR-Farkas) it is clear that all arcs but the non-SP-arcs affect the objective. Hence the objective coefficient of the variable associated with the arc (s,t) and destination l is

$$c_{st}^l = \sum_{(i,j) \in \tilde{A}_l \setminus \bar{A}_l} \gamma_{l,st}^{ij}. \tag{7.20}$$

Arcs not in $\bar{A}_l$ affect c_{st}^l; forward arcs increase c_{st}^l by 1 and backward arcs decrease it by 1. Therefore, if the entries in the fundamental cycle vector are summed up, the non-SP-arcs are counted even though they are not supposed to. Hence,

$$c_{st}^l = 1'\gamma_{l,st} - |C_{st}^l \cap \bar{A}_l| = |F| - |B| - |C_{st}^l \cap \bar{A}_l|. \tag{7.21}$$

Since $A_l \cap \bar{A}_l = \emptyset$, $T_l \subseteq A_l$ and $C_{st}^l = \{(s,t)\} \cup T_l$ it is clear that $C_{st}^l \cap \bar{A}_l \subseteq \{(s,t)\}$, so $|C_{st}^l \cap \bar{A}_l| = 1$ if $(s,t) \in \bar{A}_l$ and 0 otherwise. This yields

$$c_{st}^l = \begin{cases} |F| - |B| & \text{if } (s,t) \notin \bar{A}_l \\ |F| - |B| - 1 & \text{if } (s,t) \in \bar{A}_l. \end{cases} \tag{7.22}$$

Summarizing, the derivation above yields that (PR-Farkas) is equivalent to the following model when $\tilde{D}_l = \{(i,l) : i \neq l,\ \delta^+(i) = \emptyset\}$ for all $l \in L$.

$$\begin{array}{rcll} \sum_{l \in L} \sum_{(s,t) \in \tilde{A} \setminus T_l} c_{st}^l x_{st}^l & < & 0 & \\ \sum_{l \in L} \sum_{(s,t) \in \tilde{A} \setminus T_l} \gamma_{l,st}^{ij} x_{st}^l & \leq & 0 & (i,j) \in \tilde{A} \\ x_{st}^l & \geq & 0 & (s,t) \in P_l,\ l \in L. \end{array} \tag{PR-Farkas-CB}$$

Note that x^l_{ij} is unrestricted for all $(i,j) \in N_l$ and $l \in L$.

Since (PR-Farkas-CB) is feasible if and only if (PR-Farkas) is feasible there are no administrative weights that yield the SP-graphs A_l for all $l \in L$ unless (PR-Farkas-CB) is infeasible.

We would like to strongly emphazise that *the specific choice of trees implies that the commodity specific flow bounds are handled implicitly* by the nonnegativity constraints.

Another choice, where $T_l \not\subseteq A_l$, would require additional nontrivial constraints to satisfy the commodity specific flow bounds, potentially one for each arc and commodity. In the partial compatibility case there may be a set of SP-arcs that does not contain a spanning arborescence, this implies that there is no tree such that all commodity specific flow bounds vanish so the additional constraints have to be included to enforce the commodity specific flow bounds. Before this case is considered, some straightforward observations about model (PR-Farkas-CB) are made.

It is easy to verify that the original partial realizability model, (IPSPR-PR) in Chapter 4 and (PR) in Chapter 5, has approximately $m|L|$ constraints (not counting variable bounds), $m + n|L|$ variables and $3m|L|$ nonzero entries in the constraint matrix. Model (PR-Farkas-CB) has a capacity constraint for each arc in $\tilde{G}$ and a flow variable for each fundamental cycle and commodity. The number of fundamental cycles for destination l is $m + |\tilde{D}_l| - n + 1$ from (7.1) and each one yields a single variable. Out of these, $|A_l \cup D_l| - n + 1$ are unrestricted for destination l. Further, no cycle has more than n arcs (on average fewer in practice).

Let us also comment on the additional structure of the fundamental cycles induced by considering spanning intrees instead of general spanning trees.

A column in (PR-Farkas-CB) corresponds to a fundamental cycle, C^l_{st}, determined by the intree T_l and the arc $(s,t) \notin T_l$. In T_l there is a unique path, P^l_s, from s to l and a unique path, P^l_t, from t to l. Let node q be the apex (that is, the first common node along these paths) and denote the paths to q by $P_{sq} = P^l_s \setminus P^l_t$ and $P_{tq} = P^l_t \setminus P^l_s$. This description implies that each cycle consist of two node-disjoint paths from s to q.

One of the paths, $P_- = P_{sq}$, consists only of arcs in T_l, all these arcs yield a -1 entry in the associated column $\gamma_{l,st}$. The other path, $P_+ = (s,t) \cup P_{tq}$, begins with arc (s,t) and continues along P_{tq}, these arcs yield a 1 entry in $\gamma_{l,st}$. This yields the path-based description of the fundamental cycle C^l_{st},

$$C^l_{st} = P_+ \cup P_- = P_{sq} \cup P_{tq} \cup (s,t) = (s,t) \cup \left(P^l_s \triangle P^l_t\right). \tag{7.23}$$

Recall that the orientation of C^l_{st} is determined by the single arc not in the tree, (s,t). If the variable, x^l_{st}, associated with the cycle is positive, flow is sent forwards along arcs in P_+ and backwards along arcs in P_- . If the variable is negative, which may only occur when $(i,j) \in N_l$, it is the other way around.

This structural description of the columns as two disjoint paths will be useful when the Farkas system of (PR-Farkas-CB) is derived in Section 7.4 and when cycle basis computations are considered in Section 7.6.

7.2.2 Modelling Partial Compatibility with Cycle Bases

Now consider the partial compatibility case. In this setting, $\tilde{D}_l = \emptyset$ for all $l \in L$. This implies that it is not guaranteed that there is a reversely spanning l-arborescence among the SP-arcs. Suppose that the same modelling approach with fundamental cycle bases is used despite this fact. This will unfortunatly not yield a model where all commodity specific flow bounds are handled implicitly. Simply because there is no tree where the flow of commodity l is unrestricted if the set of SP-arcs does not contain a reversely spanning l-arborescence.

The same notation as above may be used (even though it is somewhat superfluous). The Farkas system of the partial compatibility model is obtained from (PR-Farkas) by setting $\tilde{D}_l = \emptyset$ for all $l \in L$. This yields $G = \tilde{G} = (N, \tilde{A}) = (N, A)$. Hence, the family of generalized SP-graphs $\widetilde{\mathcal{A}}_L$ is not partially compatible if and only if the following model is feasible.

$$\begin{array}{rcll}
\sum_{l \in L} \sum_{(i,j) \in A_l} \theta_{ij}^l & < & 0 & \\
\sum_{j:(i,j) \in A} \theta_{ij}^l - \sum_{j:(j,i) \in A} \theta_{ji}^l & = & 0 & i \in N,\ l \in L \\
\sum_{l \in L} \theta_{ij}^l & \leq & 0 & (i,j) \in A \\
\theta_{ij}^l & \geq & 0 & (i,j) \in A \setminus A_l.
\end{array} \tag{PC-Farkas}$$

For simplicity, it is assumed that G is biconnected so that A contains a spanning intree to each destination, $l \in L$. Let $T_l \subseteq A$ be a spanning intree to l such that $T_l \cap \bar{A}_l = \emptyset$ and the number of arcs from A_l is maximized. If no such intree exists, A_l, can not be realized, and hence, neither can $\widetilde{\mathcal{A}}_L$. As above, the cycle matrix of the associated fundamental cycle basis is denoted by Γ_l for all $l \in L$ and the incidence vector induced by an arc not in T_l by $\gamma_{l,st}$.

Circulations are modelled with the flow variables, x_{st}^l, one for each non-tree arc and commodity. Recall from (7.14) that the flow of commodity l on arc (i,j) is

$$\theta_{ij}^l = \sum_{(s,t) \in A \setminus T_l} \gamma_{l,st}^{ij} x_{st}^l, \tag{7.24}$$

for all $(i,j) \in A$ and all $l \in L$. In particular,

$$\theta_{ij}^l = x_{ij}^l, \quad (i,j) \notin T_l,\ l \in L. \tag{7.25}$$

The total amount of flow sent along an arc, $(i,j) \in A$, must be nonpositive for all arcs, which is accomplished only if

$$\sum_{l \in L} \sum_{(s,t) \in A \setminus T_l} \gamma_{l,st}^{ij} x_{st}^l \leq 0, \quad (i,j) \in A. \tag{7.26}$$

The set of arcs that are not SP-arcs is $P_l = A \setminus A_l$ and the set of SP-arcs not in the tree is $N_l = A_l \setminus T_l$. Let $\bar{P} = P_l \cap T_l$ be the subset of the arcs that are not SP-arcs but in the tree. Thus, the arcs are partitioned as $A = \left(P_l \setminus \bar{P}_l\right) \cup N_l \cup T_l$ for all $l \in L$.

The commodity specific flow bounds must be handled differently for arcs in P_l and $\bar{P}_l$. For arcs that are not SP-arcs and not in the tree, a nonnegativity constraint suffice

$$x_{st}^l \geq 0, \quad (s,t) \in P_l \setminus \bar{P}_l, \ l \in L. \tag{7.27}$$

It is more complicated to guarantee that the commodity specific flow bound associated with an arc that is not an SP-arc but in the tree is satisfied. The relation in (7.24) yields that it is required that

$$\sum_{(s,t) \in A \setminus T_l} \gamma_{l,st}^{ij} x_{st}^l \geq 0, \quad (s,t) \in \bar{P}_l, \ l \in L. \tag{7.28}$$

There is no sign restriction for an SP-arc so the variables in N_l are still unrestricted.

It remains to verify that the increment of the objective, c_{st}^l, is unaltered. This is clear since setting $\tilde{D}_l = \emptyset$ does not affect the calculations that lead to (7.22). The objective coefficients are still given by

$$c_{st}^l = \begin{cases} |F| - |B| & \text{if } (s,t) \notin \bar{A}_l \\ |F| - |B| - 1 & \text{if } (s,t) \in \bar{A}_l. \end{cases} \tag{7.29}$$

The above derivation implies that the following model is equivalent to (PC-Farkas).

$$\begin{array}{rcll} \sum_{l \in L} \sum_{(s,t) \in A \setminus T_l} c_{st}^l x_{st}^l & < & 0 & \\ \sum_{l \in L} \sum_{(s,t) \in A \setminus T_l} \gamma_{l,st}^{ij} x_{st}^l & \leq & 0 & (i,j) \in A \\ x_{st}^l & \geq & 0 & (s,t) \in P_l \setminus \bar{P}_l, \ l \in L. \\ \sum_{(s,t) \in A \setminus T_l} \gamma_{l,st}^{ij} x_{st}^l & \geq & 0 & (s,t) \in \bar{P}_l, \ l \in L, \end{array} \tag{PC-Farkas-CB}$$

where x_{ij}^l is unrestricted for all $(i,j) \in N_l, l \in L$.

Model (PC-Farkas-CB) is in our opinion not as pure as (PR-Farkas-CB) since the commodity specific flow bounds are no longer handled implicitly. The model is given for the sake of generality and to illustrate the advantages with (PR-Farkas-CB). It also shows the importance of the reversely spanning l-arborescences which further emphazise the benefits of considering partial realizability over partial compatibility. If the partial compatibility problem is to be solved at all, we prefer to use (IPSPR-PC) or the multicommodity Farkas version, (PC-Farkas).

Since partial realizability is a stronger requirement than partial compatibility and since the cycle basis model is simplified in the partial realizability case we only consider the cycle basis models for the partial realizability case in the following.

7.2.3 The Cycle Basis Model in Matrix Notation

It is sometimes convenient to put an LP in matrix form. A drawback is that this have a tendency to hide the structure of the problem, but in our case, the structure is not really revealed in (PR-Farkas-CB) anyway. We believe that using the cycle matrices instead preserves whatever structure that can be seen. Below, (PR-Farkas-CB) is rewritten into matrix form, after introducing some necessary notation.

The partial ingraphs $(A_l \cup \tilde{D}_l, \bar{A}_l)$ and a collection of intrees $T_l \subseteq A_l \cup \tilde{D}_l$ are given for all $l \in L$. The fundamental cycle basis associated with l is denoted by Γ_l.

Let ϕ_l and ψ_l be the indicator vectors of P_l and $\bar{A}_l$, respectively. Further, let Φ_l and Ψ_l be the diagonal matrices with ϕ_l and ψ_l on the diagonal, respectively. That is,

$$\phi^l_{st} = \begin{cases} 1 & \text{if } (s,t) \in P_l \\ 0 & \text{if } (s,t) \notin P_l \end{cases} \quad \text{and} \quad \psi^l_{st} = \begin{cases} 1 & \text{if } (s,t) \in \bar{A}_l \\ 0 & \text{if } (s,t) \notin \bar{A}_l. \end{cases} \tag{7.30}$$

This yields that the objective increment, c^l_{st}, of the fundamental cycle $C^l_{st} = F \cup B$ defined in (7.22) can be written as $c^l_{st} = |F| - |B| - \psi^l_{st} = 1^T \gamma_{l,st} - \psi^l_{st}$. Hence, a mixed form of model (PR-Farkas-CB) becomes

$$\begin{array}{rcll} \sum_{l \in L} (1'\Gamma_l - 1'\Psi_l)\, x^l & < & 0 & \\ \sum_{l \in L} \Gamma_l x^l & \leq & 0 & \\ \Phi^l x_l & \geq & 0 & l \in L. \end{array} \tag{PR-Farkas-CB-Matrix}$$

It is obvious how to put model (PR-Farkas-CB-Matrix) in pure matrix form, but that basically serves no purpose at all.

In our opinion, the structure is not really hidden in (PR-Farkas-CB-Matrix). One may even advocate that the structure of the capacity constraints is actually better revealed in (PR-Farkas-CB-Matrix) than in (PR-Farkas-CB). This is desirable, since that is basically where the structure is.

In matrix notation, the relation between θ and x defined in (7.14) may be written as

$$\theta^l = \Gamma^l x, \quad l \in L. \tag{7.31}$$

From this it is clear that there is a one-to-one correspondance between the set of solutions to (PR-Farkas) and the set of solutions to (PR-Farkas-CB) since Γ_l has full column rank for all $l \in L$. This is states as a theorem.

Theorem 7.1
There is a one-to-one correspondance between solutions to (PR-Farkas) *and solutions to* (PR-Farkas-CB).

This fact implies that what holds for (PR-Farkas-CB) also holds for (PR-Farkas) and vice versa and it is not necessary to explicitly state that properties presented below holds for both models.

7.3 Properties of the Cycle Basis Formulation

In this section some properties of the various models presented in the previous sections are discussed. These properties could have been derived without explicitly using the cycle basis model, due to Theorem 7.1, but it is likely that it would have been much harder to discover some of them by only considering model (PR-Farkas).

7.3.1 Constraint Redundancy

An interesting and important property of model (PR-Farkas-CB) is that the capacity constraint is in a sense binding. Therefore, it may under rather general conditions be replaced with an equality constraint according to the following theorems. This yields important practical and theoretical consequences as we will show below.

The first theorem was already mentioned (without proof) in Chapter 5. It states that a a non-saturating solution induce (several) saturating solutions.

Theorem 7.2
If $\widetilde{\mathcal{A}}_L$ is a family with at least two generalized SP-graphs and (PR-Farkas-CB) *has a nonzero feasible solution, then* (PR-Farkas-CB) *has a, not necessarily improving, feasible solution $\bar{x} \neq 0$ such that*

$$\sum_{l \in L} \Gamma_l \bar{x}^l = 0, \tag{7.32}$$

i.e. $\Theta \setminus \Theta^= \neq \emptyset \Rightarrow \Theta^= \setminus \{0\} \neq \emptyset$.

Proof: Given a nonzero solution, $\hat{x}$, to (PR-Farkas-CB) such that (7.32) does not hold for $\bar{x} = \hat{x}$, let $\gamma = \sum_{l \in L} \Gamma_l \hat{x}^l$. Using the standard form of (PR-Farkas-CB-Matrix) it is clear that $\gamma + s = 0$, where $s \geq 0$ is the contribution from the slack variables of the capacity constraints. Since γ corresponds to a circulation, s must correspond to the reverse circulation. Further, $s \geq 0$ implies that the associated circulation consists only of forward arcs, that is, the slack circulation, s, corresponds to a directed cycle.

A circulation can be expressed as a sum of fundamental cycles for some commodity, say l'. This yields $s = \Gamma_l x'$, for some x' and it follows from (7.25) that $x'_{ij} = s_{ij}$ for all $(i, j) \in \tilde{A} \setminus T_{l'}$. Therefore, $x' \geq 0$ and (with some abuse of notation) $\bar{x} = \hat{x} + x' \geq 0$ satisfies (7.32). Finally, l' can be choosen such that $\bar{x} \neq 0$ since there are at least two generalized SP-graphs. □

Remark 7.1. It has to be assumed that there are at least two generalized SP-graphs to avoid the absurd case where a directed cycle in a single generalized SP-graph yields a non-improving solution. In this case, the "fill-in" commodity is the original commodity itself, so the flow is cancelled, and $\bar{x}$ becomes the zero solution.

This theorem suggests that it may be possible to replace the capacity inequality constraint by an equality constraint which would yield the equality formulation

$$\begin{array}{rcll} \sum_{l\in L} 1'\Psi_l x^l & > & 0 & \\ \sum_{l\in L} \Gamma_l x^l & = & 0 & \\ \Phi^l x_l & \geq & 0 & l \in L, \end{array} \qquad \text{(PR-Farkas-CB-Eq)}$$

where the objective has been simplified by the capacity constraint

$$\sum_{l\in L} \Gamma_l x^l = 0. \qquad (7.33)$$

It is in general not guaranteed that a saturating solution obtained by the proof technique above is improving, therefore model (PR-Farkas-CB-Eq) can not always be used. However, it is actually under rather general conditions possible to choose the fill-in commodity so that the saturating solution obtained from a non-saturating solution becomes improving. It is sufficient and necessary that the directed slack-cycle is filled-in by a commodity such that some arc in the cycle is a non-SP-arc. This yields the following theorem.

Theorem 7.3
If $A_l \cap \bar{A}_{l'} = \emptyset$ for all $l \in L$ and there for every directed cycle C in G is a commodity l' such that $C \cap \bar{A}_{l'} \neq \emptyset$, then (PR-Farkas-CB) *has a feasible and improving solution if and only if* (PR-Farkas-CB-Eq) *has a feasible and improving solution, i.e.*

$$\Theta \setminus \Theta^{=} \neq \emptyset \Rightarrow \Theta^{=} \neq \emptyset. \qquad (7.34)$$

Proof: Trivially, if (PR-Farkas-CB) does not have a feasible and improving solution then (PR-Farkas-CB-Eq) has no feasible and improving solution. Now, let $\hat{x}$ be a nonzero solution to (PR-Farkas-CB) where the capacity constraint is satisfied with strict inequality and $s \geq 0$ are the slack variables. The directed cycle induced by the positive slack variables is denoted by C. By assumption, there is a commodity l' such that $C \cap \bar{A}_{l'} \neq \emptyset$. This yields $s = \Gamma_{l'} x'$, for some $x' \geq 0$ and $\bar{x} = \hat{x} + x' \geq 0$ is feasible w.r.t. the non-objective constraints in (PR-Farkas-CB-Eq). It remains to show that $\bar{x}$ is improving.

Assume not, then

$$\sum_{l\in L} (1'\Gamma_l - 1'\Psi_l)\, \bar{x}^{l'} = 0, \qquad (7.35)$$

since the objective is nonpositive, cf. the proof of Proposition 5.3. This implies that

$$\sum_{l\in L} 1'\Psi_l \bar{x}^{l'} = 0, \qquad (7.36)$$

and in particular that

$$\psi_{st}^{l'}\bar{x}_{st}^{l'} = 0 \quad \text{for all } (s,t) \in C. \tag{7.37}$$

By construction, $\bar{x}_{st}^{l'} > 0$ for all arcs in C. Hence, $\psi_{st}^{l'} = 0$, for all $(s,t) \in C$, which means that $(s,t) \notin \bar{A}_l$ from (7.30). Therefore, $C \cap \bar{A}_{l'} = \emptyset$, a contradiction. □

An interpretation of this theorem is that there is an improving solution to (PR-Farkas-CB) if only if at least one fundamental cycle associated with a non-SP-arc carries positive flow. This yields Theorem 2 in [24] as a corollary which states that all constraints must not be satisfied with equality. Also note that our result is more general since it is not restricted to the spanning case.

Since it could be cumbersome to verify the condition in the previous theorem a few more conditions are considered. The following three stronger conditions are worth mentioning. They are all easier to verify but yield somewhat weaker corollaries.

1. For every arc $(i,j) \in A$ there is a commodity l such that $(i,j) \in \bar{A}_l$.
2. All nodes in G is a destination, that is $L = N$.
3. Some SP-graph has been completed, that is $A = A_l \cup \bar{A}_l$, and does not contain a directed cycle.

The most important special case in practice may be when $L = N$. This implies that we may without loss of generality assume that $\delta_l^+(l) \subseteq \bar{A}_l$ for all l, cf. Lemma 4.1 in Chapter 4. Also note that case 3 yields the spanning case where all SP-graphs have been completed as a special case.

The above observations yield the following corollaries.

Corollary 7.1
If there for any arc $(i,j) \in A$ is a commodity l such that $(i,j) \in \bar{A}_l$, then (PR-Farkas-CB) *has a feasible and improving solution if and only if* (PR-Farkas-CB-Eq) *has a feasible and improving solution.*

Proof: Consider an arbitrary directed cycle C and an arc $(i,j) \in C$. Since there is a commodity l such that $(i,j) \in \bar{A}_l$, we get $C \cap \bar{A}_{l'} \neq \emptyset$. □

Corollary 7.2
If each node in G is a destination, i.e. $L = N$, then (PR-Farkas-CB) *has a feasible and improving solution if and only if* (PR-Farkas-CB-Eq) *has a feasible and improving solution.*

Proof: When $L = N$ we may without loss of generality assume that $\delta_l^+(l) \subseteq \bar{A}_l$ for all $l \in L$. This yields that the condition in Corollary 7.1 is satisfied for an arc (l,j) by commodity l. □

Corollary 7.3
If there is an SP-graph that is completed, i.e. $A = A_l \cup \bar{A}_l$ *and* A_l *does induce a directed cycle, then* (PR-Farkas-CB) *has a feasible and improving solution if and only if* (PR-Farkas-CB-Eq) *has a feasible and improving solution.*

Proof: The commodity associated with the completed SP-graph may be choosen in Theorem 7.3. □

In the reminder of this section it will be assumed that the prerequisites in Theorem 7.3 (or some of its corollaries) are satisfied so that the model with equalities may be used, that is model (PR-Farkas-CB-Eq). This implies that some constraints are redundant.

Corollary 7.4
If there for every directed cycle C *in* G *is a commodity* l' *such that* $C \cap \bar{A}_{l'} \neq \emptyset$*, then* $n-1$ *constraints are redundant in* (PR-Farkas-CB).

Proof: The prerequisites of Theorem 7.3 are satisfied. Hence, equalities may be used instead of inequalities in (PR-Farkas-CB) which yields (PR-Farkas-CB-Eq). Since the dimension of the cycle space is $\tilde{m} - n + 1$, cf.(7.1), the rank of the constraint matrix is $\tilde{m} - n + 1$. Hence, $n - 1$ out of the $\tilde{m}$ constraints are redundant. □

An alternative proof of the above corollary may yield a better understanding of the result and also shows how it may be applied in practice to strengthen the result whenever some ingraph is not a tree.

