A Mixed Integer Approach for the Transient Case of Gas Network Optimization

Vom Fachbereich Mathematik

der Technischen Universität Darmstadt

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

genehmigte Dissertation

von

Dipl.–Math. Susanne Moritz

aus Köln

Referent:	Prof. Dr. A. Martin
Korreferent:	Prof. Dr. R. Schultz
Tag der Einreichung:	24. Oktober 2006
Tag der mündlichen Prüfung:	6. Dezember 2006

Darmstadt 2007
D17

Also lautet ein Beschluß,
Daß der Mensch was lernen muß.
Nicht allein das Abc
Bringt den Menschen in die Höh’;
Nicht allein im Schreiben, Lesen
Übt sich ein vernünftig Wesen;
Nicht allein in Rechnungssachen
Soll der Mensch sich Mühe machen,
Sondern auch der Weisheit Lehren
Muß man mit Vergnügen hören.

Wilhelm Busch, Max und Moritz

Zusammenfassung

Erdgas ist der drittwichtigste Energieträger der Welt. Der Verbrauch von Erdgas steigt derzeit am stärksten im Vergleich zu anderen nicht erneuerbaren Energieträgern. Daher stellt Optimierung von Gastransport in Netzwerken eine wichtige logistische Herausforderung dar.

In dieser Dissertation betrachten wir das Problem der zeitabhängigen Optimierung von Gasnetzwerken, auch genannt Transiente Technische Optimierung (TTO). Ein Gasnetz besteht aus einer Menge von Leitungen, die das Gas von den Lieferanten zu den Abnehmern transportieren. Aufgrund von Reibung an den Rohrwänden geht Gasdruck verloren. Dieser Druckverlust wird mit sogenannten Kompressoren ausgeglichen. Das Ziel der TTO ist es, den Brenngasverbrauch der Kompressoren zu minimieren, wobei der Bedarf der Abnehmer immer gedeckt werden muß. Transiente Optimierung der Gasverteilung stellt eine große Herausforderung an die Forschung auf diesem Gebiet.

Wir formulieren einen gemischt-ganzzahligen Ansatz für das Problem der TTO, der sich auf die zeitabhängigen und diskreten Aspekte konzentriert. Dafür werden die Nichtlinearitäten, die sich aus physikalischen Bedingungen ergeben, unter Verwendung von SOS-Mengen stückweise linear angenähert. Ein Branch-and-Cut Verfahren wird entwickelt, das globale Optimalität in Abhängigkeit von der Approximationsgenauigkeit garantiert.

Hinsichtlich der Nichtlinearitäten gehen wir auf die Güte der Approximationsgitter näher ein, indem wir die Näherungsfehler betrachten. Die SOS Bedingungen werden implizit mittels geeigneter Branchingstrategien modelliert, welche durch entsprechende Preprocessing Techniken verbessert werden.

Ein heuristischer Ansatz basierend auf Simulated Annealing liefert eine obere Schranke in unserem Branch-and-Cut Verfahren. Zur Verbesserung der unteren Schranke verwenden wir zwei Separierungsalgorithmen. Der erste ergibt sich aus theoretischen Studien der so genannten Switching Polytope, die durch Laufzeitbedingungen und Schaltprozesse von Kompressoren definiert werden. Die Verknüpfung unterschiedlicher SOS Bedingungen liefert eine zweite Separierungsstrategie.

Wir präsentieren theoretische Untersuchungen der SOS 2 und SOS 3 Polytope. Diese Polytope ergeben sich aus der Modellierung von SOS Typ 2 und SOS Typ 3 Bedingungen mittels zusätzlicher binärer Variablen. Die Ergebnisse haben keine praktische Bedeutung für unseren Lösungsalgorithmus, jedoch charakterisieren wir Ungleichungen, die Facetten definieren und insgesamt eine vollständige Beschreibung dieser Polytope liefern.

Wir evaluieren das entwickelte Branch-and-Cut Verfahren mittels dreier Testnetzwerke, die uns von unserem Projektpartner E.ON Ruhrgas AG zur Verfügung gestellt wurden. Zwei dieser Netzwerke sind künstlicher Art, da sie für Testzwecke entwickelt wurden und keine realen Netze widerspiegeln. Sie enthalten alle wichtigen Elemente eines Gasnetzes, sind jedoch eher klein. Das dritte Netzwerk beschreibt den größten Teil des Transportnetzes der Ruhrgas AG im Westen Deutschlands. Wir testen Instanzen von drei bis zu 24 gekoppelten Zeitschritten.

Abstract

Natural gas is the third most important energy source in the world. Presently, the consumption of natural gas is increasing the most in comparison to other non-renewable energy sources. Therefore, optimization of gas transport in networks poses a very important industrial problem.

In this thesis we consider the problem of time-dependent optimization in gas networks, also called Transient Technical Optimization (TTO). A gas network consists of a set of pipes to transport the gas from the suppliers to the consumers. Due to friction with the pipe walls gas pressure gets lost. This pressure loss is compensated by so called compressors. The aim of TTO is to minimize the fuel consumption of the compressors, where the demands of consumers have to be satisfied. Transient optimization of gas transmission is one of the great research challenges in this area.