Let $l' \in L$ be a destination and consider the corresponding ingraph. If the constraint matrix, Γ say, is partitioned such that the topmost rows correspond to the arcs not in $T_{l'}$, then there is a permutation of the columns such that

$$\Gamma = \begin{pmatrix} I & \overline{\Gamma} \\ \overline{T}_{l'} & \overline{T} \end{pmatrix}. \tag{7.38}$$

Here $\overline{T}_{l'}$ is the tree-arc part of $\Gamma_{l'}$ and $\overline{\Gamma}$ and $\overline{T}$ are submatrices of Γ induced by $\overline{T}_{l'}$.

Since rank $\Gamma = \tilde{m} - n + 1$, and $\Gamma x = 0$, the last $n - 1$ constraints that corresponds to the tree arcs can be removed. This indeed yields the reduced row echelon form of Γ when it is partitioned as in (7.38),

$$\begin{pmatrix} I & \overline{\Gamma} \end{pmatrix}. \tag{7.39}$$

Consider an arc $(i, j) \in A_{l'} \setminus T_{l'}$. The corresponding variable, $x_{ij}^{l'}$, is unrestricted and its objective coefficient is 0 since $(i, j) \in A_{l'}$ and $A_{l'} \cap \bar{A}_{l'} = \emptyset$, cf. (7.30). Therefore, the variable and constraint may be eliminated. Extending this reasoning yields that all constraints associated with the non-tree arcs of the ingraph with destination l' can be eliminated in (PR-Farkas-CB-Eq). Corollary 7.4 holds for any SP-graph, hence the strongest result is obtained by selecting the destination with the most arcs in the ingraph.

Corollary 7.5
If there for every directed cycle C in G is a commodity l' such that $C \cap \bar{A}_{l'} \neq \emptyset$, then

$$n - 1 - \max_{l} |N_l| \tag{7.40}$$

constraints are redundant in (PR-Farkas-CB).

Using equalities and redundancy of constraints in (PR-Farkas-CB) have a natural interpretation in (PR). The consequence of using equalities is that the link weight variables in (PR) become unrestricted.

Corollary 7.6
If there for every directed cycle C in G is a commodity l' such that $C \cap \bar{A}_{l'} \neq \emptyset$, then model (PR) *has a feasible solution if and only if it has a feasible solution when the lower bound constraints are removed and the constraints*

$$w_{ij} = 1, \quad (i,j) \in T_{l'}, \tag{7.41}$$

are added for a single destination $l' \in L$.

Proof: This follows from LP-duality, call (PR-Farkas-CB) the primal and (PR) the dual. All redundant tree arc equality constraints are actually linear combinations of the non-tree arc equality constraints. Therefore, we know that there is a dual optimal solution where all dual variables associated with the tree arcs are 0. Because of the variable substitution, this corresponds to $w_{ij} = 1$ for $(i,j) \in T_{l'}$. □

Remark 7.2. The above corollary can be used to fix the values of all variables in a single ingraph. However, not all of them will be 1. The non-tree arc values are determined by the node potentials induced by the tree arcs. Note the connection to Corollary 7.5 and that the ingraph should be choosen maximal.

Let us finally elaborate on these last results about fixing link weight variables in (PR). Even though the inequalities in the capacity constraints can not in general be replaced by equalities, it is actually always feasible to fix some link weight variables in (PR). The reasoning is as follows.

Let θ be a saturating solution obtained by the technique in the proof of Theorem 7.2 that is not improving. Now consider what happens with θ when the generalized SP-graphs are extended (possibly completed). Obviously, the feasibility is not affected, but the objective value is and the argument from Theorem 7.3 will eventually apply which will make θ improving. Put differently; there is no way to complete the generalized SP-graphs to spanning SP-graphs such that θ is non-improving. This yields the following.

Proposition 7.1
If $\widetilde{\mathcal{A}}_L$ is a family of generalized SP-graphs that are realizable. Then, for any $l \in L$ and any reversely spanning arborescence, B_l, contained $A_l \cup \tilde{D}_l$ there exist a set of (possibly negative) administrative weights, $\hat{w}$, with the following properties.

- *$\hat{w}$ is fully compatible with $\widetilde{\mathcal{A}}_L$, and*

- $\hat{w}_{ij} = 1$ *for all* B_l.

In the same manner as in Remark 7.2 all link weights of arcs in $A_l \setminus B_l$ may also be fixed.

Note however that the fixation of link weights can in general only be used to determine feasibility, that is partial realizability. The link weights may be more or less useless if they are negative.

Clearly, the constraint redundancy may be very important, theoretically and practically. It remains to investigate how using unrestricted link weights and fixing link weights affect the performance. Also, the fact that the capacity constraints are binding should be taken into account when a solution method is developed, or an LP-solver choosen.

Remark 7.3. We are currently investigating the performance of different solution methods for IPSPR in an ongoing project. Our preliminary results shows that it is most efficient to either solve model (PR-Farkas-CB) with an LP-solver or to use a variable transformation that makes model (PR-Farkas) an ordinary multicommodity problem and then solve it with a specialized multicommodity solver.

7.4 The Farkas System of the Cycle Basis Model

The description of fundamental cycles as arc-disjoint paths at the end of Section 7.2.1 yields an interpretation of the Farkas system of (PR-Farkas-CB). A fundamental cycle, C^l_{st}, is uniquely determined by the intree T_l and the arc (s,t) not in the tree. It is formed by the two paths, P_+ and P_-, that starts at s and ends at the apex, that is

$$P_+ = (s,t) \cup P^l_t \setminus P^l_s \quad \text{and} \quad P_- = P^l_s \setminus P^l_t, \tag{7.42}$$

where P^l_s and P^l_t are the unique paths in T_l to l from s and t, respectively. Since a column, $\gamma_{l,st}$, in (PR-Farkas-CB) corresponds to a fundamental cycle the associated row is described by the paths P_+ and P_-. It is clear that the row must have a -1 entry for each arc in P_- and a $+1$ entry for each arc in P_+, cf. Equations (7.42) and (7.23) and the end of Section 7.2.1. Finally, the right hand side becomes $-c^l_{st}$. Recall that

$$P_l = \tilde{A} \setminus \left(A_l \cup \tilde{D}_l\right) \quad \text{and} \quad N_l = \left(A_l \cup \tilde{D}_l\right) \setminus T_l, \tag{7.43}$$

this yields the following Farkas system of (PR-Farkas-CB).

$$\begin{array}{rcll}
w_{st} + \sum\limits_{(i,j)\in P^l_t \setminus P^l_s} w_{ij} - \sum\limits_{(i,j)\in P^l_s \setminus P^l_t} w_{ij} & \geq & -c^l_{st} & (s,t) \in P_l,\ l \in L \\
w_{st} + \sum\limits_{(i,j)\in P^l_t \setminus P^l_s} w_{ij} - \sum\limits_{(i,j)\in P^l_s \setminus P^l_t} w_{ij} & = & -c^l_{st} & (s,t) \in N_l,\ l \in L \\
w_{ij} & \geq & 0 & (i,j) \in \tilde{A}.
\end{array}$$
(PR-Farkas-CB-Farkas)

Using (7.42), the right hand side can be rewritten as

$$-c_{st}^l = \begin{cases} -(|F|-|B|) = |P_-|-|P_+|-1, & \text{if } (s,t) \notin \bar{A}_l, \\ -(|F|-|B|-1) = |P_-|-|P_+|, & \text{if } (s,t) \in \bar{A}_l. \end{cases} \tag{7.44}$$

Now, a change of variables makes (PR-Farkas-CB-Farkas) very similar to (PR); substituting $\bar{w} = w + 1$, using (7.44), simplifying and renaming $\bar{w}$ back to w yields,

$$\begin{array}{rcll} w_{st} + \sum\limits_{(i,j)\in P_t^l \setminus P_s^l} w_{ij} - \sum\limits_{(i,j)\in P_s^l \setminus P_t^l} w_{ij} & \geq & 1 & (s,t) \in \bar{A}_l,\ l \in L \\ w_{st} + \sum\limits_{(i,j)\in P_t^l \setminus P_s^l} w_{ij} - \sum\limits_{(i,j)\in P_s^l \setminus P_t^l} w_{ij} & \geq & 0 & (s,t) \in U_l,\ l \in L \\ w_{st} + \sum\limits_{(i,j)\in P_t^l \setminus P_s^l} w_{ij} - \sum\limits_{(i,j)\in P_s^l \setminus P_t^l} w_{ij} & = & 0 & (s,t) \in N_l,\ l \in L \\ w_{ij} & \geq & 1 & (i,j) \in A. \end{array} \tag{PR-Path}$$

Models (PR) and (PR-Farkas-CB-Farkas) or (PR-Path) are clearly equivalent from linear programming theory since the duals of equivalent linear programs are equivalent (alternatively, Farkas lemma can be used twice). A constructive proof of this fact will however be given instead to illuminate the relation more exactly, cf. Theorem 7.4, Theorem 7.5 and Figure 7.3 below.

Theorem 7.4
Model (PR) *has a feasible solution if and only if* (PR-Path) *has a feasible solution.*

Proof: Let $(\bar{w}, \bar{\pi})$ be a feasible solution to (PR) and define $w = \bar{w} - 1 \geq 0$. The constraints in (PR) associated with tree arcs yields

$$-\bar{\pi}_t^l = \bar{\pi}_l^l + \sum_{(i,j)\in P_t^l} \bar{w}_{ij} \tag{7.45}$$

for all $t \in N$. Consider the left hand side of the constraint in (PR-Path) associated with destination $l \in L$ and arc $(s,t) \in \tilde{A} \setminus T_l$ minus $|P_s^l| - |P_t^l|$,

$$\begin{aligned} & w_{st} + \sum_{(i,j)\in P_t^l \setminus P_s^l} w_{ij} - \sum_{(i,j)\in P_s^l \setminus P_t^l} w_{ij} + |P_t^l| - |P_s^l| = \\ & = w_{st} + \sum_{(i,j)\in P_t^l} (w_{ij}+1) - \sum_{(i,j)\in P_s^l} (w_{ij}+1) = \\ & = \bar{w}_{st} - 1 + \sum_{(i,j)\in P_t^l} \bar{w}_{ij} - \sum_{(i,j)\in P_s^l} \bar{w}_{ij} = \\ & = \bar{w}_{st} - 1 + \left(\bar{\pi}_l^l - \bar{\pi}_t^l\right) - \left(\bar{\pi}_l^l - \bar{\pi}_s^l\right) = \bar{w}_{st} - 1 - \bar{\pi}_t^l + \bar{\pi}_s^l. \end{aligned} \tag{7.46}$$

Feasibility is now easily verified by comparing this left hand side with the corresponding right hand side for the three possible cases: $(s,t) \in \bar{A}_l, (s,t) \in U_l$ and $(s,t) \in \left(A_l \cup \tilde{D}_l\right) \setminus T_l$. Using the feasibility of $(\bar{w}, \bar{\pi})$ in model (PR) trivially yields compatible comparison operators and right hand sides with (7.46), which means that the constraints are feasible in (PR-Path). Hence w solves (PR-Path).

Now, let $\bar{w}$ be a feasible solution of (PR-Path) and define

$$w = \bar{w} + 1 \quad \text{and} \quad \pi_s^l = - \sum_{(i,j) \in P_s^l} w_{ij}. \tag{7.47}$$

An argument almost identical to the derivation above yields that w solves (PR).

□

From the above proof it is seen that (PR-Path) is the model obtained when the first set of constraints in (PR) is used to eliminate the unrestricted variables, that is, the node potentials.

The relations between the partial realizability models are summarized in Theorem 7.5 and Figure 7.3 below.

Theorem 7.5
Let w be a weight vector. Then, the following statements are equivalent.

1. *w is part of a feasible solution to* (PR).
2. *w is a feasible solution to* (PR-Path).
3. (PR-Farkas) *is infeasible.*
4. (PR-Farkas-CB) *is infeasible.*

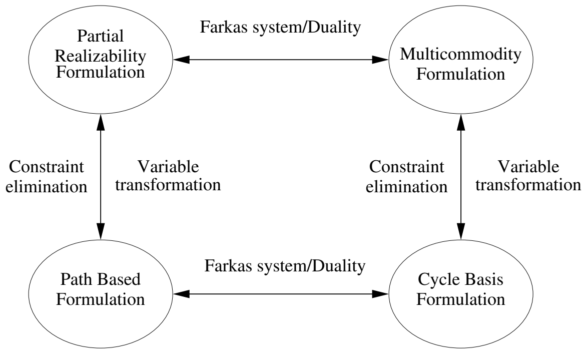


Figure 7.3: *The relation between the four partial realizability models* (PR), (PR-Farkas), (PR-Farkas-CB) *and* (PR-Path).

Remark 7.4. The equivalence between (PR) and (PR-Path) is indeed interesting. It shows that only a polynomial number of "fundamental" paths are required to appropriately model an IPSPR problem, cf. the path based models in Chapter 4 and the litterature (e.g. [9, 6]). This gives an interesting alternative separation procedure to the k-shortest path approach commonly used in constraint generation procedures. Alternatively, the fundamental subset of paths required can be determined a priori without using constraint generation.

To illustrate how the fundamental cycle bases are used and make the cycle basis formulation, (PR-Farkas-CB), more clear some examples are given in the next section.

7.5 Numerical Examples for the Cycle Basis Model

Recall that the set of feasible solutions to (PR-Farkas) was denoted by Θ in Chapter 5 and that the subset of non-improving solutions was Θ^0. The corresponding sets of solutions to (PR-Farkas-CB) will here be referred to as Λ and Λ^0, respectively.

Example 7.3

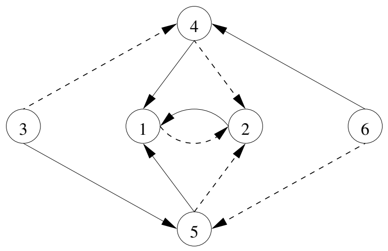


Figure 7.4: *Two SP-graphs that form a valid cycle. The SP-arc trees of the SP-graph with destination 1 and 2 are indicated by solid and dashed arcs, respectively.*

Consider the graph and SP-graphs in Figure 7.4 and assume that splitting is not allowed. The sets of SP-arcs are given by

$$A_1 = \{(2,1),\ (3,5),\ (4,1)\ (5,1),\ (6,4)\} \tag{7.48}$$

and

$$A_2 = \{(1,2),\ (3,4),\ (4,2),\ (5,2),\ (6,5)\}, \tag{7.49}$$

and the set of arcs in the graph is $A = A_1 \cup A_2$. By assumption there must be no splitting, therefore the sets of non-SP-arcs become

$$\bar{A}_1 = A \setminus A_1 = A_2 \quad \text{and} \quad \bar{A}_2 = A \setminus A_2 = A_1. \tag{7.50}$$

Since both SP-graphs are spanning intrees the outdegree is 1 for all nodes but the destination nodes in A_1 and A_2, hence no destination arcs are required. That is, $\tilde{D} = \emptyset$. Further, all arcs in the SP-graphs are in the associated tree, therefore $N_l = \emptyset$ for $l = 1, 2$. This yields the constraint matrix and cost vector given in Table 7.1. Since $N_l = \emptyset$ for $l = 1, 2$, all variables must be nonnegative in this instance.

It is easy to verify that the set of feasible solutions to (PR-Farkas-CB), denoted by Λ, becomes the ray generated by

$$x^1_{34} = x^1_{65} = x^2_{35} = x^2_{64} = \lambda,\ x^l_{ij} = 0 \text{ otherwise},\ \lambda > 0, \tag{7.51}$$

and that the set of non-improving solutions is $\Lambda_0 = \{0\}$.

The corresponing sets of solutions to (PR-Farkas), Θ and Θ^0, can be obtained from relation (7.14). This yields

dest. (l)	1	1	1	1	1	2	2	2	2	2
arc (s,t)	12	34	42	52	65	21	35	41	51	64
$(1,2)$	1					**1**		**1**	**1**	
$(2,1)$	**1**		**1**	**1**		1				
$(3,4)$		1					**-1**			
$(3,5)$		**-1**					1			
$(4,1)$		**1**	**-1**		**-1**			1		
$(4,2)$			1				**-1**	**-1**		1
$(5,1)$		**-1**		**-1**	**1**				1	
$(5,2)$				1			**1**		**-1**	**-1**
$(6,4)$					**-1**					1
$(6,5)$					1					**-1**
c^l_{st}	1	-1	0	0	-1	1	-1	0	0	-1

Table 7.1: *The constraint matrix and cost vector in Example 7.3. A bold entry in the matri represents that the arc is the associated tree.*

$$\Theta^0 = \{0\} \quad \text{and} \quad \Theta = \{\theta t \mid t > 0\}, \tag{7.52}$$

where the nonzero components of θ are

$$\begin{array}{c} \theta^1_{34} = \theta^1_{65} = \theta^2_{35} = \theta^2_{64} = 1 \\ \theta^2_{34} = \theta^2_{65} = \theta^1_{35} = \theta^1_{64} = -1. \end{array} \tag{7.53}$$

From this solution it is clear that the corresponding family of cycles is

$$\mathcal{C} = \{C^1 = F^1 \cup B^1,\ C^2 = F^2 \cup B^2\}, \tag{7.54}$$

where

$$F^1 = \{(3,4),\ (6,5)\},\ F^2 = \{(3,5),\ (6,4)\} \tag{7.55}$$

and

$$B^1 = \{(3,5),(6,4)\},\ B^2 = \{(3,4),(6,5)\}. \tag{7.56}$$

Hence the infeasible stucture is a generalized saturating valid cycle.

The family of cycles can of course also be obtained from the solution in (7.51). In general, for a solution of the form in (7.51) where $\lambda = 1$ for all nonzero components, the incidence vector of the cycle associated with a destination l is just $\Gamma^l x^l$ and from that vector the sets F and B are the elements equal to 1 and -1, respectively. Recall that Γ^l is totally unimodular.

For this instance it is clear that there are no administrative weights that yield the ingraphs in Figure 7.4 since $\Lambda \neq \emptyset$ (or, equivalently $\Theta \neq \emptyset$).

Let us adjust the previous example slightly to so that it becomes somewhat more complicated, but on the other hand illustrates some other aspects of model (PR-Farkas-CB).

Example 7.4

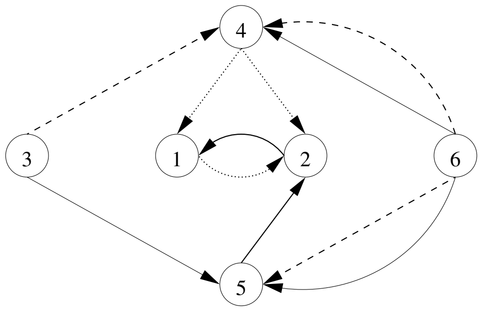


Figure 7.5: *Two SP-graphs, A_1 and A_2, that form several forcing, but no infeasible, structures. The SP-arcs of the SP-graph with destination 1 and 2 are indicated by solid and dashed arcs, respectively. Dotted arcs are in the graph as destination arcs but not in any SP-graph.*

The underlying graph in this eample is almost the same as in the previous example but the SP-graphs have been altered and are not spanning anymore. Now assume that the SP-graph with destination 1, $(A_1 \cup \tilde{D}_1, \bar{A}_1)$ is defined by

$$A_1 = \{(2,1),\ (3,5),\ (5,2),\ (6,4),\ (6,5)\} \quad \text{and} \quad \bar{A}_1 = \{(3,4)\}, \tag{7.57}$$

and that the SP-graph with destination 2, $(A_2 \cup \tilde{D}_2, \bar{A}_2)$ by

$$A_2 = \{(3,4),\ (6,4),\ (6,5)\} \quad \text{and} \quad \bar{A}_2 = \{(2,1),\ (5,2)\}. \tag{7.58}$$

The SP-arcs of these SP-graphs are given in Figure 7.5.

Since the SP-graphs no longer contain spanning intrees they have to be augmented with some appropriate destination arcs. A destination arc is added for nodes with outdegree 0, this yields

$$\tilde{D}_1 = \{(4,1)\} \quad \text{and} \quad \tilde{D}_2 = \{(1,2),\ (4,2),\ (5,2)\}. \tag{7.59}$$

Note that this implies that there are two parallell arcs from 5 to 2. There are also two choices for an intree in to both destinations, assume that arc $(6,4)$ is choosen over $(6,5)$ in both cases. This yields the trees,

$$T_1 = \underbrace{\{(4,1)\}}_{=\tilde{D}_1} \cup \underbrace{\{(2,1),\ (3,5),\ (5,2),\ (6,4)\}}_{\subseteq A_1}, \tag{7.60}$$

and

$$T_2 = \underbrace{\{(1,2),\ (4,2),\ (5,2)\}}_{=\tilde{D}_2} \cup \underbrace{\{(3,4),\ (6,4)\}}_{\subseteq A_2}. \tag{7.61}$$

From these trees, the sets of SP-arcs not in the trees become

$$N_1 = N_2 = \{(6,5)\}. \tag{7.62}$$

Since $N_l \neq \emptyset$ the variable bounds are also affected. The constraint matrix and important associated information is given in Table 7.2.

dest. (l)	1	1	1	1	1	2	2	2	2	2
arc (s,t)	$\tilde{12}$	34	$\tilde{42}$	$\tilde{52}$	65	21	35	$\tilde{41}$	52	65
$(1,2)^*$	1					**1**		**1**		
$(2,1)$	**1**	**-1**	**1**		**1**	1				
$(3,4)$		1					**-1**			
$(3,5)$		**-1**					1			
$(4,1)^*$		**1**	**-1**		**-1**			1		
$(4,2)^*$			1				**-1**	**-1**		**-1**
$(5,2)$		**-1**		**1**	**1**				1	
$(5,2)^*$				**-1**			**1**		**-1**	**1**
$(6,4)$					**-1**					**-1**
$(6,5)$					1					1
c_{st}^l	2	-2	1	0	1	1	0	1	-1	0
$\bar{A}_l/\tilde{U}_l$	$\tilde{U}_l$	$\bar{A}_l$	$\tilde{U}_l$	$\tilde{U}_l$	$\tilde{U}_l$	$\bar{A}_l$	$\tilde{U}_l$	$\tilde{U}_l$	$\bar{A}_l$	$\tilde{U}_l$
P_l/N_l	P_l	P_l	P_l	P_l	N_l	P_l	P_l	P_l	P_l	N_l

Table 7.2: *The constraint matrix and cost vector in Example 7.4. Arcs marked with an asterix are destination arcs and bold matrix entries are in the associated tree.*

For this instance, the set of feasible solutions to (PR-Farkas-CB) becomes the cone generated by the four rays

$$\begin{aligned}
\lambda^{(1)} &= \{x_{34}^1 = x_{65}^1 = x_{35}^2 = -x_{65}^2 = \lambda,\ x_{ij}^l = 0 \text{ otherwise},\ \lambda > 0\},\\
\lambda^{(2)} &= \{x_{34}^1 = x_{42}^1 = x_{52}^1 = x_{35}^2 = \lambda,\ x_{ij}^l = 0 \text{ otherwise},\ \lambda > 0\},\\
\lambda^{(3)} &= \{x_{34}^1 = x_{42}^1 = x_{35}^2 = x_{52}^2 = \lambda,\ x_{ij}^l = 0 \text{ otherwise},\ \lambda > 0\},\\
\lambda^{(4)} &= \{x_{42}^1 = x_{65}^1 = -x_{65}^2 = \lambda,\ x_{ij}^l = 0 \text{ otherwise},\ \lambda > 0\}.
\end{aligned} \tag{7.63}$$

In our last example we consider what happens when the destination arcs are ommited in the non-spanning case. That is, if model (PC-Farkas-CB) is solved instead of model (PR-Farkas-CB).

Example 7.5

Recall Example 6.1 at the beginning of Section 6.1. Consider solving this with model (PR-Farkas-CB). This yields the constraint matrix and cost vector in Table 7.3. All ingraphs are trees and there is no splitting. Thus, there are no arcs in N_l and $\bar{A}_l$ is the complement of the tree arcs for $l = 1, \ldots, 3$.

dest.	1	1	1	1	3	3	3	3	5	5	5	5
arc	13	25	32	45	25	32	34	41	21	34	41	53
$(1,3)$	1				-1	1		1	1		1	
$(2,1)$		-1	1		-1	1			1			
$(2,5)$		1			1					-1	1	1
$(3,2)$			1			1			1	-1	1	1
$(3,4)$	1	1	-1	1			1			1		
$(4,1)$	1	1	-1					1			1	
$(4,5)$				1			1	-1		1	-1	
$(5,3)$		1		1	1		1	-1				1
c_a	2	2	-1	2	-1	2	2	-1	2	-1	2	2

Table 7.3: *The constraint matrix and cost vector in Example 7.5.*

As pointed out in Example 6.1 the there is a unique ray that is feasible here. The cycle family that induce the solutions is shown in Figure 6.2 on page 107. If it is projected onto the Λ space we get

$$\Lambda = \{x \mid x^1_{32} = x^3_{25} = x^3_{41} = x^5_{34} = \lambda,\ x^l_{ij} = 0 \text{ otherwise},\ \lambda > 0\}. \tag{7.64}$$

Now consider a minor modification of this example. The graph is the same, but the SP-arcs that are not part of the conflict are removed. Put these arcs in the corresponding U_l set. The new SP-graphs are shown in Figure 7.6.

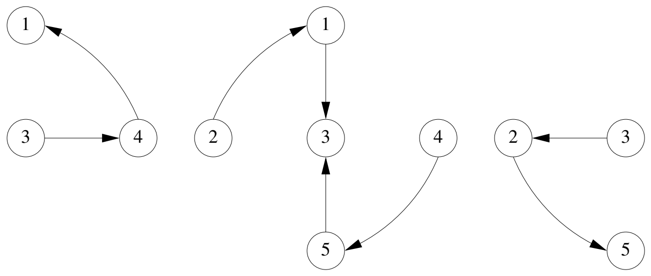


Figure 7.6: *The ingraphs graphs A_1, A_3 and A_5, respectively.*

Suppose that this instance is solved with (PC-Farkas-CB) instead of (PR-Farkas-CB). A tree has to be selected for each destination. We use the intrees from above since this yield the same aggregated capacity constraints as above. Since model (PC-Farkas-CB) is used, the constraints corresponding to arcs outside an SP-graph but in an intree must be handled. This yields four extra constraints in total. The augmented constraint matrix is given in Table 7.4

dest.	1	1	1	1	3	3	3	3	5	5	5	5
arc	13	25	32	45	25	32	34	41	21	34	41	53
$(1,3)$	1				-1	1		1	1		1	
$(2,1)$		-1	1		-1	1			1			
$(2,5)$		1			1					-1	1	1
$(3,2)$			1			1			1	-1	1	1
$(3,4)$	1	1	-1	1			1			1		
$(4,1)$	1	1	-1					1			1	
$(4,5)$				1			1	-1		1	-1	
$(5,3)$		1		1	1		1	-1				1
$(2,1)$		-1	1									
$(5,3)$		1		1								
$(1,3)$									1		1	
$(4,5)$										1	-1	
c_a	2	2	-1	2	-1	2	2	-1	2	-1	2	2

Table 7.4: *The constraint matrix and cost vector with the additional capacity constraints.*

Since the ray from above satisfies the new constraints it is a solution in the modified example as well. Clearly, there can be no other solutions since we consider a restriction of the problem. Hence,

$$\Lambda = \{x \mid x_{32}^{1} = x_{25}^{3} = x_{41}^{3} = x_{34}^{5} = \lambda,\ x_{ij}^{l} = 0 \text{ otherwise},\ \lambda > 0\}. \tag{7.65}$$

When cycle basis formulations are used to solve IPSPR problems it is clear from the last example that model (PR-Farkas-CB) is superior to (PC-Farkas-CB). It yields a smaller model and finds at least as many conflicts. In fact, model (PR-Farkas-CB) may find (potential) conflicts that (PC-Farkas-CB) can not due to the destination arcs.

7.6 Cycle Basis Computations

In this section it is shown how to obtain the cycle matrix Γ w.r.t. an intree by elementary graph operations. Efficient algorithms for multiplication of the cycle matrix with a vector are also given. Our implementations run in $\mathcal{O}(m)$ time. This is a significant improvement over the naive implementation of matrix-vector multiplication with time complexity $\mathcal{O}(m^2)$, but also over the straightforward sparse matrix-vector multiplication with a worst case time complexity $\mathcal{O}(mn)$. This could improve some solution methods where matrix-vector multiplication is a bottleneck operation, e.g. the performance of an interior point algorithm often depend on efficient solution of the projection subproblems.

7.6.1 Computing the Cycle Matrix

Let $G = (N, A)$ be a directed graph and $T \subseteq A$ a spanning intree rooted at node $r \in N$. This implies that any arc $(s, t) \in A \backslash T$ induce a unique undirected cycle $C \subseteq T \cup \{(s, t)\}$. Since T is an intree the cycle C consists of the arc (s, t) and two directed paths to r, one from s and one from t. The first common node in these two paths is referred to as the apex, cf. the description of the structure of the fundamental cycles at the end of Section 7.2.1.

For a given cycle C determined by (s, t), let P_+ be the unique path from t to the apex and P_- be the unique path from s to the apex. Using this notation we have

$$C = (s, t) \cup P_+ \cup P_-. \tag{7.66}$$

An oriented cycle is obtained from C when (s, t) and the arcs in P_+ are used as forward arcs and the arcs in P_- as backward arcs. This implies that the nonzero entries in the column of the cycle matrix Γ corresponding to the fundamental cycle induced by (s, t) are easily determined.

The efficency of an algorithm that compute the column of Γ induced by the arc (s, t) strongly depends on the data structures. Especially, which (and how) information is stored. We do not go into implementation details or data structures, but rather outline a strategy that can be used to determine the nonzero entries in the column. The most simple idea is to find the apex and then generate the two paths from s and t, respectively, to the apex. If the apex is given beforehand, the paths P_+ and P_- can just be traced via the parent (or successor) indices until the apex is reached.

If the apex is not given, which is in general likely, the algorithm must implicitly determine the apex on the fly. To achieve this we assume that an ordering, o, of the nodes in the tree is available. The order of the root is zero and $o(i) < o(j)$ if there is a path from j to i.

Our algorithm successively build up the paths P_+ and P_- until the apex is reached by augmenting the appropriate path. If the nodes i and j are on different paths, any node can be 'traced'. When they are on the same path one of them must be the apex and then the other should be traced, this is achieved if the node with the largest order is traced.

Remark 7.5. Note that any ordering with this property will do, e.g. the depth index or the topological ordering. These two ordering have the advantageous property that they may be computed in linear time.

Algorithm 7.6.1. Given $G = (N, A)$, T, $(s, t) \in A \setminus T$ and an ordering o, compute the column γ_{st} of Γ_T associated with (s, t).

1. Let $P_+ = P_- = \emptyset$ and $(i, j) = (s, t)$.

```
2. If $o(j) < o(i)$,
       let $p$ be the successor of $i$ in $T$ and add $(i,p)$ to $P_-$. Set $i = p$. Goto 4.
3. If $o(j) \not< o(i)$,
       let $p$ be the successor of $j$ in $T$ and add $(j,p)$ to $P_+$ and set $j = p$.
4. If $i \neq j$,
       goto 2.
5. Set $\gamma_{ij} = \begin{cases} 1 & (i,j) = (s,t) \\ 1 & (i,j) \in P_+ \\ -1 & (i,j) \in P_- \\ 0 & \text{otherwise.} \end{cases}$
```

It is easy to verify that Algorithm 7.6.1 requires $\mathcal{O}(n)$ operations in the worst case since $|C_{st}|$ could be as large as n.

7.6.2 Computing Cycle Matrix Vector Multiplications

The efficiency of a matrix vector multiplication algorithm plays an important role in several algorithms. Two nontrivial algorithms are given here that theoretically computes the cycle matrix vector product much faster than the straightforward implementations that explicitly use the cycle matrix, Γ.

Recall that $G = (N, A)$ is a directed graph, $T \subseteq A$ a spanning intree to a the root $r \in N$ and that Γ is the cycle matrix w.r.t. T. Each column in Γ corresponds to an oriented cycle uniquely determined by an arc not in T. These cycles can be represented by the arc, (s,t) say, and two paths as follows

$$C_{st} = (s,t) \cup P_+ \cup P_-. \tag{7.67}$$

The entries in the column are 1 for (s,t) and arcs in P_+ and -1 for arcs in P_-. We first consider how to compute $y = \Gamma' x$ and then $y = \Gamma x$.

Computing the Cycle Matrix Transpose Vector Product

It is possible to determine the product $y = \Gamma' x$ for an $x \in \mathbb{R}^A$ very efficienctly if we use the following interpretation of $y \in \mathbb{R}^{A \setminus T}$. The (s,t)- component, y_{st}, of y is the length of the fundamental cycle C_{st} where x are the arc lengths and the orientation is taken into account.

Let the fundamental cycle C_{st} be described as in (7.67),

$$C_{st} = (s,t) \cup P_+ \cup P_-. \tag{7.68}$$

This yields

$$\gamma'_{st} x = x_{st} + \sum_{(i,j)\in P_+} x_{ij} - \sum_{(i,j)\in P_-} x_{ij} = x_{st} + \pi_t - \pi_s, \tag{7.69}$$

where the π:s are the length of the associated path to the root. Note that the contribution to $\gamma'_{st}x$ from the common path from the apex to the root is cancelled.

These observations yields the following algorithm that begins by computing the distances from the root to all nodes in the tree, and then compute $\gamma'_{st}x$ from (7.69).

Algorithm 7.6.2. Given $x \in \mathbb{R}^m$, compute $\gamma = \Gamma(T)^T x$.

1. Set the distance to the root of the tree to $\pi_r = 0$.
2. Traverse the tree in depth first order and at every node j do
    - For each $(i, j) \in T$, set $\pi_i = \pi_j + x_{ij}$.
3. Set $y_{st} = w_{st} + \pi_t - \pi_s$, for all $(s, t) \in A \setminus T$.

Remark 7.6. In step 2, nodes are visited in a depth first order, this is not important, any ordering based on the depth of the nodes in the tree will do.

We get the following lemma.

Lemma 7.1
If Γ is the cycle matrix associated with a fundamental cycle basis, then the matrix vector product $\Gamma' x$ can be determined in time complexity $\mathcal{O}(m)$.

Proof: Algorithm 7.6.2 calculates $\Gamma' x$. Step 2 involves n iterations, one for each tree arc. The last step require $(m - n)$ operations, one for each arc outside the tree. □

A small example is used to illustrate how Algorithm 7.6.2 works.

Example 7.6

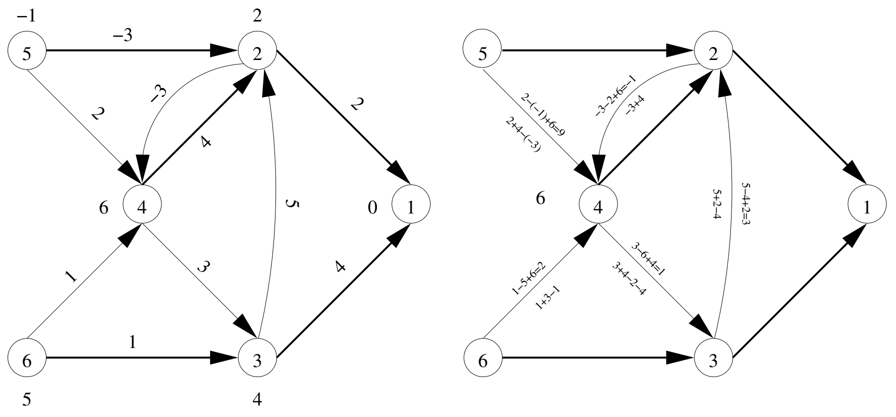


Figure 7.7: The graph, arc lengths, tree distances and fundamental cycle lengths in the example.

Consider the left graph in Figure 7.7 where the thick arcs correspond to the spanning intree T and the length of the arcs are given next to them. This yields

$$\Gamma = \left(\begin{array}{c|ccccc} & 24 & 32 & 43 & 54 & 64 \\ \hline \mathbf{21} & & 1 & -1 & & 1 \\ 24 & 1 & & & & \\ \mathbf{31} & & -1 & 1 & & -1 \\ 32 & & 1 & & & \\ \mathbf{42} & 1 & & -1 & 1 & 1 \\ 43 & & & 1 & & \\ \mathbf{52} & & & & -1 & \\ 54 & & & & 1 & \\ \mathbf{63} & & & & & -1 \\ 64 & & & & & 1 \end{array}\right), \quad x = \left(\begin{array}{c|c} \mathbf{21} & 2 \\ 24 & -3 \\ \mathbf{31} & 4 \\ 32 & 5 \\ \mathbf{42} & 4 \\ 43 & 3 \\ \mathbf{52} & -3 \\ 54 & 2 \\ \mathbf{63} & 1 \\ 64 & 1 \end{array}\right). \tag{7.70}$$

In the second step of Algorithm 7.6.2 the distances from the root are determined. The result is given next to the nodes in the left graph in Figure 7.7. Given these distances, the cycle lengths are calculated in step 3. The values are given in the right graph of Figure 7.7, as the the sum of the contributing arc lengths and as the formula in step 3.

From Figure 7.7 we get

$$\Gamma x = \left(\begin{array}{c|c} 24 & -1 \\ 32 & 3 \\ 43 & 1 \\ 54 & 9 \\ 64 & 2 \end{array}\right), \tag{7.71}$$

which is also obtained from (7.70).

Computing the Cycle Matrix Vector Product

Now consider the computation of $y = \Gamma x$ for an $x \in \mathbb{R}^{A \setminus T}$. The interpretation of $y \in \mathbb{R}^A$ is as follows. The (i,j)- component, y_{ij}, of y is the aggregated flow on the arc (i,j) when the flow in the fundamental cycle associated with $(s,t) \in A \setminus T$ is x_{st}.

If the fundamental cycle C_{st} are partitioned into forward arcs F_{st} and backward arcs B_{st} as $C_{st} = F_{st} \cup B_{st}$, we get

$$y_{ij} = \sum_{(s,t)\in A\setminus T:(i,j)\in F_{st}} x_{st} - \sum_{(s,t)\in A\setminus T:(i,j)\in B_{st}} x_{st} \tag{7.72}$$

The following observations are the foundation of the algorithm.

- For an arc (s,t) not in T the contribution to the aggregated flow comes solely from the cycle C_{st}, hence,

$$y_{ij} = x_{ij}, \text{ for all } (i,j) \in A \setminus T. \tag{7.73}$$

- The net flow into node i is refered to as the node imbalance and is determined by

$$\pi_i = \sum_{j:(i,j)\in A\setminus T} x_{ij} - \sum_{j:(j,i)\in A\setminus T} x_{ji}, \tag{7.74}$$

  for all $i \in N$.

- The node imbalance at i must be propagated through the path from i towards the root since all node balances must be 0. This yields that the aggregated flow on a tree arc (i,j) becomes the total sum of all node imbalances from the nodes in the tree T_i of descendents of node i.

$$y_{ij} = \sum_{u\in T_i} \pi_u, \text{ for all } (i,j) \in T. \tag{7.75}$$

The calculations in the last observation above can be done in linear time if one propagates the imbalance from the leaves towards the root. This yields the following algorithm.

Algorithm 7.6.3. Given $x \in \mathbb{R}^{A\setminus T}$, compute $y = \Gamma x$.

1. Let $y_{ij} = x_{ij}$, for all $(i,j) \in A \setminus T$.
2. Let $\pi_i = \sum_{j:(i,j)\in A\setminus T} x_{ij} - \sum_{j:(j,i)\in A\setminus T} x_{ji}$, for all $i \in N$.
3. Set $\bar{T} = T$. Repeat until $\bar{T} = \emptyset$:
   - Select a leaf, i, in $\bar{T}$, and the unique arc $(i,j) \in \bar{T}$.
   - Set $y_{ij} = \pi_i$ and $\pi_j = \pi_j + \pi_i$.
   - Remove (i,j) from $\bar{T}$.

This algorithm yields the following lemma.

Lemma 7.2
If Γ is the cycle matrix associated with a fundamental cycle basis, then the matrix vector product Γx can be determined in time complexity $\mathcal{O}(m)$.

Proof: Algorithm 7.6.3 calculates Γx. Step 1 and 2 involves $3(m-n)$ arc operations, since in step 2 all arcs in $A \setminus T$ are required twice. The last step requires n arc operations. □

We illustrate the algorithm with a small example.

Example 7.7

Consider the left graph in Figure 7.8, where the thick arcs corresponds to the spanning intree T. The flow in the fundamental cycles are indicated by the number on the thin arcs and the flow on tree arcs is written as a sum of flows. This yields

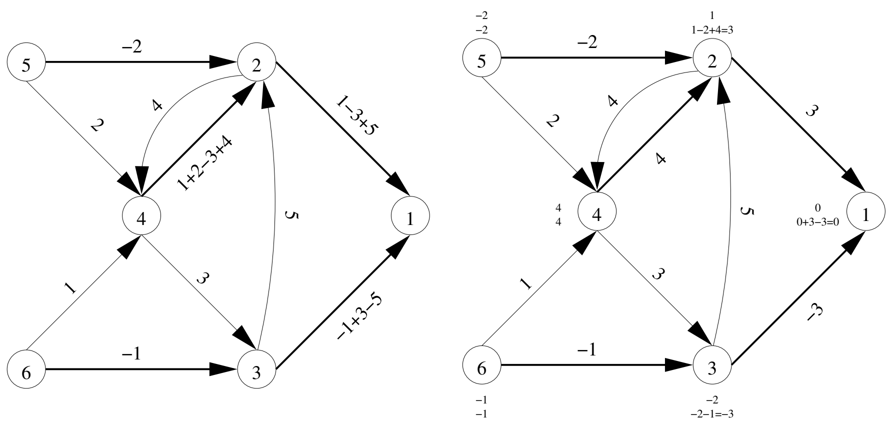


Figure 7.8: The graph, flows and node imbalances in the example.

$$\Gamma = \left(\begin{array}{c|ccccc} & 24 & 32 & 43 & 54 & 64 \\ \hline \mathbf{21} & & 1 & -1 & & 1 \\ 24 & 1 & & & & \\ \mathbf{31} & & -1 & 1 & & -1 \\ 32 & & 1 & & & \\ \mathbf{42} & 1 & & -1 & 1 & 1 \\ 43 & & & 1 & & \\ \mathbf{52} & & & & -1 & \\ 54 & & & & 1 & \\ \mathbf{63} & & & & & -1 \\ 64 & & & & & 1 \end{array}\right), \quad x = \begin{pmatrix} x_{24} \\ x_{32} \\ x_{43} \\ x_{54} \\ x_{64} \end{pmatrix} = \begin{pmatrix} 4 \\ 5 \\ 3 \\ 2 \\ 1 \end{pmatrix}. \qquad (7.76)$$

In step 2 of Algorithm 7.6.3 the imbalances at all nodes are determined. In the last step, the aggregated node imbalance at all nodes are determined along with the flow on the tree arcs. The values are given in the right graph in Figure 7.8. The topmost number adjacent to the corresponing node is the individual imbalance due to the net flow into the node and the number at the bottom is the aggregated imbalance due to propagation.

From Figure 7.8 it is clear that

$$\Gamma x = \left(\begin{array}{c|c} \mathbf{21} & 3 \\ 24 & 4 \\ \mathbf{31} & -3 \\ 32 & 5 \\ \mathbf{42} & 4 \\ 43 & 3 \\ \mathbf{52} & -2 \\ 54 & 2 \\ \mathbf{63} & -1 \\ 64 & 1 \end{array}\right), \qquad (7.77)$$

which is easily verified from (7.76).

A comment about the above algorithms may be in place. Despite the fact that the algorithms are theoretically superior to the straightforward matrix-vector multiplication implementations it is not certain that they actually improve computational performance. One has to investigate for which graphs it is actually worthwhile to implement the algorithms.

8 Shortest Path Routing Design

THIS chapter is devoted to the study of shortest path routing design (SPRD) problems. In the background, it was mentioned that it is easy to determine the routing and traffic flow induced by a metric, cf. Section 2.2.2 in Chapter 2. This is the foundation of several heuristics, but here we will only consider exact solution methods based on mixed integer linear programming (MILP).

The modelling of SPRD problems is first considered in Section 8.1. The approach used here does not include weight variables which implies that feasible routing patterns have to be modelled using only design variables. This issue is adressed in Section 8.2. In particular, a combinatorial description of valid inequalities based on prohibiting routing conflicts is given in Section 8.2.2. The set of partial feasible routing patterns is also described via the conflict hypergraph associated with an independence system and transitive packings, respectively, in Section 8.2.4. Finally, the important issue of separation of valid inequalities is considered in Section 8.3 where some efficient separation algorithms are given for cuts associated with at most two SP-graphs.

8.1 Modelling SPRD Problems

A summary of the introduction to SPRD problems in Section 2.2.3 in Chapter 2 is first given here. A common property of all SPRD problems is that all traffic must be routed along shortest paths. Further, if there are several shortest path, it is required that the flow is evenly split in accordance with the equal cost multi-path (ECMP) principle. The seemingly natural, but appearantly naive, approach to model an SPRD problem is to explicitly simulate the protocol by introducing link weight variables. This yields a set of bilinear constraints, cf. Equation (2.1) on page 15. However, when these constraints are linearized via big-M:s, cf. Equation (2.2), the *associated LP-relaxation becomes extremely weak*.

Currently, a commonly used modelling approach is to exclude the weigth variables from the model and replace the shortest path routing (SPR) constraints with shortest path compatibility constraints that only involve the binary design variables. This is the approach that will be used here as well.

There are basically two choices of binary design variables for an SPRD formulation: path variables or arc variables. For the single path case where all shortest paths are unique, both choices have their benefits, but in the ECMP case, it seems much more natural to use arc variables instead of variables for path sets. Therefore, only arc formulations are considered below; except from some remarks, in particular the following about properties of the two approaches that must be taken into consideration.

Remark 8.1. First consider a path based formulation of an SPRD problem. Given a destination it seems most natural to specify the shortest paths (or set of paths in the ECMP case) from each node to a given destination. An advantage with this approache is that realizability can be determined in polynomial time since it is equivalent to partial compatibility, cf. Chapter 4. Consider an arc based formulation approach instead. This implies that the SP-arcs and the non-SP-arcs are specified via generalized SP-graphs. Suppose that some of them are disconnected. This implies that it is in general NP-complete to determine if the generalized SP-graphs are realizable, cf. Chapter 4, which may affect the pruning of an enumeration tree. However, also note that a disconnected SP-graph contains some additional information that is not available in a path based formulation. Namely, each generalized SP-graph will eventually be completed with some path from each node with outdegree 0 to its respective destination, cf. Chapters 4 and 5 and in particular the definition of the destination arcs $\tilde{D}$ in Equation (5.3). This information may be crucial to discover potential conflicts in advance which could improve the overall computational burden significantly.

Let us now present a matematical model that constitue the core of most recent SPRD models. As mentioned earlier, the focus is on the SPR and ECMP aspect of SPRD only. Therefore, no objective function, nor any other constraints, are included in model (SPRD-C) below.

Let $G = (N, A)$ be a simple, strongly connected directed graph, $L \subseteq N$ the set of destinations and, $N_l \subseteq N \setminus \{l\}$ the set of origins for a destination l. For each destination, l, the traffic demand to be routed from the origin, $k \in N_l$, to the destination is d_{kl}.