We formulate a mixed integer approach for the problem of TTO which concentrates on time-dependent and discrete aspects. Thereby, the nonlinearities resulting from physical constraints are approximated using SOS (Special Ordered Set) conditions. A branch-and-cut algorithm is developed which guarantees global optimality in dependence on the approximation accuracy.

Concerning the nonlinearities, we discuss the quality of approximation grids by calculating approximation errors. The SOS conditions are implicitly handled via a branching scheme, supported by adequate preprocessing techniques.

A heuristic approach based on simulated annealing yields an upper bound in our branch-and-cut framework. To improve the lower bound, we incorporate two separation algorithms. The first one results from theoretical studies of the so called switching polytopes which are defined by runtime conditions and switching processes of compressors. Linking of different SOS conditions gives a second separation strategy.

We present theoretical investigations of the SOS 2 and SOS 3 polytope. These polytopes arise from the modeling of SOS Type 2 and SOS Type 3 conditions using additional binary variables. The results do not have practical relevance for our solution algorithm, but we characterize facet-defining inequalities providing complete linear descriptions of these polytopes.

We evaluate the developed branch-and-cut algorithm using three test networks provided by our project partner E.ON Ruhrgas AG. Two are of artificial nature, as they were developed for test purposes. They contain all important elements of a gas network, but are rather small. The third network characterizes the major part of the Ruhrgas AG network in Western Germany. We test instances from three up to 24 coupled time steps.

Acknowledgements

First of all, I want to express my thanks to my supervisor Prof. Dr. Alexander Martin. He enabled me the work in this interesting real-world application. He supported me with helpful suggestions, and gave me the freedom necessary to accomplish my research. It was always a friendly discussion atmosphere.

I wish to thank Prof. Dr. Rüdiger Schultz for the acceptance of being referee.
Furthermore, I want to thank Prof. Dr. Jens Lang for giving me support in the development of the grid generator “MeshTool” extracted from KARDOS. Moreover, I am grateful to him, Prof. Dr. Günter Leugering, and Prof. Dr. Stefan Ulbrich for discussions in the field of discretization of partial differential equations.

I would like to thank our industry partners, in particular Klaus Reith from E.ON Ruhrgas AG and Dr. Erwin Sekirnjak from PSI AG who gave me an introduction in the technical details of gas transport and provided their expert knowledge.

Many thanks to all my colleagues of the Optimization group as well as the group of Numerical Analysis and Scientific Computing. I always enjoyed the nice and friendly working atmosphere. Especially, I wish to thank Dr. Peter Lietz for his help with the proof of Theorem 6.4.1, and Sven Herrmann for the support in the determination of Delaunay triangulations. Furthermore, I want to thank Dr. Markus Möller for the discussions and cooperation at the beginning of the project. Finally, I am grateful to Ute Günther, Debora Mahlke, and Andrea Zelmer as well as my sister Anna Moritz for the careful reading of this work, and for their criticism and their suggestions.

My special thanks go to my family. My parents for always supporting me in my studies. My daughter who showed me the important things in life. And finally, I am very thankful to my husband Oliver for his patience and his great belief in me.

Darmstadt, October 2006 Susanne Moritz

Contents

1 Introduction 15

1.1 The Problem . . . 16

1.2 Literature Survey . . . 17

2 Mathematical Background 21

2.1 Optimization Problems . . . 21

2.2 Graphs . . . 22

2.3 Heuristics . . . 24

2.4 Polyhedral Theory . . . 25

2.5 Branch-and-Cut Algorithm . . . 28

3 Approximation of Nonlinear Functions 31

3.1 Approximation of a One-Dimensional Function . . . 32

3.2 Approximation of a Higher Dimensional Function . . . 34

4 Physical Characteristics of a Gas Network 39

4.1 Gas Dynamics in Pipes . . . 39

4.1.1 The Continuity Equation 40
4.1.2 The Momentum Equation 40
4.1.3 The Energy Equation 40
4.1.4 The State Equation of Gas 41
4.1.5 Simplification of the Equations 41
4.1.6 Discretization of the Equations 43
4.2 Compressors 44
4.3 Variables and Constants 47

5 The Model **49**

5.1 Graph Structure and Variables 49
5.2 The Components of a Gas Network 51
5.2.1 Properties of a Pipe 51
5.2.2 Properties of a Connection 52
5.2.3 Properties of a Valve 52
5.2.4 Properties of a Control Valve 53
5.2.5 Properties of a Compressor 54
5.2.6 Properties of Nodes 56
5.3 Linearization of the discretized Partial Differential Equations 57
5.4 Further Transient Conditions 60
5.4.1 Minimum Runtime and Downtime 61
5.4.2 Switching Processes 61

5.4.3 Initial State 62
5.4.4 Fixations of controllable Segments 63
5.4.5 Terminal Condition 63
5.5 Objective Function 64

6 Handling of SOS Conditions **65**

6.1 Linearizing Grids 65
6.2 Branching for SOS Conditions 70
6.3 Preprocessing for SOS Conditions 74
6.4 Separation for SOS Conditions 77