Two types of variables are used: binary SPR design variables, y, and fractional flow variables, x. For each destination, $l \in L$, and each arc $(i, j) \in A$, the binary design variable y^l_{ij} should be

$$y^l_{ij} = \begin{cases} 1 & \text{if } (i,j) \text{ is on a shortest simple path to destination node } l, \\ 0 & \text{otherwise.} \end{cases} \tag{8.1}$$

Recall that OSPF weights are greater than or equal to 1, therefore all shortest paths are simple. The flow variables, x, are defined for each destination, $l \in L$, origin, $k \in N_l$, and arc $(i,j) \in A$. Let x_{ij}^{kl} be the fraction of the traffic demand, d_{kl}, from k to l that is routed along (i,j).

The binary design variables determine which arcs that are on shortest simple paths to a given destination, l. Therefore, it is clear that the graph induced by all design variables associated with the destination equal to 1 is an acyclic ingraph to node l since it is the union of all shortest path trees to l. In particular, this induced graph must also contain as a subgraph a reverse arborescence rooted at l that spans all nodes in N_l. To formalize the above, the function, $Y^l(w) : \mathbb{Q}^A \rightarrow \mathbb{B}^A$, that maps a vector of link weights to an incidence vector of an ingraph to node l is introduced. Let

$$Y^l(w)_{ij} = \begin{cases} 1 & \text{if } (i,j) \text{ is in some simple shortest path (w.r.t. } w\text{) to node } l, \\ 0 & \text{otherwise.} \end{cases} \tag{8.2}$$

In words, the graph induced by $Y^l(w)$ is the union of all reversely spanning shortest path trees rooted at l that can be obtained from the link weight vector w. The set of all incidence vectors corresponding to a spanning acyclic ingraph to node l is denoted by $\mathcal{I}_l \subset \mathbb{B}^A$. Using the function $Y^l(w)$, or Equation (8.2), the set $\mathcal{I}_l$ is obtained as

$$\mathcal{I}_l = \left\{ y^l \in \mathbb{B}^A \mid \text{there exist a } w \in \mathbb{N}^A \text{ such that } y^l = Y^l(w) \right\}. \tag{8.3}$$

The set of incidence vectors of a collection of ingraphs to all destinations is

$$\mathcal{I} = \left\{ y = \left(y^l\right)_{l \in L} \mid y^l \in \mathcal{I}_l \text{ for all } l \in L \right\}. \tag{8.4}$$

To describe the set of feasible routing patterns it is of course not sufficient that all design variables describe ingraphs. It is required that *the ingraphs are simultaneously realizable in an SPR protocol.* Denote the subset of incidence vectors corresponding to simultaneously feasible ingraphs by $\mathcal{Y} \subset \mathcal{I}$. In a similar manner as above, the set of of feasible routing patterns can by defined as

$$\mathcal{Y} = \left\{ y \in \mathcal{I} \mid \text{there exist a } w \in \mathbb{N}^A \text{ such that } y^l = Y^l(w) \text{ for all } l \in L \right\}. \tag{8.5}$$

Note that *all ingraphs are obtained from the same vector of link weights* in the definition of $\mathcal{Y}$ in (8.5).

A simple, but very naive, procedure to express $\mathcal{Y}$ in terms of linear constraints is described here. For a binary vector $\bar{y} \notin \mathcal{Y}$, form the pair, $(\chi(\bar{y}), \bar{\chi}(\bar{y}))$, where

$$\chi(\bar{y}) = \{(i,j,l) : \bar{y}_{ij}^l = 1\} \quad \text{and} \quad \bar{\chi}(\bar{y}) = \{(i,j,l) : \bar{y}_{ij}^l = 1\} \tag{8.6}$$

and use the linear inequalities

$$\sum_{(i,j,l) \in \chi} (1 - y_{ij}^l) + \sum_{(i,j,l) \in \bar{\chi}} y_{ij}^l \geq 1 \quad \chi = \chi(\bar{y}),\ \bar{\chi} = \bar{\chi}(\bar{y}),\ \bar{y} \in \mathcal{I} \setminus \mathcal{Y} \tag{8.7}$$

to exclude all points not in $\mathcal{Y}$.

The crude modelling of $\mathcal{Y}$ in (8.7) is obviously not applicable in practice. In Sections 8.3 and 8.2, stronger linear inequality formulations of the sets $\mathcal{I}_l$ and $\mathcal{Y}$ are developed by using the infeasibility of the inverse shortest path problems from Chapters 4 and 5. However, (8.7) suffice to give the following mathematical model of the core of SPRD problems

$$\begin{array}{rcll}
\sum\limits_{(j,i)\in A} x_{ji}^{kl} - \sum\limits_{(i,j)\in A} x_{ji}^{kl} & = & b_i^{kl} & i \in N,\ k \in N_l,\ l \in L \\
x_{ij}^{kl} & \leq & y_{ij}^l & (i,j) \in A,\ k \in N_l,\ l \in L \\
x_{ij}^{kl} - x_{ij'}^{kl} + y_{ij'}^l & \leq & 1 & (i,j) \in A,\ (i,j') \in A,\ k \in N_l,\ l \in L \\
\sum\limits_{(i,j,l)\in\chi} (1 - y_{ij}^l) + \sum\limits_{(i,j,l)\in\bar{\chi}} y_{ij}^l & \geq & 1 & \chi = \chi(\bar{y}),\ \bar{\chi} = \bar{\chi}(\bar{y}),\ \bar{y} \in \mathcal{I} \setminus \mathcal{Y} \\
0 \leq x_{ij}^{kl} & \leq & 1 & (i,j) \in A,\ k \in N_l,\ l \in L \\
y_{ij}^l & \in & \mathbb{B} & (i,j) \in A,\ k \in N_l,\ l \in L,
\end{array} \tag{SPRD-C}$$

where,

$$b_i^{kl} = \begin{cases} -1 & \text{if } i = k \\ 1 & \text{if } i = l \\ 0 & \text{otherwise.} \end{cases} \tag{8.8}$$

An explanation and a motivation of correctness of model (SPRD-C) were given on page 17 in Chapter 2. Most of the constraints are common in network models, the exceptions being the traffic split constraints and the SPR compatibility constraints. The split constraints make all strictly positive flow on emanating arcs from a certain node equal. The SPR constraints excludes binary vectors not in the compability set $\mathcal{Y}$ which guarantees that there are no routing conflicts between ingraphs.

Remark 8.2. Two comments about (SPRD-C) are justified. First, the routing compatibility constraints make sure that there are no directed cycles, therefore the node balance and coupling constraints guarantee that all paths are simple and that $y \in \mathcal{I}$. Second, note that model (SPRD-C) forces the ingraph to destination l to span all nodes, even the nodes that are not origins associated with destination l. This is not a problem since any non-spanning ingraph is augmentable to a spanning ingraph by simply using some compatible metric.

Before the outline of the branch and cut approach to solving SPRD problems is given we comment on the complexity of optimizing SPRD problems and in particular over the polytope induced by (SPRD-C).

8.1.1 SPRD Problems are Hard

Several results on the computational complexity of SPRD related problems have been reported on in the litterature, e.g. [81, 49, 10].

In [80, 81] 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 [49, 50] it is shown that it is NP hard to approximate an optimal routing given demands and a piecewise linear cost function (that implicitly handles the arc capacities). Three versions of the single path case are considered in [10] where it is shown that it is NP hard to approximate an optimal SPR, w.r.t. three natural objectives. Here we prove that it is NP hard to optimize over (SPRD-C). That is, to solve many SPRD problems even without the arc capacity constraints.

Theorem 8.1
Let $c \in \{0, \pm 1\}^{A \times L}$. Then the problem

$$\max c'y, \quad s.t. \ (x, y) \ solves \ \text{(SPRD-C)} \tag{8.9}$$

is NP hard.

Proof: Consider an instance of the realizability problem, that is, given a family of generalized SP-graphs, $\tilde{\mathcal{A}}_L = \{A_l, \bar{A}_l\}_{l \in L}$, decide if $\tilde{\mathcal{A}}_L$ is realizable. In Theorem 4.5 in Chapter 4 it was shown that the realizability problem is NP-complete. Now form c as follows. If $(i, j) \in A_l$, let $c^l_{ij} = 1$ and if $(i, j) \in \bar{A}_l$, let $c^l_{ij} = -1$, otherwise let $c^l_{ij} = 0$. Let z^* denote the optimal value of (8.9) 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 $\bar{A}_l$) must be 0. This implies that $\tilde{\mathcal{A}}_L$ 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. That is, $\tilde{\mathcal{A}}_L$ is not realizable. This proves that $\tilde{\mathcal{A}}_L$ is realizable if and obly if $z^* = M$, and that it is NP hard to solve (8.9). □

Corollary 8.1
If $c \in \mathbb{R}^{A \times L}$, then problem (8.9) *is NP hard.*

It follows immediately that problem (8.9) is NP hard also in the single path case since the realizability results in Chapter 4 holds also without splitting.

Corollary 8.2
If $c \in \mathbb{R}^{A \times L}$, then problem (8.9) *is NP hard also in the single path case.*

Note that there is a significant difference between our results and the NP completeness results in [81, 49, 10]. The proof in [81] indicates that the *ECMP splitting and capacity* part of SPRD problems is the source of hardness, the SPR protocol aspect is actually not considered at all since only a single SP-graph is used. The splitting and capacity are central also in [49], but now also with the SPR aspect taken into account. The single path case problems in [10] involve arc capacities and the proof also include the SPR aspect. Our results show that the routing in accordance with an SPR protocol is a source of hardness in SPRD problems alone both with and without traffic splitting.

8.1.2 A Branch and Cut Approach to SPRD Problems

Since SPRD problems belong to the class of NP hard combinatorial problems it is natural to approach them by the implicit enumertaion techniques commonly used in integer and combinatorial optimization. The foundation of general branch and cut algorithms to a great extent relies on efficient (re-)optimization of the LP relaxations. The success of these methods tailored to specific classes of problems depend on several factors. In particular, the efficiency often depends on the quality of upper and lower bounds, the quality of cuts and also efficient solution of the separation problems for the most important cuts.

Primal bounds can be obtained by applying feasibility heuristics by some of the methods considered earlier and dual bounds are obtained from the LP relaxations. The bound quality aspect is not considered here even though this issue is of great importance, cf. Chapter 9 in [9]. We choose to focus on cuts and how to efficiently separate some classes of cuts here.

Since the naive SPR inequalities in (8.7) are very weak they will be replaced by stronger linear inequality descriptions of the sets $\mathcal{I}$ and $\mathcal{Y}$. This still yields an exponential number of constraints in the formulation of (SPRD-C) so constraint generation must be incorporated into the branch and cut scheme. In the remaining part of this chapter we present the stronger formulations of $\mathcal{I}$ and $\mathcal{Y}$ and consider how to separate the associated inequalities.

Similar branch and cut approaches are used in practice, e.g. in [9, 13, 79, 14].

8.2 The Set of Feasible Routing Patterns

In [9], the collection of path sets that are simultaneously realizable in the single path case is described as an independence system and also in terms of the associated conflict hypergraph. A similar approach is taken on here. We consider the arc formulation for the partial realizability case with ECMP. This actually yields some significant differences.

It was established in the previous couple of chapters that a set of spanning ingraphs are realizable if and only if they are partially realizable (or equivalently, partially compatible since there are no destination arcs in the spanning case). It was proved that partial realizability is not sufficient for realizability for non-spanning and disconnected ingraphs and that the determination of realizability is actually NP-complete. Further, it was shown that a family of SP-graphs is partially realizable if and only if it does not exist a routing conflict (in the sense used in Remark 4.1 in Chapter 4). Because of these facts, the following alternative description of the set of feasible routing patterns, $\mathcal{Y}$, will be considered,

$$\mathcal{Y} = \mathcal{Y}^* \cap \mathcal{I}. \tag{8.10}$$

Here, $\mathcal{Y}^*$ is the set of (possibly fractional) vectors such that the induced family of generalized SP-graphs does not contain a routing conflict. That is, the family is partially realizable. A formal definition is given in Section 8.2.1.

Since partial realizability is computationally tractable and also a necessary and sufficient condition for realizability for spanning ingraphs, the intersection in (8.10) is a valid and suitable description of $\mathcal{Y}$. Therefore, it is feasible to proceed by analyzing the polytopes associated with $\mathcal{I}$ and $\mathcal{Y}^*$ instead of $\mathcal{Y}$, this is the main issue in Sections 8.2.1 through

8.2.5. Obviously, it suffice to consider the ingraph polytope for a single destination at a time instead of $\mathcal{I}$. A breif analysis is performed in Section 8.2.5.

Recall from Remark 8.2 that $\mathcal{I}$ is already modelled via the flow and coupling constraints. Therefore, the intersection in (8.10) yields that the crude conflict inequalities in (SPRD-C) that prohibit points not in $\mathcal{Y}$ can be replaced by valid inequailities for the convex hull of $\mathcal{Y}^* \cap \mathbb{B}^{A\times L}$. Denote this set by $\mathcal{P} = \mathcal{Y}^* \cap \mathbb{B}^{A\times L}$.

8.2.1 Vectors that Induce Partially Realizable SP-graphs

The partial realizability models from Chapters 4 and 5 can be used to describe $\mathcal{Y}^*$ implicitly. The characterization of forcing and infeasible routing patterns in Chapther 5 makes it straightforward to develop an integer linear inequality description of $\mathcal{P}$ by prohibiting the existance of improving families of cycles. Similar approaches have been presented in [14, 96, 83].

For notational convenience, let $B = [0,1]^{A\times L}$. Given a vector $y \in B$, let $A^0(y)$ and $A^1(y)$ be the index sets of arc destination pairs where y is 0 and 1, respectively. That is,

$$A^0(y) = \{((i,j),l) \in A \times L \mid y^l_{ij} = 0\} \tag{8.11}$$

and

$$A^1(y) = \{((i,j),l) \in A \times L \mid y^l_{ij} = 1\}. \tag{8.12}$$

Also let $A^0_l(y)$ and $A^1_l(y)$ be the part of $A^0(y)$ and $A^1(y)$, respectively, associated with destination l. Given y, define the set $\tilde{D}_l(y)$ in the obvious manner,

$$\tilde{D}_l(y) = \left\{(i,l) : \delta^+(i) \cap A^1_l(y) = \emptyset\right\}, \tag{8.13}$$

and let $\tilde{D}(y) = \bigcup_{l\in L} \tilde{D}_l(y)$ and $\tilde{A}(y) = A \cup \tilde{D}(y)$. This yields the associated partial realizability instance, (PR(y)), induced by y,

$$\begin{array}{rcll}
\tilde{w}_{ij} + \pi^l_i - \pi^l_j & = & 0 & (i,j) \in A^1_l(y) \cup \tilde{D}_l(y),\ l \in L \\
\tilde{w}_{ij} + \pi^l_i - \pi^l_j & \geq & 1 & (i,j) \in A^0_l(y),\ l \in L \\
\tilde{w}_{ij} + \pi^l_i - \pi^l_j & \geq & 0 & (i,j) \in \tilde{A}(y),\ l \in L \\
\tilde{w}_{ij} & \geq & 1 & (i,j) \in \tilde{A}(y).
\end{array} \tag{PR(y)}$$

This yields an implicit definition of $\mathcal{Y}^*$ as

$$\mathcal{Y}^* = \{y \in B \mid \text{ model (PR}(y)\text{) is feasible for } y\}. \tag{8.14}$$

Note that $\mathcal{Y}^*$ is not a closed set since all binary points are limit points, but not all of them are actually contained in $\mathcal{Y}^*$. Therefore, the closure of $\mathcal{Y}^*$ is B. However, since the real interest lies in (the convex hull of) $\mathcal{P} = \mathcal{Y}^* \cap \mathbb{B}^{A\times L}$, this is of no concern. Also observe that

$$\mathcal{Y} = \mathcal{Y}^* \cap \mathcal{I} = \mathcal{Y}^* \cap \mathbb{B}^{A\times L} \cap \mathcal{I} = \mathcal{P} \cap \mathcal{I}. \tag{8.15}$$

Since (PR(y)) is feasible if and only if the induced family of generalized SP-graphs contains no improving family of cycles it is sufficient to prohibit these to describe $\mathcal{P}$. Alternatively, one can essentailly say that the support of an unbounded dual Farkas ray of (PR(y)) can be used to construct a valid inequality for the convex hull of $\mathcal{P}$.

8.2.2 Valid Inequalities From Routing Conflicts

As in [14, 96] pairs of arc destination subsets, $(Q, \bar{Q})$, are considered. That is, $Q, \bar{Q} \subset A \times L$. If the induced family of partial SP-graphs are not realizable, then

$$\sum_{((i,j),l)\in Q} (1 - y_{ij}^l) + \sum_{((i,j),l)\in \bar{Q}} y_{ij}^l \geq 1 \tag{8.16}$$

is clearly a valid inequality for conv $\mathcal{Y}$. Due to the complexity of realizability, it is unlikely that there exists a good characterization of the pairs of subsets, Q and $\bar{Q}$, that make (8.16) valid. However, limiting the scope to routing conflicts that arise from partial unrealizability implies that the combinatorial characterization of forcing and infeasible routing patterns in Chapther 5 can be used. This yields a comprehensive and combinatorial description of the pairs $(Q, \bar{Q})$ where (8.16) constitue a valid inequality for conv $\mathcal{P}$, and thus also for conv $\mathcal{Y}$. Note that this important subfamily is not complete. But, it is sufficient as long as the intersection with the set $\mathcal{I}$ is considered, that is, when the ingraphs are spanning.

From now on, we only consider the pairs, $(Q, \bar{Q})$, that correspond to routing conflicts due to partial unrealizability. Denote the family of all such pairs by $\mathcal{Q}$. This yields the following naive integer linear formulation of $\mathcal{P}$.

$$\begin{array}{rcll} \displaystyle\sum_{((i,j),l)\in Q} (1 - y_{ij}^l) + \sum_{((i,j),l)\in \bar{Q}} y_{ij}^l & \geq & 1 & (Q, \bar{Q}) \in \mathcal{Q} \\ y_{ij}^l & \in & \mathbb{B} & (i,j) \in A,\ l \in L. \end{array} \tag{$\mathcal{P}_Q$}$$

Obviously, not all inequalities in ($\mathcal{P}_Q$) are necessary. Let us investigate the inequalities closer and (try to) determine which of them that are necessary, or, at least give some conditions for necessity.

Let $\mathcal{C}$ be a feasible family of cycles,

$$\mathcal{C} = \left\{\mathcal{C}^l\right\}_{l\in L}, \tag{8.17}$$

where

$$\mathcal{C}^l = \left\{ C_k^l \subset \tilde{A} \mid C_k^l = F_k^l \cup B_k^l,\ B_k^l \subseteq A_l \cup \tilde{D}_l \right\}. \tag{8.18}$$

If all the design variables associated with the backward arcs in $\mathcal{C}$ are 1, then $\mathcal{C}$ is feasible and corresponds to a potential conflict. The design vector may still be feasible depending on if the induced cycle family is improving or not.

To express our valid inequalities smoothly, it is convenient to collect the indices of the forward and backward arcs associated with a feasible cycle family $\mathcal{C}$. Let

$$\tilde{F}(\mathcal{C}) = \left\{((i,j),l) \in \tilde{A} \times L \mid (i,j) \in F_k^l \text{ for some } k\right\}, \tag{8.19}$$

and

$$\tilde{B}(\mathcal{C}) = \left\{((i,j),l) \in \tilde{A} \times L \mid (i,j) \in B_k^l \text{ for some } k\right\}. \tag{8.20}$$

Also define the arc sets associated with a certain destination,

$$\tilde{F}_l(\mathcal{C}) = \left\{(i,j) \in \tilde{A} \mid ((i,j),l) \in \tilde{F}(\mathcal{C})\right\}, \tag{8.21}$$

and

$$\tilde{B}_l(\mathcal{C}) = \left\{(i,j) \in \tilde{A} \mid ((i,j),l) \in \tilde{B}(\mathcal{C})\right\}. \tag{8.22}$$

Note that a cycle in $\mathcal{C}$ may contain destination 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. Let

$$B(\mathcal{C}) = \tilde{B}(\mathcal{C}) \cap (A \times L) \quad \text{and} \quad F(\mathcal{C}) = \tilde{F}(\mathcal{C}) \cap (A \times L), \tag{8.23}$$

and also let,

$$B_l(\mathcal{C}) = \tilde{B}_l(\mathcal{C}) \cap A \quad \text{and} \quad F_l(\mathcal{C}) = \tilde{F}_l(\mathcal{C}) \cap A. \tag{8.24}$$

Equations (8.23) and (8.24) 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 cuts.

Remark 8.3. When the destination arcs are omitted, the collections of arc destination pairs, $B(\mathcal{C})$ and $F(\mathcal{C})$, no longer induce cycles. This implies that the structure is not as clearly revealed as in Chapter 5 before the projection.

Now consider when the cycle family $\mathcal{C}$ is improving. Begin with the non-saturating case. From Proposition 5.2 it is clear that $\mathcal{C}$ is always improving. Therefore, if $y \in \mathcal{P}$, all design variables associated with the backward arcs in $\mathcal{C}$ must not be 1. That is,

$$\min_{((i,j),l) \in B(\mathcal{C})} y_{ij}^l = 0. \tag{8.25}$$

The convexification of (8.25), yields the following linear valid inequality.

Proposition 8.1
Let $\mathcal{C}$ be a non-saturating family of cycles. Then, the inequality

$$\sum_{l \in L} \sum_{(i,j) \in B_l(\mathcal{C})} \left(1 - y_{ij}^l\right) \geq 1 \tag{8.26}$$

is valid for $\mathcal{P}$.

In the saturating case, Proposition 5.3 states that a saturating solution, θ say, is improving if and only if there is a commodity l and an arc (i,j) such that $(i,j) \in A_l^0(y)$ and $\theta_{ij}^l > 0$. 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; if $\mathcal{C}$ is saturating and all backward arc design variables are 1, then all forward arc design variables must also be 1. That is,

$$\min_{((i,j),l)\in B(\mathcal{C})} y_{ij}^l \le \min_{((i,j),l)\in F(\mathcal{C})} y_{ij}^l. \tag{8.27}$$

Again, the convexification of (8.27) yields a set of linear valid inequalities.

Proposition 8.2
Let $\mathcal{C}$ be a saturating family of cycles. Then, the inequalities

$$y_{i'j'}^{l'} + \sum_{l\in L} \sum_{(i,j)\in B_l(\mathcal{C})} \left(1 - y_{ij}^l\right) \ge 1, \quad (i',j') \in F_{l'}(\mathcal{C}), l' \in L, \tag{8.28}$$

are valid for $\mathcal{P}$.

The propositions above yield that the necessary pairs $(Q,\bar{Q}) \in \mathcal{Q}$ all have $|\bar{Q}| \le 1$. It is possible to restrict the required pairs further by only considering irreducible solutions.

Proposition 8.3
Let $\mathcal{C}$ be a family of cycles and (8.28) *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, it follows from Definition 5.11 on page 89 that there exists a family of cycles $\widetilde{\mathcal{C}}$, such that $B(\widetilde{\mathcal{C}}) \subset B(\mathcal{C})$. Clearly, an inequality associated with $B(\mathcal{C})$ is dominated by a (corresponding) inequality associated with $B(\widetilde{\mathcal{C}})$. □

To obtain a linear inequality descripion of $\mathcal{P}$ it is sufficient to enumerate all feasible families of cycles and include the valid inequalities from Proposition 8.1 and 8.2. Let $\mathcal{Q}_I$ and $\mathcal{Q}_{NS}$ be the collections of all irreducible and feasible saturating and non-saturating families of cycles, respectively.

Proposition 8.4
The incidence vector $y \in \mathbb{B}^{A\times L}$ lies in $\mathcal{P}$, if and only if it is a feasible solution to the following system

$$\begin{array}{rcll} \sum_{l\in L} \sum_{(i,j)\in B_l(\mathcal{C})} \left(1 - y_{ij}^l\right) & \ge & 1 & \mathcal{C} \in \mathcal{Q}_{NS} \\ y_{i'j'}^{l'} + \sum_{l\in L} \sum_{(i,j)\in B_l(\mathcal{C})} \left(1 - y_{ij}^l\right) & \ge & 1 & ((i',j'),l') \in F(\mathcal{C}),\ \mathcal{C} \in \mathcal{Q}_I. \end{array} \tag{$\mathcal{P}$}$$

From Corollary 7.3 on page 151 it follows that the inequalities associated with non-saturating families of cycles are not really required to describe $\mathcal{Y}$. Therefore, the relaxation of $\mathcal{P}$ obtained when only inequalities associated with saturated cycle families are used will often be considered. Denote this relaxation by $\mathcal{P}_S$. That is, $y \in \mathcal{P}_S$ if it solves

$$\begin{array}{rcll} y^{l'}_{i'j'} + \sum\limits_{l \in L} \sum\limits_{(i,j) \in B_l(\mathcal{C})} \left(1 - y^l_{ij}\right) & \geq & 1 & ((i',j'),l') \in F(\mathcal{C}),\ \mathcal{C} \in \mathcal{Q}_I \\ y^l_{ij} & \in & \mathbb{B} & (i,j) \in A, \in L. \end{array} \qquad (\mathcal{P}_S)$$

Combining this formulation and spanning ingraph inequalities yields an explicit formulation of $\mathcal{Y}$ that is superior to the crude and naive models above.

It is straightforward to develop an explicit integer linear formulation of the set of spanning ingraphs, $\mathcal{I}$. It suffices to require that all nodes but the root have outdegree at least 1 and that there are no directed cycles. Denote the family of directed cycles not including l in G by $\mathcal{D}_l$.

Proposition 8.5
The incidence vector $y^l \in \mathbb{B}^A$ corresponds to an ingraph to node l, i.e. $y^l \in \mathcal{I}_l$, if and only if it is a feasible solution to the following system

$$\begin{array}{rcll} \sum\limits_{(i,j) \in \delta^+(i)} y^l_{ij} & \geq & 1 & i \neq l,\ i \in N \\ \sum\limits_{(i,j) \in C} y^l_{ij} & \leq & |C| - 1 & C \in \mathcal{D}_l \\ y^l_{ij} & \in & \mathbb{B} & (i,j) \in A \\ y^l_{li} & = & 0 & (l,i) \in A,\ i \in \delta^+(l). \end{array} \qquad (\text{IG}_l)$$

The relaxation of (IG$_l$) without outdegree constraints is known as the acyclic subgraph polytope and has been studied rather extensively in the litterature, e.g. in [55, 60, 70].

To obtain the explicit integer linear formulation for $\mathcal{Y}$ we use

$$\mathcal{Y} = \mathcal{P} \cap \mathcal{I} = \mathcal{P}_S \cap \mathcal{I}. \qquad (8.29)$$

Now Corollary 7.3, Propositions 8.4 and 8.5 yield that $\mathcal{Y}$ is characterized as follows.

Theorem 8.2
A binary vector y is the incidence vector of a set of feasible routing patterns, i.e. $y \in \mathcal{Y}$, if and only if it is a solution of the following system.

$$\begin{array}{rcll} \sum\limits_{(i,j) \in \delta^+(i)} y^l_{ij} & \geq & 1 & i \neq l,\ i \in N,\ l \in L \\ \sum\limits_{(i,j) \in C} \left(1 - y^l_{ij}\right) & \geq & 1 & C \in \mathcal{D}_l \\ y^{l'}_{i'j'} + \sum\limits_{l \in L} \sum\limits_{(i,j) \in B_l(\mathcal{C})} \left(1 - y^l_{ij}\right) & \geq & 1 & ((i',j'),l') \in F(\mathcal{C}),\ \mathcal{C} \in \mathcal{Q}_I \\ y^l_{ij} & \in & \mathbb{B} & (i,j) \in A, \in L. \end{array} \qquad (\mathcal{Y})$$

To further analyze the polytope $\mathcal{Y}$ via $\mathcal{P}_S$ it is possible to interpret the feasible partial routing patterns as transitive packings or to consider the associated conflict hypergraph. This could be very useful when the facial structure is considered.

8.2.3 Independence Systems and Transitive Packings

Independence systems have been studied extensively in the combinatorial optimization litterature. Several solution methods, exact and heuristics, have been considered and much is know about the facial structure of the associated polytope, cf. [77, 69, 37, 43, 72, 51]. An independence system is defined as follows.

Definition 8.1
Let V be a finite base set and $\mathcal{I} \subseteq 2^V$ a collection of subsets of V. The pair $(V,\mathcal{I})$ is an independence system if

$$\emptyset \in \mathcal{E} \quad \textit{and} \quad I \in \mathcal{I},\ J \subset I \Rightarrow J \in \mathcal{I}. \tag{8.30}$$

Remark 8.4. 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 $\mathcal{I}$ are called independent sets and a subset $H \subseteq V$ such that $H \notin \mathcal{I}$ 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 $\mathcal{I}$. Given a set of weights, c_i for all $i \in V$, the independent set problem is to find an independent set $I \in \mathcal{I}$ of maximal weight.

A convenient description of an independence system is via vertex packings in a hypergraph. Denote the set of circuits in the independence system $(V,\mathcal{I})$ by $\mathcal{E}$. Then, the conflict hypergraph, $\mathcal{H}(V,\mathcal{I}) = (V,\mathcal{E})$, is the hypergraph with a vertex for each element in V and a hyperedge for each circuit in $(V,\mathcal{I})$. Now, the vertex packings in the hypergraph, $(V,\mathcal{E})$, correspond to the independent sets in $(V,\mathcal{I})$.

Denoted the incidence matrix of an arbrirary hypergraph $\mathcal{H}$ by $A_{\mathcal{H}}$ and let $p_{\mathcal{H}}$ be the vector where element i is the number of positive entries in row i of $A_{\mathcal{H}}$. That is, $p_{\mathcal{H}} = 1'(A_{\mathcal{H}})^+$. For any weight vector, c, this yields the weighted hypergraph vertex packing problem formulation

$$\begin{array}{lrcl} \max & c'x & & \\ s.t. & A_{\mathcal{H}}x & \leq & p_{\mathcal{H}} - 1 \\ & x & \in & \mathbb{B}^V. \end{array} \tag{HVP}$$

Remark 8.5. Note that any 0/1 matrix can be interpreted as a hypergraph incidence matrix (wlog assume that there are is empty edge and no loop since these cases can trivially be reduced).

If $\mathcal{H}$ is the conflict hypergraph of an independence system, $(V,\mathcal{I})$, then the weighted hypergraph vertex packing problem in $\mathcal{H}$ is equivalent to the independent set problem over $(V,\mathcal{I})$. Further, by construction, all hyperedges in $\mathcal{H}$ correspond to circuits of the independence system and the hypergraph is a so called clutter. That is, no hyperedge is contained in another. Therefore, no row of $A_{\mathcal{H}}$ is trivially dominated in (HVP).

In [76, 75, 88], transitive packings were introduced as a unifying concept in combinatorial optimization. They generalize hypergraph vertex packings by also taking transitive elements into account. This implies that the incidence matrix can be almost any $0/\pm 1$ matrices in (HVP). Clearly, very many combinatorial optimization problems fit into this framework, e.g. the acyclic subgraph problem and the clique partitioning problem, for details see [75].

Let us formally introduce the transitive packing problem in hypergraphs, a minor deviation from the presentation in [75] is made to cover *an arbrirary* $0/\pm 1$ matrix. This implies that even more combinatorial optimization problems fit into the framework, e.g. the set covering, partitioning and packing problems and also the acyclic ingraph problem considered in Theorem 8.3 in Section 8.2.5.

Definition 8.2 (cf. Definition 2.1 in [75])
An extended hypergraph is a triple, $\mathcal{H} = (V, \mathcal{E}, tr)$, consisting of a set of vertices, V, a multiset of hyperedges, $\mathcal{E}$, and a transitivity mapping, $tr : \mathcal{E} \to 2^V$. It also holds for each hyperedge $E \in \mathcal{E}$ that $E \subseteq 2^V$ and $tr(E) \subseteq V \setminus E$.

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 8.3 (cf. Definition 2.1 in [75])
Let $\mathcal{H} = (V, \mathcal{E}, tr)$ be an extended hypergraph. The vertex subset $S \subset V$ is a transitive packing in $\mathcal{H}$ if for every $E \in \mathcal{E}$ where $E \subseteq S$ there is a vartex $i \in S \cap tr(E)$.

Definition 8.4 (cf. Section 2 [75])
Let $\mathcal{H} = (V, \mathcal{E}, tr)$ 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 $\mathcal{H}$ 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 (HVP) yields that (HVP) is a valid formulation of the maximum weight transitive packing problem. A less compact, but more comprehensible, formulation is also given here.

$$\begin{array}{llll} \max & \sum_{i\in V} w_i x_i & & \\ s.t. & \sum_{i\in E} x_i - \sum_{i\in \mathrm{tr}(E)} x_i \le |E| - 1 & E \in \mathcal{E} & \\ & x_i \in \mathbb{B} & i \in V. & \end{array} \tag{TP}$$

Remark 8.6. If there are no transitive elements, that is $\mathrm{tr}(E) = \emptyset$ 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 8.7. Note that restricting the hyperedge cardinality to be at least 2 implies that models (IG_l) and ($\mathcal{Y}$) do not fit into the transitive packing framework, whereas ($\mathcal{P}$) and ($\mathcal{P}_S$) still do. To also include models (IG_l) and ($\mathcal{Y}$), empty hyperedges and loops must also be allowed.

Several results about the transitive packing polytope are given for the case where the edges in $\mathcal{H}$ have cardinality at least 2 in [75]. 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 [60, 70], cf. [75] for details.

Our main interest in transitive packings lies in the special case where all edges are transitively mapped to singletons. That is, for all $E \in \mathcal{E}$ it holds that $\text{tr}(E) = \{u\}$ for some $u \in V$. This restriction yields what we call 1-transitive packings. Refining model (TP) to 1-transitive packings and rewriting yields the following model.

$$\begin{array}{lrcll} \max & \sum_{i \in V} c_i x_i & & & \\ s.t. & x_u + \sum_{i \in E} (1 - x_i) & \geq & 1 & \text{tr}(E) = \{u\},\ E \in \mathcal{E} \\ & x_i & \in & \mathbb{B} & i \in V. \end{array} \tag{1-TP}$$

From above, it can be seen that the models (IG$_l$), ($\mathcal{Y}$), ($\mathcal{P}$), ($\mathcal{P}_S$) and ($\mathcal{P}_Q$) fit into the transitive packing framework. In particular, the models ($\mathcal{P}$) and ($\mathcal{P}_S$) describe 1-transitive packings. Since there is no transitivity in the single path case, the models corresponding to ($\mathcal{P}$) and ($\mathcal{P}_S$) describe ordinary hypergraph packings (independent sets).

8.2.4 The Conflict Hypergraph for Routing Patterns

Here, we explicitly consider hypergraph (transitive) packing formulations associated with $\mathcal{P}_S$ and an associated conflict hypergraph similar to the conflict hypergraph for independence systems. Our reason is that it is sometimes easier to think of routing patterns in this hypergraph representation. Especially, and more importantly, it is more convenient when classes of valid inequalities are derived based on the rich hypergraph structures for hypergraph (transitive) packings in [75], e.g. generalized cycle, generalized clique, generalized antihole, generalized antiweb, and odd partition inequalities.

As previously mentioned, an independence system approach based on collections of path sets and partial compatibility is used in [9]. Here, partial realizability and the arc formulation is considered. Since arcs are used, the (hyper)nodes in our routing pattern conflict hypergraph correspond to SP-arcs and the hyperedges correspond to (irreducible) cycle families. Less obvious is that the conflict hypergraph is no longer a clutter since partial realizability is considered. First, the single path case is considered, and then the ECMP case.

The Single Path Case

To analyze the single path case analogue of $\mathcal{P}_S$ recall from Chapter 5 that a cycle family $\mathcal{C}$ is improving in the single path case except for the particular special case mentioned in Remark 5.8 on page 80. If these non-improving cycle families are omitted from consideration and all improving cycle families are prohibited a relaxation of the single path analogue of $\mathcal{P}_S$ is obtained. Further, since there must be no splitting, it is also reasonable to require that no two SP-arcs emanate from the same node. Denote this modified single path version of $\mathcal{P}_S$ by $\mathcal{P}_{USPS}$. That is, $\mathcal{P}_{USPS}$ is "almost" the set of incidence vectors of partially realizable families of generalized SP-graphs in the single path case. Here, USPS is the abbrevation for Unique Single Path System used in [9].

Let $\mathcal{Q}_{ISP}$ be the collection of all irreducible and improving cycle families in the single path case. This yields a specialization of the valid inequalities to the single path case and gives the following description of $\mathcal{P}_{USPS}$

$$\begin{array}{rcll}\sum\limits_{l\in L}\sum\limits_{(i,j)\in B_l(\mathcal{C})} y_{ij}^l & \leq & |B(\mathcal{C})|-1 & \mathcal{C}\in\mathcal{Q}_{ISP} \\ y_{ij}^l + y_{ik}^l & \leq & 1 & (i,j)\in A,\ (i,k)\in A,\ l\in L \\ y_{ij}^l & \in & \mathbb{B} & (i,j)\in A,\ l\in L.\end{array} \qquad (\mathcal{P}_{USPS})$$

This model clearly shows that the subsets of design variables that correspond to incidence vectors in $\mathcal{P}_{USPS}$ form an independence system since $(\mathcal{P}_{USPS})$ describes a hypergraph vertex packing polytope.

The actual conflict hypergraph, $\mathcal{H}_{SP}$, is formed by introducing a vertex in $\mathcal{H}_{SP}$ for each arc destination pair $((i,j),l)\in A\times L$. Then, a hyperedge is created for each irreducible and improving cycle family. That is, for each $\mathcal{C}\in\mathcal{Q}_{ISP}$, introduce the hyperedge $E\in\mathcal{E}$ where $E=B(\mathcal{C})$. Finally, a hyperedge is created between vertices that share the destination and starting node to avoid splitting, that is, between $((i,j),l)$ and $((i,k),l)$. An example of this construction is given below after a few remarks have been given.

The reason for only considering improving cycle families and no-splitting constraints is to maintain the independence system structure. This could be very useful when the facial structure is considered. Note that the inequalities associated with non-improving cycle families would destroy the independence system structure, cf. Equation (8.31) and (8.32) in Remark 8.8

Remark 8.8. In the single path case all non-improving cycle families are induced by a shortest path, P_{st}, from s to t via l. This yields that a subpath $P_l\subset P_{st}$ to l must also be a shortest path. The induced inequalities become as follows.

$$y_{uv}^l + \sum_{(i,j)\in P_l}\left(1-y_{ij}^t\right)\geq 1 \qquad (u,v)\in P_l,\ P_l\subset P_{st}. \tag{8.31}$$

In particular, the inequality from $P_l=\{(i,l)\}$ yields the important special case,

$$y_{il}^t\geq y_{il}^l \qquad (i,l)\in A,\ l,t\in L. \tag{8.32}$$

Remark 8.9. In model $(\mathcal{P}_{USPS})$ we choose to present the no splitting inequalities via pairs of emanating arcs from a node. This is only to emphazise the independence system structure. In practice these inequalities are very bad since they are dominated by the much stronger induced clique inequalities

$$\sum_{j:(i,j)\in A} y_{ij}^l \le 1 \qquad i \in N,\ l \in L. \tag{8.33}$$

Remark 8.10. Just as in the path formulation case analyzed in [9], it is easy to verify that the independence system $(\mathcal{P}_{USPS})$ is not a matroid in general. Consider the following three elements in $\mathcal{P}_{USPS}$ induced by a graph with at least four nodes. The arc destination pair sets $S_1 = \{((1,2),3),((1,2),4)\}$ and $S_2 = \{((1,3),2)\}$ are clearly independent sets (that is, feasible USPSs). However, neither $\{((1,2),3),((1,3),2)\}$, nor $\{((1,2),4),((1,3),2)\}$, is an USPS since there may be no splitting. Hence, the independence system can not be a matroid.

Example 8.1

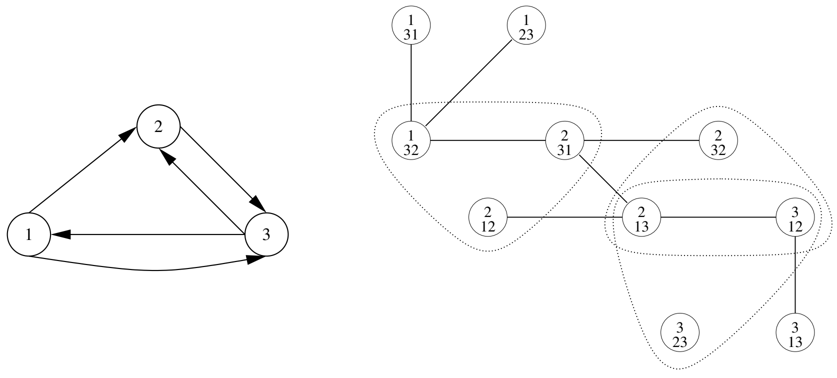


Figure 8.1: *A graph (left) and the conflict hypergraph (rigth) for the instance induced by the graph and all three destinations. The nodes in the hypergraph are labelled with the destination on top of the arc. Each hyperedge corresponds to an SPR conflict.*

Consider the graph with three nodes on the left in Figure 8.1. There are 10 valid arc destination combinations which yields 10 design variables or hypernodes in the associated conflict hypergraph $\mathcal{H}_{SP}$.

The explanation for the existence of the hyperedges is as follows.

Directed cycles in the SP-graphs are prohibited by the hyperedges

$$\begin{aligned}&\{((2,3),1),\ ((3,2),1)\} \text{ and}\\ &\{((1,3),2),\ ((3,1),2)\}.\end{aligned} \tag{8.34}$$

Splitting in the SP-graphs are prohibited by the hyperedges

$$\begin{aligned}&\{((3,1),1),\ ((3,2),1)\},\\ &\{((3,1),2),\ ((3,2),2)\},\\ &\{((1,2),2),\ ((1,3),2)\} \text{ and}\\ &\{((1,2),3),\ ((1,3),3)\}.\end{aligned} \tag{8.35}$$

Routing conflicts between two SP-graphs are prohibited by the hyperedges

$$\begin{array}{l}\{((3,2),1),\ ((1,2),2),\ ((3,1),2)\},\\ \{((1,3),2),\ ((1,2),2),\ ((1,2),3)\}\text{ and}\\ \{((1,3),2),\ ((1,2),2),\ ((2,3),3)\},\end{array}\tag{8.36}$$

which are dominated by the hyperedges

$$\begin{array}{l}\{((1,3),2),\ ((1,2),3)\},\\ \{((3,2),1),\ ((3,1),2)\}\text{ and}\\ \{((3,2),1),\ ((3,1),2)\},\text{respectively.}\end{array}\tag{8.37}$$

The above example shows that the conflict hypergraph, $\mathcal{H}_{SP}$, is in general not a clutter even though only irreducible cycle families are considered. The reason is that the destination arcs may yield dominance. The problem essentially arise since we compare partial compatibility conflicts with the stronger projected partial realizability conflicts.

Unless one is willing to accept that the hypergraph is not a clutter a suggestion is to redefine the concept of irreducibility to take the projection into account. Define

$$\widetilde{\mathcal{Q}}_{ISP} = \{\mathcal{C} \in \mathcal{Q}_{ISP} \mid \text{There exists no } C' \in Q_{ISP} \text{ such that } B(\mathcal{C}') \subset B(\mathcal{C})\}\,. \tag{8.38}$$

Using $\widetilde{\mathcal{Q}}_{ISP}$ instead of $\mathcal{Q}_{ISP}$ yields a conflict hypergraph that is a clutter. This removes some redundant inequalities and seems appropriate in the design context.

Example 8.1: continued

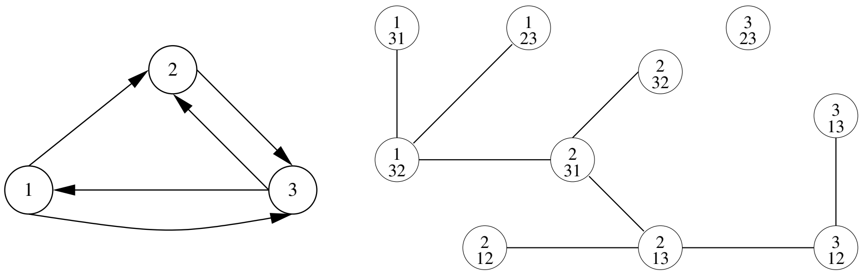


Figure 8.2: *(Left) The graph from Example 8.1. (Right) The conflict hypergraph for the induced instance when the reduced conflict family $\widetilde{\mathcal{Q}}_{ISP}$ is used. The nodes in the hypergraph are labelled with the destination on top of the arc.*

Let us reconsider the previous example when the conflict family $\widetilde{\mathcal{Q}}_{ISP}$ is used instead of $\mathcal{Q}_{ISP}$. There are still 10 hypernodes in the conflict hypergraph $\mathcal{H}_{SP}$ associated with the instance induced by the graph on the left in Figure 8.2 (and 8.1). However, there are now only 8 hyperedges and the conflict hypergraph is a clutter. It is in fact an ordinary loopless graph since all hyperedges have cardinality two, but that is just a coincidence. The explanation for the existence of the hyperedges is the same as before and can be found above in Example 8.1.

Let us now apply the same ideas to the ECMP case.

The ECMP Case

A description of families of generalized SP-graphs where there is no saturating conflict is given in ($\mathcal{P}_S$) on page 179. This system clearly fits into the transitive packing framework. More precicely, it models 1-transitive packings since ($\mathcal{P}_S$) can be written as (1-TP).

Recall that the collection of all feasible and irreducible cycle families is denoted by $\mathcal{Q}_I$ and that some induced inequalities are trivially dominated. In the single path case it was easy to resolve this issue by only using the relevant irreducible conflicts in $\widetilde{\mathcal{Q}}_{ISP}$ defined in (8.38) instead of all irreducible conflicts in $\mathcal{Q}_{ISP}$. In the ECMP case, the transitivity issue makes it more complicated, consider the following example.

Example 8.2

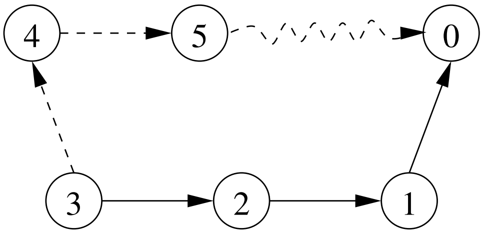


Figure 8.3: *A cycle family where all but one of the associated valid inequalities are dominated. The dashed arcs correspond to design variables associated with destination node 0 and the destination of the solid arcs is unspecified.*

Consider the cycle family $\mathcal{C} \in \mathcal{Q}_I$ induced by the graph in Figure 8.3 where the destination of the dashed arcs is node 0 and the destination of the solid arcs is unspecified, say a. There are five inequalities associated with this potential conflict,

$$\begin{array}{rcl} y^0_{34} + y^0_{45} + y^a_{32} + y^a_{21} + y^a_{10} & \leq & 4 + y^0_{32} \\ y^0_{34} + y^0_{45} + y^a_{32} + y^a_{21} + y^a_{10} & \leq & 4 + y^0_{21} \\ y^0_{34} + y^0_{45} + y^a_{32} + y^a_{21} + y^a_{10} & \leq & 4 + y^0_{10} \\ y^0_{34} + y^0_{45} + y^a_{32} + y^a_{21} + y^a_{10} & \leq & 4 + y^a_{34} \\ y^0_{34} + y^0_{45} + y^a_{32} + y^a_{21} + y^a_{10} & \leq & 4 + y^a_{45}. \end{array} \tag{8.39}$$

It is easily verified that the first four inequalities above are dominated by the following four inequalities,

$$\begin{array}{rcl} y^a_{32}+y^a_{21}+y^a_{10} & \leq & 2+y^0_{32} \\ y^a_{21}+y^a_{10} & \leq & 1+y^0_{21} \\ y^a_{10} & \leq & 0+y^0_{10} \\ y^0_{34}+y^a_{32}+y^a_{21}+y^a_{10} & \leq & 3+y^a_{34} \end{array} \tag{8.40}$$

Note that all these dominating inequalities stems from cycle families that only use a subset of the arcs in Figure 8.3 which suggests that the definition of $\widetilde{\mathcal{Q}}_{ISP}$ in (8.38) could simply be adapted to the ECMP case. This is however not feasible w.r.t. the fifth inequality since it is not dominated and using the ECMP version of (8.38) would discard the cycle family in Figure 8.3 and therefore disregard the non-dominated fifth inequality. This would be very undesirable.

A modification that that take transitivity into account is required. In this example, the arc destiantion pairs $((3,2),0),((2,1),0),((1,0),0)$ and $((3,4),a)$ are all *included in another smaller cycle family contained in $\mathcal{C}$*. However, the arc destiantion pair $((4,5),a)$ is not included in a smaller cycle family contained in $\mathcal{C}$, it is in a sense a "critical" arc..

In this example with two destinations it is simple to find the smaller cycle families by using a destination arc from nodes 4, 3, 2 and 1, respectively, to node 0. It is more complicated when there are more destinations or more destination arcs.

In general, it is easy to resolve the trivial domination issue when there are no destination arcs, it suffices to only consider irreducible cycle families that are minimal w.r.t. backward arc set inclusion as in (8.38). Let $\widetilde{\mathcal{Q}}_M$ be the collection of inclusionwise minimal cycle families,

$$\widetilde{\mathcal{Q}}_M = \{\mathcal{C} \in \mathcal{Q}_I \mid \text{There exists no } C' \in Q_I \text{ such that } B(\mathcal{C}') \subset B(\mathcal{C})\} \tag{8.41}$$

For irreducible cycle families with destination arcs, it may be necessary to keep the cycle family if it contains a critical arc, so to speak. If so, it suffice to only consider some of the associated valid inequalities, or equivalently, some corresponding transitivity elements for the associated hyperedge. Denote the collection of minimal irreducible cycle families that contain a given critical arc destination pair, $((i,j),l)$, by

$$\widetilde{\mathcal{Q}}_{ijl} = \left\{ \mathcal{C} \in \mathcal{Q}_I \,\middle|\, \begin{array}{l} \text{There exists no } C' \in Q_I \text{ such that} \\ B(\mathcal{C}') \subset B(\mathcal{C}) \text{ and } ((i,j),l) \in F(\mathcal{C}') \end{array} \right\}. \tag{8.42}$$

From the collections above a definition of relevant irreducible cycle families similar to the definition in (8.38) that also takes transitivity for cycle families with destination arcs correctly into account is now made. Let

$$\widetilde{\mathcal{Q}} = \widetilde{\mathcal{Q}}_M \cup \bigcup_{((i,j),l)\in A\times L} \widetilde{\mathcal{Q}}_{ijl}. \tag{8.43}$$

An example is given to clarify these definitions.

Example 8.3

Note that there may be several destination arcs for an irreducible cycle family which makes things more complicated than in Example 8.2. Consider for example the situation on the left in Figure 8.4. Call this cycle family $\mathcal{C}_0$ and the right cycle families $\mathcal{C}_{23}, \mathcal{C}_{21}, \mathcal{C}_{4b}$ and $\mathcal{C}_{4a}$, respectively, depending on the square marked arc on the right in Figure 8.4.

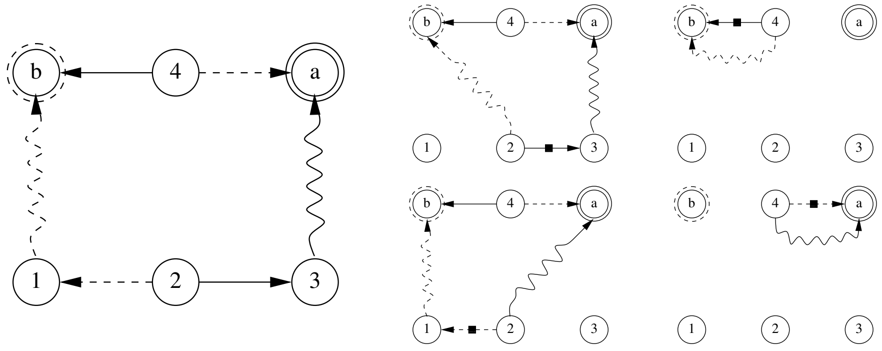


Figure 8.4: *(Left) A cycle family where all associated valid inequalities are dominated. (Right) Four cycle families with at least one associated valid inequality. That is, these four cycle families are in $\widetilde{\mathcal{Q}}_{ijl}$ for some $((i,j),l) \in A \times L$. In all graphs, solid and dashed arcs correspond to design variables associated with destination node a and b, respectively. Further, cycle families on the right can be used to verify the redundance of the dominated valid inequalities associated with the left cycle family. The square marked arcs shows which family that should be used for the verification.*

The cycle family $\mathcal{C}_0 \in \mathcal{Q}_I$ is not in $\widetilde{\mathcal{Q}}$ since it is not minimal (not in $\widetilde{\mathcal{Q}}_M$), nor critical for an arc destination pair (not in $\widetilde{\mathcal{Q}}_{ijl}$ for some $((i,j),l) \in A \times L$).

The inequality associated with arc $(2,1)$ is dominated by the corresponding inequality associated with arc $(2,1)$ for $\mathcal{C}_{21}$. In fact, $\mathcal{C}_{21} \in \widetilde{\mathcal{Q}}_{21b}$. The analogues results for arcs $(2,3)$, $(4,b)$ and $(4,a)$ are easily verified with aid of cycle families $\mathcal{C}_{23}, \mathcal{C}_{4b}$ and $\mathcal{C}_{4a}$.

It is now possible to define a conflict hypergraph in a similar manner as in in the single path case above. The conflict hypergraph, $\mathcal{H}_{PR} = (V, \mathcal{E})$ is formed by introducing a vertex in $\mathcal{H}_{PR}$ for each arc destination pair $((i,j),l) \in A \times L$. Then, a set of hyperedges is created for each relevant irreducible cycle family in $\widetilde{\mathcal{Q}}$. For each $\mathcal{C} \in \widetilde{\mathcal{Q}}_M$ and each $((i,j),l) \in F(\mathcal{C})$, create the hyperedge $E_{ijl} \in \mathcal{E}$ where $E = B(\mathcal{C})$ and $\text{tr}(E) = \{((i,j),l)\}$. For each $\mathcal{C} \in \widetilde{\mathcal{Q}} \setminus \widetilde{\mathcal{Q}}_M$ and each $((i,j),l) \in F(\mathcal{C})$ such that $\mathcal{C} \in \widetilde{\mathcal{Q}}_{ijl}$, create the hyperedge $E_{ijl} \in \mathcal{E}$ where $E = B(\mathcal{C})$ and $\text{tr}(E) = \{((i,j),l)\}$.

The conflict hypergraph, $\mathcal{H}_{PR}$, in general has multiple hyperedges in the ECMP case. The projection onto the simple underlying hypergraph (without multiple hyperedges) does not yield a clutter even though $\widetilde{\mathcal{Q}}$ is now "appropriately" defined. It is more convenient to present this underlying simple hypergraph than $\mathcal{H}_{PR}$. Moreover, the transitive elements is in a sense "clear" from the context anyway.

Example 8.4

Consider a problem where the graph on the left in Figure 8.5 is a subgraph. The restriction to only design variables associated with these arcs and two destinations, destination 1 and an unspecified destination, 0 say, yields the restricted conflict hypergraph, $\mathcal{H}_{PR}$, on the right in Figure 8.5.

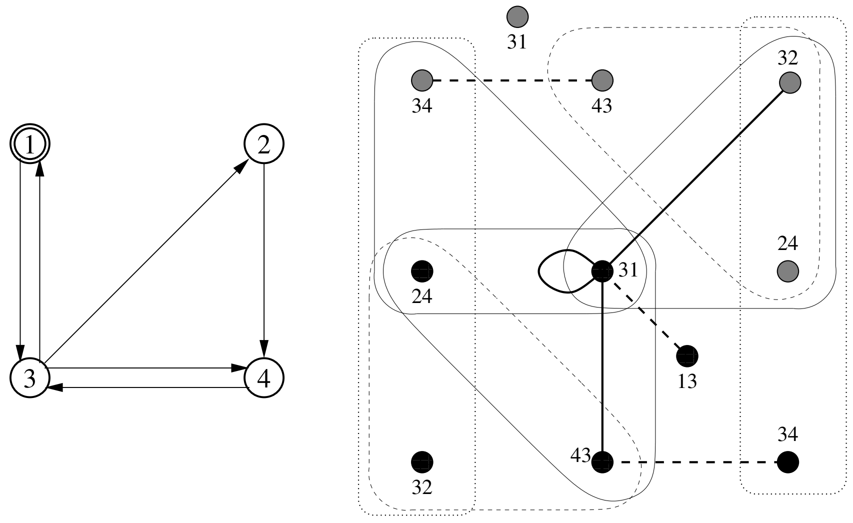


Figure 8.5: A subgraph (left) and the restricted conflict hypergraph (rigth) for the the part of the instance induced by the subgraph and destination 1 and 0. The nodes in the hypergraph are labelled with the arc. A solid black node is associated with destination 0 and a grey node is associated with destination 1. Each hyperedge corresponds to at least one potential SPR conflict.

There are 6 arcs in the original subgraph and two destinations are considered, but one of the arcs, $(1, 3)$, emanates from a destination, node 1. Therefore, there are $11 = 2 \cdot 6 - 1$ hypernodes in $\mathcal{H}_{PR}$.

There are several (potential) conflicts which all yield 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 destination arcs and can not involve node 1. There is one hyperedge for each arc in the conflict. The solid hyperedges represent routing conflicts (valid cycles) that depend on destination arcs and must therefore involve node 1. There is just one hyperedge associated with each of these conflicts.

A more detailed description in terms of the valid inequalities now follows.

Directed cycles correspond to dashed hyperedges. This yields 5 hyperedges without transitive elements and 5 valid inequalities. Namely,

$$y^0_{13} + y^0_{31} \leq 1, \quad y^0_{43} + y^0_{34} \leq 1 \quad \text{and} \quad y^0_{24} + y^0_{32} + y^0_{43} \leq 2, \tag{8.44}$$

for destination 0, and

$$y^1_{43} + y^1_{34} \leq 1, \quad \text{and} \quad y^1_{24} + y^1_{32} + y^1_{43} \leq 2. \tag{8.45}$$

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 $\{((3,4),0),\ ((3,2),1),\ ((2,4),1)\}$ we get,

$$\begin{aligned} y^0_{34} + y^1_{24} + y^1_{32} &\leq 2 + y^0_{24}, \\ y^0_{34} + y^1_{24} + y^1_{32} &\leq 2 + y^0_{32}, \\ y^0_{34} + y^1_{24} + y^1_{32} &\leq 2 + y^1_{34}, \end{aligned} \tag{8.46}$$

and the hyperedge $\{((2,4),0),\ ((3,2),0),\ ((3,4),1)\}$ yields

$$\begin{aligned} y^0_{24} + y^0_{32} + y^1_{34} &\leq 2 + y^0_{34}, \\ y^0_{24} + y^0_{32} + y^1_{34} &\leq 2 + y^1_{24}, \\ y^0_{24} + y^0_{32} + y^1_{34} &\leq 2 + y^1_{32}. \end{aligned} \tag{8.47}$$

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 hyperedge $\{((2,4),0),\ ((3,1),0),\ ((4,3),0)\}$ and its subhyperedges we get,

$$\begin{aligned} y^0_{24} + y^0_{43} + y^0_{31} &\leq 2 + y^1_{24}, \\ y^0_{43} + y^0_{31} &\leq 1 + y^1_{43}, \\ y^0_{31} &\leq 0 + y^1_{31}. \end{aligned} \tag{8.48}$$

Hyperedge $\{((3,2),1),\ ((2,4),1),\ ((3,1),0)\}$ and its subhyperedges yield,

$$\begin{aligned} y^1_{32} + y^1_{24} + y^0_{31} &\leq 2 + y^0_{32}, \\ +y^1_{24} + y^0_{31} &\leq 1 + y^0_{24}, \\ +y^0_{31} &\leq 0 + y^1_{31}. \end{aligned} \tag{8.49}$$

Finally, the last hyperedge $\{((3,4),1),\ ((2,4),0),\ ((3,1),0)\}$ and its subhyperedges yield,

$$\begin{aligned} y^0_{24} + y^1_{34} + y^0_{31} &\leq 2 + y^1_{24}, \\ y^1_{34} + y^0_{31} &\leq 1 + y^0_{34}, \\ y^0_{31} &\leq 0 + y^0_{31}. \end{aligned} \tag{8.50}$$

This hypergraph description may be very fruitful if one wants to derive new classes of valid inequalities based on routing conflicts. Alternatively, the approach can be used to derive simple lifting techniques for valid inequalities that do not induce facets of $\mathcal{Y}$. We will not invesitage this furter here but strongly believe that it is a very promising area for further research.

8.2.5 The Acyclic Ingraph Polytope

A description of the acyclic ingraph polytope, $\mathcal{I}_l$, to node l was given in Proposition 8.5. In this section it is assumed that $G = (N, A)$ is a directed graph where $A = (N \setminus \{l\}) \times N$. Let $\mathcal{D}_l$ be the family of directed cycles in G. The ingraphs are the binary vectors $y^l \in \mathbb{B}^A$ that solves the following system.

$$\begin{array}{rcll} \sum\limits_{(i,j)\in\delta^+(i)} y^l_{ij} & \geq & 1 & i \neq l,\ i \in N \\ \sum\limits_{(i,j)\in C} y^l_{ij} & \leq & |C| - 1 & C \in \mathcal{D}_l. \end{array} \tag{IG$_l$}$$

The polytope described by (IG$_l$) is indeed interesting in its own right. A few related results are given here, but since the primary object under study is $\mathcal{Y}$, we will keep it short and omit some proofs.

Theorem 8.3
Let $c \in \mathbb{Q}^A$ be a cost vector. Then the problem

$$\min c'y, \quad s.t.\ y \in conv\, \mathcal{I}_l \tag{8.51}$$

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 reversely spanning arborescence. There exist several polynomial time algorithms for solving this problem, e.g. [35, 44, 20, 95, 47]. Now assume that $c \notin \mathbb{Q}^A_+$. Then it is easy to polynomially reduce a linear ordering (LO), instance on a graph with n nodes to an instance of (8.51) on a graph with $n + 1$ nodes. For a thorough treatment of the LO problem we refer to [58, 59]. In short, the problem is to find a permutation $p = p_1, \ldots, p_n$ of the nodes that minimizes

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c(p_i, p_j). \tag{8.52}$$

Consider an LO 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 new arcs and solve (8.51) on the augmented graph, with costs $c - M$, where M is a vector where all components are equal to the largest element in c + 1. This implies that all costs are negative and that an optimal solution have the maximum $n(n + 1)/2$ arcs. An optimal linear ordering is obtained by deleting the augmented root node, 0. $\square$

The proof utilizes that the LO polytope is obtained as a face of conv $\mathcal{I}_l$ when the arcs to the augmented root are projected to 0. A similar proof that augments a root node can be used with a reduction from the acyclic subgraph problem, cf. [60], which is also closely related to (8.51). More precisely, the acyclic subrgaph polytope is the polytope obtained when the outdegree constraints in (IG$_l$) are relaxed.

In the case where $c \in \mathbb{Q}_+^A$ in Theorem 8.3, problem (8.51) is equivalent to the shortest spanning arborescence problem. The associated polytopes, the arborescence and rooted arborescence polytope are studied in [44, 57], cf. Chapter 52 in [87]. Especially, a purely graph theoretical characterization of the facets are derived in [57]. Further, the adjacency of branchings, that is, nonspanning arborescences, are studied in [54], their results carry over to arborescences straightforward.

Finally, and most importantly, the dominant of the rooted arborescence polytope can be seen a projection onto the y_{ij}^l variables of the following polyhedron,

$$\begin{array}{rcll} \sum_{(j,i)\in A} x_{ji}^k - \sum_{(i,j)\in A} x_{ji}^k & = & b_i^{kl}, & i \in N,\ k \in N_l \\ x_{ij}^k & \leq & y_{ij}^l & (i,j) \in A,\ k \in N_l \\ 0 \leq y_{ij}^l & \leq & 1 & (i,j) \in A_l, \end{array} \tag{8.53}$$

where b_i^{kl} is given by (8.8), cf. [87] and the references therein. This extended formulation reveals a connection to model (SPRD-C) and further indicates, together with Theorem 8.3, one reason why the ECMP case is much harder than the single path case.

Consider again the polytope conv $\mathcal{I}_l$. It is possible to show the following results about the inequalities in (IG$_l$).

Proposition 8.6
Let $G = (N, A)$ be a directed graph where $A = (N \setminus \{l\}) \times N$. Then the following statements are true.

(a) The dimension of conv $\mathcal{I}_l$ is $m = (n-1)^2$, when $n \geq 3$.

(b) The variable lower bound, $y_{ij} \geq 0$, induces a facet of conv $\mathcal{I}_l$ for any arc $(i,j) \in A$, when $n \geq 4$.

(c) The dimension of the face induced by the variable upper bound, $y_{ij} \leq 1$, is $m-1$ if $j = r$ and $m-2$ otherwise, when $n \geq 3$. That is,the variable upper bound, $y_{ij} \leq 1$, induces a facet of conv $\mathcal{I}_l$ for the arc $(i,r) \in A$, when $n \geq 3$.

(d) The inequality,

$$\sum_{(i,j)\in\delta^+(i)} y_{ij} \geq 1 \tag{8.54}$$

induces a facet of conv $\mathcal{I}_l$ for any node $i \neq l$.

(e) The valid inequality,

$$\sum_{(i,j)\in C} y_{ij} \le |C| - 1 \tag{8.55}$$

induces a facet of conv $\mathcal{I}_l$ for any cycle $C \in \mathcal{D}_l$.

It is possible to derive more classes of facets for conv $\mathcal{I}_l$. Here, we will mention just one. Denote by $\mathcal{F}(R,S)$ the collection of rooted forests that have its roots in the node set R and spans the node set S. Also, the notation $\gamma(S,T) = \{(i,j) \in A : i \in S, j \in T\}$ is used for the set of arcs from node set S to T.

Proposition 8.7
Let $G = (N,A)$ be a directed graph where $A = (N \setminus \{l\}) \times N$. For all $Q, R \subset N$, where $Q \cap R = \emptyset$ and $l \in Q$, the following inequality is valid for any $F \in \mathcal{F}(R, N \setminus Q)$.

$$\sum_{(i,j)\in F} y_{ij} - \sum_{(i,j)\in\gamma(R,Q)} y_{ij} \le |F| - 1 \tag{8.56}$$

Validity of this inequality is obvious; if all arcs the forest are included, then there must be an emanating arc from some root node in R. It is in fact the case that (8.56) very often induces a facet. The only exception is when $Q = \{l\}, R = \{r_0\}$ and $F \in \mathcal{F}(R, N \setminus Q)$ contains a node i with indegree 0 and $(i, r_0) \in F$. A special case (that induce a facet) is the root indegree inequality obtained when $Q = \{l\}$ and $R = N \setminus Q$. That is,

$$\sum_{(i,j)\in\delta^-(l)} y_{ij} \ge 1. \tag{8.57}$$

Rewriting (8.56) into

$$\sum_{(i,j)\in F} (1 - y_{ij}) + \sum_{(i,j)\in\gamma(R,Q)} y_{ij} \ge 1 \tag{8.58}$$

reveals that it is in some cases very simple to separate a valid inequality. If Q or R is fixed, then a min-sum arborescence problem can be solved on an auxilliary graph to find a most violated valid inequality. As mention above, several polynomial time algorithms exist for solving this problem, cf. the proof of Theorem 8.3.

8.3 Separation of Some Combinatorial SPR Cuts

The SPR cuts presented above arise from feasible families of cycles, which is a clear combinatorial structure, at least before the projection. Therefore, all these SPR cuts are in principle combinatorial in nature, as is the induced description of $\mathcal{P}$.

As demonstrated in Chapter 5, the class of feasible cycle families that involve at most two destinations is very rich and comprehensive. The associated cuts that come from this class subsumes many combinatorial cuts presented in the litterature, e.g. the cycle, split and transit cuts in [96] (cf. Sections 8.3.3 and 8.3.2 below). Before our rich class of cuts is analyzed some comments on separation for SPRD problems in general are given.

8.3.1 Separation of SPR Cuts in General

The separation problem considered is to find an inequality of the general form (8.16) that separates a given point $\bar{y} \in [0,1]^{A \times L}$ from conv $\mathcal{P}$. Ideally, a best cut (in some sense) should be found. Two commonly used measures are the most violated cut and the cut whose support is of minimal cardinality. We also distinguish between the cases where $\bar{y}$ is seen as a fractional point and where only the binary part of $\bar{y}$ is considered.

In general, if $\bar{y}$ is fractional it is NP complete to determine if $\bar{y} \in$ conv $\mathcal{P}$ and NP-hard to find a most violated inequality, cf. [9, 14]. In [96], a MIP is given to separate a fractional solution from a set very similar to conv $\mathcal{P}$ via inequalities of the form (8.16). They also provide some combinatorial inequalities along with efficient separation algorithms that we will consider below. Note that it is much easier to solve the separation problem for these combinatorial inequalities than their general MIP since it is possible to utilize the special structure of the cuts.

An alternative to separating a fractional point $\bar{y}$ is to only consider the integral part of $\bar{y}$. That is, to use the index sets $A^0(y)$ and $A^1(y)$ defined in Equation (8.11) and (8.12) in Section 8.2.1. This yields an incomplete, but polynomial, approach to separate some fractional points from conv $\mathcal{P}$ via the set Y^*. It is easy to determine if $\bar{y} \in Y^*$ or not by solving (PR(y)) or one of the equivalent linear programs in the previous chapters. When $\bar{y} \notin Y^*$, the infeasibility certificate yields an improving cycle family that can be turned into a set of valid inequalities for conv $\mathcal{P}$, preferably after reducing the cycle family to an irreducible cycle family if necessary. If $\bar{y} \in Y^*$, the separation procedure fails and is therefore incomplete. Recall that it is NP-complete to decide realisability, therefore this procedure can never be both complete and polynomial (unless P = NP).

When only the integral part of $\bar{y}$ is used, the violation of an inequality of the general form (8.16) is always 1. Therefore, it is more natural to minimize the cardinality of the support of a violated inequality, or equivalently the cycle family, when (partial) integral solutions are separated. It is in general NP-hard to find an infeasible subsystem of minimal cardinality for an arbrirary LP [4, 46]. In our case this is amounts to finding a violated inequality of minimal support. This problem is also NP-hard, cf. [9, 11, 12].

Because of the hardness results related to separation of general valid inequalities of the form (8.16) only the most simple classes of SPR cuts are considered below. Namely, the ones that arise from directed cycles, subpath inconsistence and valid cycles, respectively. Since these structures are well understood and not to complex, it is possible to derive efficient separation algorithms for these specialized cuts.

8.3.2 Directed Cycle Cuts

The directed cycle is a trivial infeasible structure that yields a solution to the Farkas system of (PR(y)), therefore all directed cycles must be prohibited. Also note that the existence of a directed cycle in an ingraph is infeasible and prohibited by the constraints defining $\mathcal{I}_l$ in Proposition 8.5. The class of directed cycle cuts is defined by the following inequalities,

$$\sum_{(i,j)\in C} y^l_{ij} \leq |C| - 1, \quad C \in \mathcal{D}_l, \tag{8.59}$$

where $\mathcal{D}_l$ is the collection of directed cycles not containing node l. The inequalities can also be written as

$$\sum_{(i,j)\in C} (1 - y^l_{ij}) \geq 1, \quad C \in \mathcal{D}_l. \tag{8.60}$$

From this representation it easily seen that a most violated inequality is obtained by solving the following separation problem,

$$\min \sum_{(i,j)\in C} (1 - y^l_{ij}), \quad s.t.\, C \in \mathcal{D}_l. \tag{8.61}$$

That is, a shortest cycle is sought w.r.t. the weights $1 - y$. This problem is sometimes referred to as the weighted girth problem. It is NP-hard in general, but easily solved when all weights are positive, as in (8.61), cf. [39].

In [96, 14], a simple method is suggested to solve the separation problem (8.61). Apply Floyd-Warshalls algorithm with weights $1-y$ for all $l \in L$ to find all pairs shortest paths. Then search for a node pair, (s,t), that minimize the sum of the lengths of the paths $s-t$ and $t-s$. This yields a complexity of $\mathcal{O}\left(Ln^3\right)$ to find a most violated inequality. It is possible to improve this by applying some more efficient shortest path algorithm to each node instead, but in practice it may be easier to use Floyd-Warshalls algorithm which also has a low "overhead" cost.

Remark 8.11. It is quite likely that cycles are short (w.r.t. the number of arcs) in practice. In many applications there is also a so called hop limit that requires that no routing path has more than this fixed number of arcs. Typically, this limit is very small, about 3 arcs. Therefore, adaptions of shortest path algorithms that finds a shortest path with a maximum number of hops can be used if this improves the computational performance.

An alternative to finding a most violated inequality is to search for a violated inequality whose support is of minimal cardinality. That is, to solve

$$\min_{C\in\mathcal{D}_l} |C|, \quad s.t. \sum_{(i,j)\in C} (1 - y^l_{ij}) < 1. \tag{8.62}$$

To solve this problem, cycles that violate the inequality can be found by a modification of Dijkstras algorithm. Use $1 - y$ as arc resources and use a bucket of n node potentials for each node. That is, for $i \in N$ and $k \in \{1,\ldots,n\}$, the value of the node potential, π^k_i is the amount of resources required to reach node i with a path of length at most k. The overhead of using several node potentials for each node increases the complexity by a factor of n and yields an overall complexity of $\mathcal{O}\left(Ln^2SP(n)\right)$ if the shortest path algorithm takes $\mathcal{O}\left(SP(n)\right)$ time.

Remark 8.12. Just as in Remark 8.11 it is possible to assume that cycles are short and limit their length. This yields that fewer node potentials have to be calculated. A hybrid approach that search for a most violated inequality and terminates in advance if a violated inequality whose support is small may be a good idea in practice.

8.3.3 Cuts from Subpath Inconsistence Conflicts

The most simple non-trivial potentially infeasible structure is the one induced by two arc-disjoint shortest subpaths with the same origin and destination. The induced SPR feasibility concept is referred to as subpath optimality, the Bellman property and subpath consistency in the litterature, cf. the discussion on page 95 in Chapter 5. Some examples of instances where this infeasible structure is present were given in Chapter 5, cf. Example 5.1 and 5.3. Valid inequalities based on this class of solutions have earlier been presented, e.g. in [92, 40, 96, 9, 22, 14].

Consider a node pair, (s,t), and two disjoint (s,t)-paths in G, say P' and P''. That is, P' and P'' only share the terminal nodes s and t. This yields that the induced cycle family, $\mathcal{C} = \{\mathcal{C}', \mathcal{C}''\}$, where

$$\mathcal{C}' = F' \cup B' = P'' \cup P' \quad \text{and} \quad \mathcal{C}'' = F'' \cup B'' = P' \cup P'' \tag{8.63}$$

is feasible. Applying Proposition 8.2 gives the following valid inequalities

$$\begin{array}{lcll} y^{l''}_{uv} + \sum\limits_{(i,j)\in P'} \left(1 - y^{l'}_{ij}\right) + \sum\limits_{(i,j)\in P''} \left(1 - y^{l''}_{ij}\right) & \geq & 1 & (u,v) \in P' \\ y^{l'}_{uv} + \sum\limits_{(i,j)\in P'} \left(1 - y^{l'}_{ij}\right) + \sum\limits_{(i,j)\in P''} \left(1 - y^{l''}_{ij}\right) & \geq & 1 & (u,v) \in P'' \end{array} \tag{8.64}$$

for any pair of destinations $l', l'' \in L$.

Remark 8.13. The inequalities in (8.64) are valid for any pair of (s,t)-paths, but the inequalities are stronger if the paths are disjoint, since this implies that $\mathcal{C}$ corresponds to an irreducible conflict. It is easy to see that the inequalities obtained from two (s,t)-paths that share an internal node, say i, are dominated by the inequalities obtained from the corresponding (s,i)-subpaths.

Since the paths P' and P'' were assumed to be completely in A, no destination arcs are included and the additional information that the design variables y^l induce paths to the destination node l has not been used. If paths in $\tilde{A}$ are considered, this vital information is taken into account and cycle families that yield stronger inequalities will also be considered. This yields some cuts that have previously been presented in the litterature, cf. [40, 96, 14]. The motivation for these cuts is based on shortest path properties for one or two paths. Our derivation is more general and demonstrates the significant importance of the destination arcs in $\tilde{D}$.

Let (s,l) be a node pair where $s \in N$ and $l \in L$ and P an (s,l)-path. This yields a feasible cycle family, $\mathcal{C} = \{\mathcal{C}', \mathcal{C}^l\}$, where

$$\mathcal{C}' = F' \cup B' = (s,l) \cup P \quad \text{and} \quad \mathcal{C}^l = F^l \cup B^l = P \cup (s,l) \tag{8.65}$$

and $(s,l) \in \tilde{D}_l$. Proposition 8.2 gives the valid inequalities

$$y^l_{uv} + \sum_{(i,j)\in P} \left(1 - y^{l'}_{ij}\right) \;\geq\; 1 \qquad (u,v) \in P \tag{8.66}$$

for any $l' \in L$. Actually, the only inequality in (8.66) that is not dominated is the one obtained from the first arc in P, say (s,k). That is,

$$y^l_{sk} + \sum_{(i,j)\in P} \left(1 - y^{l'}_{ij}\right) \geq 1. \tag{8.67}$$

It is easy to see that all other inequalities in (8.66) are dominated.

The inequalities in (8.67) were presented in [96] were they are called transit cuts beacause node k is a transit node on the shortest path from s to l. An earlier, but weaker, version of the transit cuts were given in [40].

In [96], a simple and efficient separation algorithm is given for the transit cuts. From (8.67) it is easily seen that it suffices to solve an all pairs shortest path problem with weights $1 - y^{l'}$. Then, a most violated transit cut is found by augmenting a shortest path with the initial arc (s,k). Testing all combinations for s, k, l, l' yields a complexity of $\mathcal{O}\left(|L|n^3 + |L|^2 n^2\right)$ if the all pairs shortest path problem is solved by Floyd-Warshalls algorithm, cf. Remark 8.11.

Finally note the special case transit cut obtained from (8.67) when P is the path consisting of the single arc (s,l) which yields

$$y^l_{sl} \geq y^{l'}_{sl} \tag{8.68}$$

for $l' \in L$. Intuitively these cuts are quite strong and not to many, approximately n^3, therefore one should consider to add them to the SPRD model a priori.

Let us now consider another cut from the litterature that can be derived via the destination arcs. As above, (s,l) is a node pair and P an (s,l)-path. If k' is a node not in P and P' is an (s,k')-path, then another feasible cycle family, $\mathcal{C} = \{\mathcal{C}', \mathcal{C}^l\}$, is obtained by setting

$$\mathcal{C}' = F' \cup B' = (P' \cup (k',l)) \cup P \quad \text{and} \quad \mathcal{C}^l = F^l \cup B^l = P \cup (P' \cup (k',l)), \tag{8.69}$$

where $(k',l) \in \tilde{D}_l$. Proposition 8.2 gives the valid inequalities

$$\begin{aligned} y^{l'}_{uv} + \sum_{(i,j)\in P'} (1 - y^l_{ij}) + \sum_{(i,j)\in P} \left(1 - y^{l'}_{ij}\right) \;&\geq\; 1 \qquad (u,v) \in P' \\ y^l_{uv} + \sum_{(i,j)\in P'} (1 - y^l_{ij}) + \sum_{(i,j)\in P} \left(1 - y^{l'}_{ij}\right) \;&\geq\; 1 \qquad (u,v) \in P \end{aligned} \tag{8.70}$$

for any $l' \in L$.

Observe that (8.66) is obtained from (8.70) by setting $P' = \emptyset$. This shows that the first set of inequalities in (8.70) associated with arcs in P' are all dominated by transit cuts. An inequality in the second set of inequalities associated with an intermediate arc $(u,k) \in P'$ is dominated by the inequality induced by the (s,k)-subpath of P'. Therefore, the only non-dominated inequality in (8.70) is the one associated with the last arc in P'. That is,

$$y^{l'}_{uk'} + \sum_{(i,j)\in P'} \left(1 - y^{l}_{ij}\right) + \sum_{(i,j)\in P} \left(1 - y^{l'}_{ij}\right) \geq 1, \tag{8.71}$$

where (u,k') is the last arc in P'.

In the special case where P' consists of a single arc, (s,k) say, the following inequality is obtained

$$y^{l'}_{sk} + \left(1 - y^{l}_{sk}\right) + \sum_{(i,j)\in P} \left(1 - y^{l'}_{ij}\right) \geq 1. \tag{8.72}$$

Inequalities of this form are called split inequalities in [96]. There, an efficient separation algorithm for the split cuts are give. We use the same idea to find the most violated cut of the form (8.71). From (8.71) it is seen that it suffices to solve two all pairs shortest path problems with weights $1 - y^{l}$ and $1 - y^{l'}$. Then a most violated cut is found by augmenting a shortest (s,u)-path with the terminal arc (u,k'). Testing all combinations for s, u, k', l, l' yields a complexity of $\mathcal{O}\left(|L|^2 n^3\right)$. To find a most violated split cut, set $u = s$, which yields a complexity of $\mathcal{O}\left(|L|n^3 + |L|^2 n^2\right)$ if the all pairs shortest path problem is solved by Floyd-Warshalls algorithm, cf. Remark 8.11. We will suggest a more efficient algorithm below that do not test all combinations. This will reduce the overall complexity by a factor n.

The above derivation shows that inequalities induced from cycle families with destination arcs can be stronger than a corresponding inequality induced by a cycle family that only uses ordinary arcs. It also demonstrated how some previously known cuts stronger than ordinary subpath inconsistency cuts from the partial compatibility case can be interpreted very nicely via the combinatorial cycle family structure with destination arcs. We continue to use this approach to also find cuts and separation algorithms for valid cycles.

8.3.4 Cuts from Valid Cycles

A very important class of infeasible structures is the valid cycles. It subsumes the class of subpath inconsistence structures. More precisely, a subpath inconsistency conflict is a least complex valid cycle in the sense that it is formed by exactly two paths and a valid cycle can contain any number of paths, cf Section 5.4 in Chapter 5.

The discussions in Section 5.4 on page 89 and [25] motivates that the case with two SP-graphs is very important. The computational experiments in [25] showed that almost all infeasible instances with spanning SP-graphs contain a valid cycle. A majority of these conflicts might however be due to subpath inconsistency. Regardless, the family of valid inequalities that prohibits (generalized saturating) valid cycles is very rich and important.

Let C be a simple undirected cycle in $\tilde{G}$. Consider a partitioning of $C = F \cup B$ such that the cycle induced when all F arcs are used forwards and all B arcs backwards becomes a directed cycle contained in $\tilde{A}$. Denote the induced cycle family by, $\mathcal{C} = \{\mathcal{C}', \mathcal{C}''\}$, where

$$\mathcal{C}' = F' \cup B' = F \cup B \quad \text{and} \quad \mathcal{C}'' = F'' \cup B'' = B \cup F. \tag{8.73}$$

For any pair of destinations $l', l'' \in L \subseteq N$, the family $\mathcal{C}$ is feasible since it corresponds to a generalized saturating valid cycle. Applying Proposition 8.2 to $\mathcal{C}$ gives the following valid inequalities

$$\begin{array}{lcll} y_{uv}^{l'} + \sum\limits_{(i,j)\in B\cap A} \left(1 - y_{ij}^{l'}\right) + \sum\limits_{(i,j)\in F\cap A} \left(1 - y_{ij}^{l''}\right) & \geq & 1 & (u,v) \in F \cap A \\ y_{uv}^{l''} + \sum\limits_{(i,j)\in B\cap A} \left(1 - y_{ij}^{l'}\right) + \sum\limits_{(i,j)\in F\cap A} \left(1 - y_{ij}^{l''}\right) & \geq & 1 & (u,v) \in B \cap A. \end{array} \tag{8.74}$$

From these inequalities it is seen that all valid inequalities in Section 8.3.3 can be obtained as valid cycle inequalities of the form (8.74), essentially by setting

$$F = P' \quad \text{and} \quad B = P''. \tag{8.75}$$

To find a most violated valid cycle inequality the decomposition of a valid cycle into alternating path segments is useful, cf. Equations (5.66) and (5.67) and the template in Figure 5.7 in Chapter 5.

Recall that F and B can be decomposed into path segments as

$$F = \bigcup_{p=1}^{K} \overrightarrow{P}_p \quad \text{and} \quad B = \bigcup_{p=1}^{K} \overleftarrow{P}_p, \tag{8.76}$$

where $\overrightarrow{P}_p$ are forward path segments, $\overleftarrow{P}_p$ are backward path segments, and K is the number of segments. This yields the alternating path segment decomposition of C,

$$C = \overrightarrow{P}_1 \overleftarrow{P}_1 \cdots \overrightarrow{P}_K \overleftarrow{P}_K. \tag{8.77}$$

Combining this valid cycle description with the inequalities in (8.74) yields the equivalent valid cycle inequalities,

$$y_{uv}^{l} + \sum_{p=1}^{K} \sum_{(i,j)\in \overleftarrow{P}_p \cap A} \left(1 - y_{ij}^{l'}\right) + \sum_{p=1}^{K} \sum_{(i,j)\in \overrightarrow{P}_p \cap A} \left(1 - y_{ij}^{l''}\right) \geq 1, \tag{8.78}$$

for suitably choosen $l \in \{l', l''\}$ and $(u,v) \in C$.

We now describe an approach to find a most violated valid cycle inequality for fixed l' and l'' that leads to a dynamic programming algorithm. A fundamental observation is that the contributions to the left hand side essentially come as a sum of several shortest path distances and a complicating extra term associated with some arc in some path.

More precisely, if a valid cycle, C, minimize the left hand side in (8.78), then *the contribution from all path segments* $\overrightarrow{P}_p$ *and* $\overleftarrow{P}_p$ *except the one containing the arc* (u,v) *must be the shortest path distances w.r.t. the weights* $1-y^{l'}$ *and* $1-y^{l''}$, respectively. The path segment that contains the arc (u,v) minimize the sum of its length, w.r.t. $1-y^{l'}$ or $1-y^{l''}$, plus the length of arc (u,v), w.r.t. $1-y^{l''}$ or $1-y^{l'}$. We call the latter path segments transitive.

Using this crucial observation, an auxilliary multigraph, $G(l',l'')=(N,\overleftrightarrow{A})$, is defined such that directed "transitive cycles" in $G(l',l'')$ correspond to valid cycles in G. The arc set $\overleftrightarrow{A}$ contains four arcs, $(s,t)'$, $(s,t)''$, $(\overline{s,t})'$ and $(\overline{s,t})''$ for each node pair $(s,t)\in N\times N$. The arcs, $(s,t)'$ and $(t,s)''$ represent a shortest path from s to t in the original graph, G, while the arcs, $(\overline{s,t})'$ and $(\overline{t,s})''$ represent a shortest path from s to t that also pays the fee for an appropriate (u,v) arc in the original graph, so to speak. That is, a shortest transitive path segment from s to t. Note that the arcs associated with l'' have been reversed.

From these interpretations it follows that the weights on the arcs should be set as follows. The weight of arcs, $(s,t)'$ and $(s,t)''$, are denoted by p'_{st} and p''_{st}, respectively. Here p'_{st} is the shortest path distance from s to t in G using weights $1-y^{l'}$ and p''_{st} is the shortest path distance from t to s using $1-y^{l''}$. Formally,

$$p'_{st}=\min_{P\in\mathcal{P}_{st}}\sum_{(i,j)\in P}\left(1-y^{l'}\right) \quad\text{and}\quad p''_{st}=\min_{P\in\mathcal{P}_{ts}}\sum_{(i,j)\in P}\left(1-y^{l''}\right), \tag{8.79}$$

where $\mathcal{P}_{st}$ is the set of all paths from s to t in G. The weights of the transitive arcs, $(\overline{s,t})'$ and $(\overline{s,t})''$, are respectively, $\bar{p}'_{st}$ and $\bar{p}''_{st}$, where $\bar{p}'_{st}$ and $\bar{p}''_{st}$ are defined as

$$\bar{p}'_{st}=\min_{P\in\mathcal{P}_{st}}\left\{\sum_{(i,j)\in P}\left(1-y^{l'}\right)+\min_{(i,j)\in P}y^{l''}\right\}, \tag{8.80}$$

and

$$\bar{p}''_{st}=\min_{P\in\mathcal{P}_{ts}}\left\{\sum_{(i,j)\in P}\left(1-y^{l''}\right)+\min_{(i,j)\in P}y^{l'}\right\}. \tag{8.81}$$

That is, $\bar{p}'_{st}$ and $\bar{p}'_{st}$ are the distances of paths that include, and compensate for, an arc in the path that is associated with the other destination.

From the alternating path segment decomposition of C, in (8.77) it is clear that a directed cycle in $G(l',l'')$ correspond to valid cycle in G. To find a most violated valid cycle inequality of type (8.78) it is sufficient to *find a shortest directed cycle in $G(l',l'')$ that uses at least on transitive arc*. This crucial observation is stated as a theorem.

Theorem 8.4
For fixed l' and l'', let $C = F \cup B$, $(u,v) \in C$ and $l \in \{l', l''\}$ be the valid cycle and associated arc and destination that minimize

$$y^l_{uv} + \sum_{(i,j)\in B\cap A} \left(1 - y^{l'}_{ij}\right) + \sum_{(i,j)\in F\cap A} \left(1 - y^{l''}_{ij}\right) \tag{8.82}$$

over the family of all valid cycle inequalities. Then, C corresponds to a directed cycle in $G(l',l'')$ that uses exactly one transitive arc of minimal length.

Let us now describe a dynamic programming method to solve this problem. First, the quantities $p'_{st}, p''_{st}, \bar{p}'_{st}$ and $\bar{p}''_{st}$ are determined. Then, the shortest "transitive cycle" in $G(l',l'')$ with these arc weights is found.

Clearly, p'_{st} can be found by solving an all pairs shortest path problem in G using weights $1 - y^{l'}$. Note that the cost of an arc (i,l') should be set to 0 since there is always a destination arc to l'. Define $p'_{st,k}$ to be the length of the shortest path from s to t with weights $1 - y^{l'}$ using only nodes $1,\ldots,k$ as intermediate nodes. This yields the following standard recurrence relation solved by the Floyd-Warshall algorithm.

- If $k = 0$, let
$$p'_{st,k} = \begin{cases} 1 - y^{l'}_{st} & \text{if } t \neq l', \\ 0 & \text{if } t = l'. \end{cases} \tag{8.83}$$

- If $k > 0$, let
$$p'_{st,k} = \min\left\{p'_{st,k-1},\ p'_{sk,k-1} + p'_{kt,k-1}\right\}. \tag{8.84}$$

Remark 8.14. Note that, if the direct arc (s,t) is not in G, then $p_{st,0} = 1$ and will not be considered in an optimal solution with value less than 1. Hence, there is no need to set $p_{st,0} = \infty$ when a violated inequality is sought.

The p''_{ts} weights are found equivalently. Note the reversal of s and t here.

To find the weights for the transitive arcs, $(\overline{s,t})'$, the minimization problem in (8.80) is solved by a modification of the Floyd-Warshall algorithm. Let $\bar{p}'_{st,k}$ be the length of the shortest transitive path from s to t using only nodes $1,\ldots,k$ as intermediate nodes. This yields the following recurrence relation.

- If $k = 0$, let $\bar{p}'_{st,k} = 1 - y^{l'}_{st} + y^{l''}_{st}$.

- If $k > 0$, let
$$\bar{p}'_{st,k} = \min\left\{\bar{p}'_{st,k-1},\ \bar{p}'_{sk,k-1} + p'_{kt,k-1},\ p'_{sk,k-1} + \bar{p}'_{kt,k-1}\right\}. \tag{8.85}$$

The motivation for this 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 $\bar{p}''_{ts}$ weights are again found equivalently.

Solving the four recurrence relations for $p'_{st}, p''_{st}, \bar{p}'_{ts}$ and $\bar{p}''_{ts}$ can be done in $\mathcal{O}\left(n^3\right)$ time by dynamic programming.

To find a shortest transitive cycle in $G(l', l'')$ an all pairs shortest transitive path problem is solved in $G(l', l'')$ with weights $p'_{st}, p''_{st}, \bar{p}'_{ts}$ and $\bar{p}''_{ts}$ in a similar manner as above. Let $d_{st,k}$ be the length of the shortest path from s to t in $G(l', l'')$ and $\bar{d}_{st,k}$ the length of the shortest transitive path from s to t in $G(l', l'')$ using only nodes $1, \ldots, k$ as intermediate nodes. This yields the following recurrence relations.

- If $k = 0$, let

$$d_{st,k} = \min\left\{p'_{st,k}, p''_{st,k}\right\} \quad \text{and} \quad \bar{d}_{st,k} = \min\left\{\bar{p}'_{st,k}, \bar{p}''_{st,k}\right\}. \tag{8.86}$$

- If $k > 0$, let

$$d'_{st,k} = \min\left\{d_{st,k-1},\ d_{sk,k-1} + d_{kt,k-1}\right\} \tag{8.87}$$

and

$$\bar{d}_{st,k} = \min\left\{\bar{d}_{st,k-1},\ \bar{d}_{sk,k-1} + d_{kt,k-1},\ d_{sk,k-1} + \bar{d}_{kt,k-1}\right\}. \tag{8.88}$$

These recurrence relations can again be solved in $\mathcal{O}\left(n^3\right)$ time by dynamic programming. Then, a most violated valid cycle inequality corresponds to a shortest transitive cycle which is associated with a minimal diagonal element $\bar{d}_{ss,n} < 1$. To find the actual cycle and arc, the solution can be found in a similar manner as for ordinary all pair shortest path problems.

If it suffices to find a most violated subpath inconsistency inequality a more efficient algorithm can be derived. Since these special valid cycles have only two path segments it suffices to find an s that minimizes $\bar{d}^0_{ss}$, where

$$\bar{d}^0_{ss} = \min_t\left\{\bar{d}_{st,0} + d_{ts,0},\ d_{st,0} + \bar{d}_{ts,0}\right\}. \tag{8.89}$$

That is, a cycle that only uses two path segments, (s, t) and (t, s). This reduces the time complexity to $\mathcal{O}\left(n^2\right)$, if $p'_{st}, p''_{st}, \bar{p}'_{ts}$ and $\bar{p}''_{ts}$ are given.

Theorem 8.5
A most violated valid cycle inequality can be found in $\mathcal{O}\left(|L|^2 n^3\right)$ time and a most violated subpath inconsistency inequality can be found in $\mathcal{O}\left(|L|^2 n^2 + |L| n^3\right)$ time.

Proof: The quantities $p'_{st}, p''_{st}, \bar{p}'_{ts}$ and $\bar{p}''_{ts}$ may be calculated in $\mathcal{O}\left(|L| n^3\right)$ as described above. Then, testing all combinations of destinations yields the result. □

Remark 8.15. Note that the subpath inconsistency separation algorithm proposed in this section is more efficient than the one in the previous section.

Remark 8.16. In practice one should probably check for violated subpath inconsistency inequalities once the quantities p_{st} and $\bar{p}_{ts}$ have been calculated for appropriate destinations. Then, the separation problem can be solved for a valid cycle if there is not a "sufficiently violated" subpath inconsistency inequality.

Remark 8.17. From the derivation above it is clear that the separation of a most violated valid cycle inequality becomes much easier to describe in the single path case where there are no complicating transitive arcs to take into account. In this case the valid cycles corresponds to ordinary directed cycles in $G(l', l'')$ which is now also a simple graph since the arc weights can be set to

$$w_{ij} = \min\left\{p'_{st,n},\ p''_{st,n}\right\}. \tag{8.90}$$

The separation problem is an ordinary shortest cycle problem with positive weights. This does however not affect the computational complexity for finding the most violated inequality. This is due to the fact that there is a fixed number of transitive arcs to take into account, in this case just one.

Finally we mention that it is (rather) straightforward to adapt the above algorithms to find a violated valid cycle inequality of minimal cardinality by using the dynamic programming principle with multiple node potentials described for the directed cycle cuts above. This would worsen the time and space complexity by a factor n.

A breif conclusion and some suggestions for interesting areas of further research conclude this thesis.

9 Conclusion

Two aspects of shortest path routing (SPR) have been considered in this thesis: the inverse problem, which was referred to as IPSPR and the design problem (SPRD). Several aspects of the inverse problem were covered in depth in Chapter 3-7. The results from this study were then used to breifly analyze the design problem in Chapter 8.

This chapter contains a summary and some directions for further research.

9.1 Summary

Chapter 1-3 contain a technical background and an introduction to IPSPR problems for some common settings.

The first chapter with significant new results is Chapter 4 where our first major contribution is also presented. That is, adressing the issues of realizability. A family of partial ingraphs is realizable if it is realizable in an SPR protocol. It is shown that the ordinary IPSPR formulation for partial ingraphs is incomplete and a complete bilinear model is proposed. The reason that the ordinary IPSPR formulation does not suffice to determine realizability is that it is not guaranteed that the node potentials are tight. Therefore, a solution can not be used to verify realizability as is shown by several examples. Our main theoretical result in the chapter is that the problem to determine if a family of partial ingraphs is realizable in an SPR protocol is NP-complete.

Our complete realizability model forces all SP-graphs to contain an arborescence, which implies that node potentials are tight. From this model, a set of valid inequalities are derived, which yields a significant improvement of the ordinary IPSPR model. This new model is referred to as the partial realizability (PR) model. It yields stronger necessary conditions for realizability, but it is still not complete.

SPR conflicts are considered in Chapter 5 and 6. By analyzing the mulitcommodity structured Farkas system of the PR model a combinatorial characterization of a very large class of SPR conflicts is derived. A characterization of all (potentially) infeasible routing patterns is obtained. The conflicts are classified, in a sense w.r.t. their complexity, into general, binary, unitary, simplicial and valid cycle conflicts. It is shown that all these conflict classes are necessary by a set of examples.

The most comprehensible classes are the valid cycle and simplicial conflicts. These conflicts are thoughly analyzed and simplicial extreme and irreducible solutions are characterized with aid of a concept similar to graph duality. An efficient algorithm is also developed for finding valid cycles.

In Chapter 7, an alternative PR model is proposed. The circulation structure of the Farkas system of the PR model is exploited to derive a novel ISPR model based on fundamental cycle bases.

The cycle basis model is more compact since it does not include the flow conservation constraints but only the capacity constraints. The circulation constraints are in a sense encoded in the fundamental cycle variables. This comes at the price of less structured and more dense capacity constraints. Our preliminary computational experiments suggest that the cycle basis model can be solved more efficiently than the ordinary model.

Further, the cycle basis structure in a sense leaves only a part of the PR model which makes it easier to analyze. This yields us to be able to derive some important theoretical insights about the model which can then easily be translated to the original problem. For instance, it is under rather general assumptions possible to remove a set of constraints from the cycle basis or PR model which sometimes significantly reduces the time required to solve an instance.

Finally, in Chapter 8 a mixed integer linear programming formulation for the core of the SPRD problems is presented. Our model does not contain weight variables since this yields extremely weak LP-relaxations. Instead, the characterization from Chapter 5 is used to derive valid inequalities that prohibit parts of routing patterns that are not realizable.

Using the PR model instead of the ordinary IPSPR model it is possible to detect more SPR conflicts earlier. The associated valid inequalities are also stronger than the corresponding valid inequalities derived from the ordinary model since they have in a sense automatically been lifted and projected.

The structure of these valid inequalities is analyzed and the set of partial routing patterns that do not contain a conflict is described as packings in an associated conflict hypergraph. Since the PR model is considered, transitive packings have to be used for the ECMP case. The transitivity issue implies that the conflict hypergraph is not a clutter and leads to some problems concerning the dominance and redundancy for the associated constraints. A few requirements for non-domination are mentioned. The hypergraph structure is also useful since several results on transitive packing polytopes can be used to derive more SPR valid inequalities.

The most comprehensible class of conflicts is the class of valid cycles. That is, conflicts that involve two partial ingraphs. Some empirical evidence suggest that valid cycles explain most conflicts.

We consider the PR based valid inequalities with two partial ingraphs in detail. Our valid inequalities subsume and explain all combinatorial cuts from the litterature based on conflicts with two partial ingraphs, further illustrating the superiority of the PR model over the ordinary model. The analysis allows us to develop algorithms that efficiently separate a fractional solution from a most violated valid cycle inequality.

9.2 Further Work

In the end, solving traffic enginering problems in IP networks is the important application of the theory developed in this thesis. The primary reason for studying the inverse problem is just to be able to draw conclusions for the design problem. Therefore, the emphasis on our future work will be on exact solution methods for the design problem based on mixed integer linear programming.

The major issue for us is of course to implement and analyze the framwork described in Chapter 8 and provide some competitive computational results.

An outline of some general directions for furter research follows.

- A computational comparison of methods for solving IPSPR problems is currently being done. Solving the IPSPR subproblem for partial integral solutions is important to be able to prune the enumeration tree or derive violated valid inequalities.
- The separation problem for fraction solutions and cuts based on simplicial conflicts may be important when the valid cycle separation fails. Developing a good model for this problem may be important.
- An interesting theoretical question to be answered is which of the valid inequalities in Chapter 8 that define facets.
- Deriving more valid inequalities based on hypergraph structures and analyzing their strength may also be good.
- Analyzing the polytope associated with the relaxation where only conflicts that involve two SP-graphs are prohibited is also interesting since these conflicts seem to be important.
- Further analyzing the acyclic ingraph polytope may also yield strong valid inequalities.

Bibliography

[1] E. Aarts and J. Korst. *Simulated annealing and Boltzmann machines.* John Wiley & Sons New York, 1989.

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

[3] A. Altın, B. Fortz, M. Thorup, and H. Ümit. Intra-domain traffic engineering with shortest path routing protocols. *4OR: A Quarterly Journal of Operations Research*, 7(4):301–335, 2009.

[4] E. Amaldi, M.E. Pfetsch, and L.E. Trotter, Jr. On the maximum feasible subsystem problem, IISs and IIS-hypergraphs. *Mathematical Programming*, 95(3):533–554, 2003.

[5] M.A. Armstrong. *Basic Topology. Undergraduate Texts in Mathematics.* Springer-Verlag, 1983.

[6] Walid Ben-Ameur and Eric Gourdin. Internet routing and related topology issues. *SIAM Journal on Discrete Mathematics*, 17:18–49, 2003.

[7] N. Biggs. *Algebraic graph theory.* Cambridge Univ Pr, 1993.

[8] N. Biggs, E.K. Lloyd, and R.J. Wilson. *Graph theory, 1736-1936.* Oxford University Press, USA, 1986.

[9] A. Bley. *Routing and Capacity Optimization for IP Networks.* PhD thesis, TU Berlin, 2007.

[10] A. Bley. Approximability of unsplittable shortest path routing problems. *Networks*, 54(1):23–46, 2009.

[11] A. Bley. Logarithmic inapproximability results for the minimum shortest path routing conflict problem. Technical Report 025-2009, Institute of Mathematics, TU, 2009. Submitted to Networks.

[12] A. Bley. On the hardness of finding small shortest path routing conflicts. In *Proceedings of the 4th International Network Optimization Conference (INOC 2009), Pisa, Italy*, 2009.

[13] A. Bley. An integer programming algorithm for routing optimization in IP networks. *Algorithmica*, To appear.

[14] A. Bley, B. Fortz, E. Gourdin, K. Holmberg, O. Klopfenstein, M. Pióro, A. Tomaszewski, and H. Ümit. Optimization of OSPF routing in IP networks. In A. M. C. A. Koster and X. Muñoz, editors, *Graphs and Algorithms in Communication Networks: Studies in Broadband, Optical, Wireless and Ad Hoc Networks*, chapter 8, pages 199–240. Springer, 2009.

[15] A. Bley, M. Grötschel, and R. Wessäly. Design of broadband virtual private private networks: Model and heuristics for the B-WiN. Technical report, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, 1998.

[16] A. Bley and T. Koch. Integer programming approaches to access and backbone IP-network planning. In *Modeling, Simulation and Optimization of Complex Processes: Proceedings of the 3rd International Conference on High Performance Scientific Computing, Hanoi, Vietnam*, pages 87–110. Springer, 2006.

[17] Andreas Bley. Inapproximability results for the inverse shortest paths problem with integer lengths and unique shortest paths. *Netw.*, 50(1):29–36, 2007.

[18] Andreas Bley, Martin Grötschel, and Roland Wessäly. Design of broadband virtual private private networks: Model and heuristics for the B-WiN. In Nathaniel Dean, D. Frank Hsu, and R. Rav, editors, *Robust Communication Networks : Interconnection and Survivability*, volume 53 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 1–16. AMS, 2000.

[19] Andreas Bley and Thorsten Koch. Integer programming approaches to access and backbone IP-network planning. Technical Report ZR-02-41, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, 2002.

[20] F. Bock. An algorithm to construct a minimum directed spanning tree in a directed network. *Developments in operations research*, 1:29–44, 1971.

[21] B. Bollobás. *Modern graph theory*. Springer Verlag, 1998.

[22] P. Broström and K. Holmberg. Design of OSPF networks using subpath consistent routing patterns. *Telecommunication Systems*, 41(4):293–309, 2009.

[23] Peter Broström and Kaj Holmberg. Stronger necessary conditions for the existence of a compatible OSPF metric. Research Report LiTH-MAT-R-2004-08, Department of Mathematics, Linköping Institute of Technology, Sweden, 2004.

[24] Peter Broström and Kaj Holmberg. On the extremal structure of an OSPF related cone. *Vietnam Journal of Mathematics*, 35:507–522, 2007.

[25] Peter Broström and Kaj Holmberg. Valid cycles: A source of infeasibility in OSPF routing. *Networks*, 52:206–215, 2008.

[26] Peter Broström and Kaj Holmberg. Compatible weights and valid cycles in non-spanning OSPF routing patterns. *Algorithmic Operations Research*, 4:19–35, 2009.

[27] Peter Broström. *Optimization Models and Methods for Telecommunication Networks using OSPF.* PhD thesis, Linköping UniversityLinköping University, Optimization , The Institute of Technology, 2006.

[28] E. Burger. Über homogene lineare ungleichungssysteme. *Zeitschrift für Angewandte Mathematik und Mechanik*, 36(3-4):135–139, 1956.

[29] L.S. Buriol, M.G.C. Resende, C.C. Ribeiro, and M. Thorup. A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing. *Networks*, 46(1):36–56, 2005.

[30] Luciana S. Buriol, Mauricio G. C. Resende, and Mikkel Thorup. Speeding Up Dynamic Shortest-Path Algorithms. *INFORMS JOURNAL ON COMPUTING*, 20(2):191–204, 2008.

[31] E.K. Burke and G. Kendall. *Search methodologies: introductory tutorials in optimization and decision support techniques.* Springer Verlag, 2005.

[32] D. Burton and P.L. Toint. On an instance of the inverse shortest paths problem. *Mathematical Programming*, 53:45–61, 1992.

[33] R.W. Callon. Use of OSI IS-IS for routing in TCP/IP and dual environments. *Network*, 1990.

[34] M. Chan. A survey of the cycle double cover conjecture, 2009.

[35] Y.J. Chu and T.H. Liu. On the shortest arborescence of a directed graph. *Science Sinica*, 14(1396-1400):270, 1965.

[36] Cisco. Configuring OSPF, 1997.

[37] M. Conforti and M. Laurent. On the facial structure of independence system polyhedra. *Mathematics of Operations Research*, 13(4):543–555, 1988.

[38] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, New York, NY, USA, 1971. ACM.

[39] R. Coullard William and R. Collette. On cycle cones and polyhedra. *Linear Algebra and its Applications*, 114:613–640, 1989.

[40] L. De Giovanni, B. Fortz, and M. Labbé. A lower bound for the internet protocol network design problem. In L. Gouveia and C. Mourao, editors, *INOC 2005*, pages 401–408. University of Lisbon, Lisbon, Portugal, 2005.

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

[42] M. Dzida, M. Zagozdzon, and M. Pióro. Optimization of Resilient IP Networks with Shortest Path Routing. In *International Workshop on the Design of Reliable Communication Networks (DRCN), La Rochelle, France*, 2007.

[43] T. Easton, K. Hooker, and E.K. Lee. Facets of the independent set polytope. *Mathematical Programming*, 98(1):177–199, 2003.

[44] J. Edmonds. Optimum branchings. *J. Res. Natl. Bur. Stand., Sect. B*, 71:233–240, 1967.

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

[46] M. Fischetti, D. Salvagnin, and A. Zanette. Minimal infeasible subsystems and Benders cuts. *Mathematical Programming to appear*, 2009.

[47] M. Fischetti and P. Toth. An efficient algorithm for the min-sum arborescence problem on complete digraphs. *INFORMS Journal on Computing*, 5(4):426, 1993.

[48] Bernard Fortz. Internet traffic engineering by optimizing ospf weights. In *in Proc. IEEE INFOCOM*, pages 519–528, 2000.

[49] Bernard Fortz and Mikkel Thorup. Increasing internet capacity using local search, 2000.

[50] Bernard Fortz and Mikkel Thorup. Increasing internet capacity using local search. *Computational Optimization and Applications*, 29(1):13–48, 2004.

[51] P. Fouilhoux, M. Labbé, A.R. Mahjoub, and H. Yaman. Generating facets for the independence system polytope. *SIAM Journal on Discrete Mathematics*, 23(3):1484–1506, 2009.

[52] András. Frank. Connectivity and network flows. In A.L. Graham, M. Grötschel, and L. Lovász, editors, *Handbook of Combinatorics*, volume 1, pages 111–177. North-Holland, 1995.

[53] H.N. Gabow. Path-Based Depth-first Search for Strong and Biconnected Components. *Information Processing Letters*, 2000.

[54] R. Giles and D. Hausmann. Characterizations of adjacency on the branching polyhedron. *Discrete Mathematics*, 26(3):219–226, 1979.

[55] M. Goemans and L. Hall. The strongest facets of the acyclic subgraph polytope are unknown. *Integer Programming and Combinatorial Optimization*, pages 415–429, 1996.

[56] E. Gourdin and O. Klopfenstein. Comparison of different QoS-oriented objectives for multicommodity flow routing optimization. In *Proceedings of the International Conference on Telecommunications (ICT 2006)*, 2006.

[57] M. Grötschel. Strong blocks and the optimum branching problem. *Numerische Methoden bei graphentheoretischen und kombinatorischen Problemen: Band 2: Vortragsauszüge der Tagung über Numerische Methoden bei Graphentheoretischen und Kombinatorischen Problemen vom 7. bis 12. Mai 1978 im Mathematischen Forschungsinstitut Oberwolfach (Schwarzwald)*, page 112, 1979.

[58] M. Grötschel, M. Jünger, and G. Reinelt. A cutting plane algorithm for the linear ordering problem. *Operations research*, 32(6):1195–1220, 1984.

[59] M. Grötschel, M. Jünger, and G. Reinelt. Facets of the linear ordering polytope. *Mathematical Programming*, 33(1):43–60, 1985.

[60] M. Grötschel, M. Jünger, and G. Reinelt. On the acyclic subgraph polytope. *Mathematical Programming*, 33(1):28–42, 1985.

[61] G. Haggard. Edmonds characterization of disc embeddings. In *Proceedings of the Eighth Southeastern Conference on Combinatorics, Graph Theory, and Computing, Louisiana State University, Baton Rouge, February 28-March 3, 1977*, page 291. Utilitas Mathematica, 1977.

[62] M. Hartmann, D. Hock, C. Schwartz, and M. Menth. Objective Functions for Optimization for Resilient and Non-Resilient IP Routing. In *7th International Workshop on Design of Reliable Communication Networks (DRCN), Washington, DC, USA*. Citeseer, 2009.

[63] Kaj Holmberg and Di Yuan. Optimization of Internet Protocol network design and routing. Research Report LiTH-MAT-R-2001-07, Department of Mathematics, Linköping Institute of Technology, Sweden, 2001. Revised April 2002.

[64] Kaj Holmberg and Di Yuan. Optimization of Internet Protocol network design and routing. *Networks*, 43:39–53, 2004.

[65] F. Jaeger. A survey of the cycle double cover conjecture. *North-Holland Mathematics Studies*, 115:1–12, 1985.

[66] F. Jaeger. Nowhere-zero flow problems. *Selected topics in graph theory*, 3:71–95, 1988.

[67] Telikepalli Kavitha, Christian Liebchen, Kurt Mehlhorn, Dimitrios Michail, Romeo Rizzi, Torsten Ueckerdt, and Katharina A. Zweig. Cycle bases in graphs characterization, algorithms, complexity, and applications. *Computer Science Review*, 3(4):199–243, 2009.

[68] Gustav Robert Kirchhoff. Über die auflösung der gleichungen, auf welche man bei der untersuchung der linearen verteilung galvanischer ströme geführt wird. *Ann. Phys. Chem.*, 72(12):497–508, 1847.

[69] B. Korte and D. Hausmann. An analysis of the greedy heuristic for independence systems. *Algorithmic Aspects of Combinatorics*, 2:65–74, 1978.

[70] J. Leung and J. Lee. More facets from fences for linear ordering and acyclic subgraph polytopes. *Discrete Applied Mathematics*, 50(2):185–200, 1994.

[71] Christian Liebchen and Romeo Rizzi. Classes of cycle bases. *Discrete Applied Mathematics*, 155(3):337–355, 2007.

[72] S. Maheshwary. *Facets of conflict hypergraphs*. Phd thesis, Georgia Institute of Technology, 2008.

[73] Björn Morén. Non-trivial shortest path routing conflicts - classification and search methods. Master's thesis, Linköping University, 2010. LiTH-MAT-EX–2010/19–SE.

[74] J. Moy. OSPF version 2. IETF Internet RFC 2740, 1998.

[75] Rudolf Müller and Andreas S. Schulz. Transitive packing: A unifying concept in combinatorial optimization. *SIAM J. on Optimization*, 13(2):335–367, 2002.

[76] Rudolf Müller and Andreas S. Schulz. Transitive packing. In *Integer Programming and Combinatorial Optimizationb, Lecture Notes in Computer Science 1084*, pages 430–444. Springer, 1996.

[77] G.L. Nemhauser and L.E. Trotter. Properties of vertex packing and independence system polyhedra. *Mathematical Programming*, 6(1):48–61, 1974.

[78] M. Padberg and T.Y. Sung. An analytical comparison of different formulations of the travelling salesman problem. *Mathematical Programming*, 52(1):315–357, 1991.

[79] A. Parmar, S. Ahmed, and J. Sokol. An integer programming approach to the OSPF weight setting problem. *Optimization Online*, 2006.

[80] M. Pioro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers, 2004.

[81] M. Pioro, A. Szentesi, J. Harmantos, A. Jüttner, P. Gajowniczek, and S. Kozdrowski. On open shortest path first related network optimization problems. *Performance Evaluation*, 48:201–223, 2002.

[82] M. Pioro, A. Szentesi, J. Harmatos, A. Jüttner, P. Gajowniczek, and S. Kozdrowski. On open shortest path first related network optimisation problems. *Performance Evaluation*, 48(1-4):201–223, 2002.

[83] M. Pióro and A. Tomaszewski. Feasibility issues in shortest-path routing with traffic flow split. In *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*, 2007.

[84] Mikael Prytz. *On Optimization in Design of Telecommunications Networks with Multicast and Unicast Traffic*. Phd dissertation, Royal Institute of Technology, Sweden, 2002. TRITA-MAT-02-OS-05.

[85] R.T. Rockafellar. *Convex analysis.* Princeton University Press, 1996.

[86] A. Schrijver. *Theory of linear and integer programming.* John Wiley & Sons Inc, 1998.

[87] A. Schrijver. *Combinatorial optimization: polyhedra and efficiency.* Springer, 2003.

[88] A. Schulz. Polytopes and scheduling. *PhDthesis, FachbereichMathematik, Technische Universitt Berlin, Berlin, Germany*, 1996.

[89] PD Seymour. *Sums of circuits.* Academic Press, 1979.

[90] M. Sharir. A strong-connectivity algorithm and its application in data flow analysis. *Computers and Mathematics with Applications*, 7:67–72, 1981.

[91] K. Simon. An improved algorithm for transitive closure on acyclic digraphs. *Theoretical Computer Science*, 58(1-3):325–346, 1988.

[92] D. Staehle, S. Köhler, and U. Kohlhaas. Towards an optimization of the routing parameters for IP networks. *Univ. Würzburg, Germany, Tech. Rep*, 258, 2000.

[93] G. Szekeres. Polyhedral decompositions of cubic graphs. *Bulletin of the Australian Mathematical Society*, 8(03):367–387, 1973.

[94] R. E. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal of Computing*, 1:146–160, 1972.

[95] R. E. Tarjan. Finding optimum branchings. *Networks*, 7(1):25–35, 1977.

[96] A. Tomaszewski, M. Pióro, M. Dzida, M. Mycek, and M. Zagozdzon. Valid inequalities for a shortest-path routing optimization problem. In *3rd International Network Optimization Conference. Université Libre de Bruxels, Brussels*, 2007.

[97] Ümit, H. and Fortz, B. Fast heuristic techniques for intra-domain routing metric optimization. In *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*, volume 6, 2007.

[98] Di Yuan. *Optimization Models and Methods for Communication Network Design and Routing.* Phd dissertation, Linköping University, Sweden, 2001. Linköping Studies in Science and Technology. Dissertation no. 682.

[99] C.Q. Zhang. *Integer flows and cycle covers of graphs.* CRC, 1997.