7 A Primal Heuristic: Simulated Annealing **81**

7.1 The Simulated Annealing Algorithm 81
7.1.1 The Idea 81
7.1.2 The Algorithm 83
7.2 The Simulated Annealing Algorithm for TTO 84
7.2.1 Constraint-Handling and Cost Function 85
7.2.2 Neighborhood Structure 86
7.2.3 Initial Solution 89
7.2.4 Cooling Schedule 90
7.2.5 Solution for the Linearized Problem 91
7.3 Computational Results 92

8 Switching Polytopes **95**

8.1 Mathematical Formulation 95

8.2 Literature Survey 96

8.3 Investigation of the Switching Polytopes 98

8.3.1 Polyhedral Studies 98

8.3.2 Separation 104

9 Linearization of Functions: SOS Polytopes **105**

9.1 Binary Linearization Model 105

9.2 The SOS 2 Polytope 109

9.3 The SOS 3 Polytope 116

10 Computational Results **141**

10.1 Test Networks 142

10.2 Approximation Accuracy 143

10.3 Binary Approach using CPLEX 148

10.4 SOS Branching 150

10.5 Heuristic 154

10.6 Preprocessing 158

10.7 Separation for Runtime Conditions and Switching Processes 160

10.8 Separation for SOS Conditions 163

10.9 Tests under practical Conditions 166

10.10 Concluding Remarks 169

11 Conclusions **173**

Chapter 1

Introduction

Natural gas is the third most important energy source in the world, and it shares about 24% of the world-wide primary energy consumption. In comparison to other non-renewable energy sources as petroleum and coal, the consumption of natural gas is currently increasing the most.
In Germany the share of gas in primary energy consumption was 22.7% in 2005, and it increased slightly in comparison to the previous year. The applications of natural gas are versatile. There are three main sectors in which gas is used. At first, there is the residential and commercial sector which comprises private households as well as commercial and service companies. In 2005, it accounted 47% of the total gas consumption. Industry, the second sector, took around a quarter of all gas consumed. Finally, the third main sector is given by power generation with an amount of about 14%.
The German gas demand is mainly covered by imports. In 2005, 85% came from imports and 15% from indigenous fields. The main supplier was Russia providing 34% of gas needed. Norway shared 25% of the total German gas supply. Gas from the Netherlands accounted for 20% of German gas supplies. Finally, The United Kingdom, Denmark, and some other countries provided the remaining 6%.
In contrast to petroleum or coal, natural gas can directly be used as primary energy. Moreover it is eco-friendly and causes less carbon dioxide and nitrogen oxide emission. The natural gas reserves which are economically exploitable at the moment satisfy the needs till 2070. If we add the resources which can be gained in the future, we obtain a range of about 170 years.
Another important fact in connection with natural gas is the liberalization of the European gas market, see [Oos98]. In the near future, the gas transmission companies need strategies to react rapidly and flexibly on the global market. Thus, the development of a control system is necessary. For more informations and facts concerning natural gas, see for example [BGR, GAS, RA].

By these facts the important role of gas transport is shown. Also, the optimization of gas transport in networks poses a difficult problem to the companies. Therefore simulation and optimization of the gas transmission process gain more and more significance, and the development of adequate software tools is of great importance.

1.1 The Problem

In this thesis we consider the problem of time-dependent gas network optimization, also called *Transient Technical Optimization*, or *TTO* for short. The problem is the following. The gas flows through the pipes, and due to friction with the pipe walls gas pressure gets lost. To guarantee the continuation of the gas transport this pressure loss has to be compensated. To this end certain machines, the so called compressors, are used. The gas flows through the compressor and the pressure of the gas is increased at the cost of consuming energy typically fuel gas itself. Further on, there are consumers, in our case local distribution companies, having a certain gas demand. Finally, there are suppliers that deliver the gas. The aim of TTO is to operate the gas transmission in such a way that the consumer demands are satisfied and the compressors work cost-efficiently.

The most common element of a gas network is a pipe. Then there are compressors to increase the gas pressure. Moreover, we find a lot of valves in a network which can be open or close, thus the gas flow can be directed. To give an impression of the dimension of such a gas network, we consider the network driven by our project partner, the German E.ON Ruhrgas AG. It comprises some hundred pipes with a total length of more than 10.000 kilometers. Further on, it features 22 compressor stations with a total of 70 compressor units to generate the pressure needed for the gas transport, see [RA]. Until now, there are no possibilities to optimize with proven quality such a gas network or even major parts of it using a mathematical optimization tool. At a gas company there are people, the so called dispatchers, handling the operation of the gas transmission. They decide which compressors to set in and direct the gas flow based on their knowledge and operating experience.

The subject of this thesis arises from a project with the German industry partners E.ON Ruhrgas AG and PSI AG in collaboration with the Universität Duisburg-Essen and with the Konrad-Zuse-Zentrum für Informationstechnik in Berlin. The idea is to develop an optimization tool for the problem of TTO supporting the dispatcher at work. The dispatcher specifies the state of the gas network, and then the optimization tool calculates an optimal control of the network for the subsequent time. As all network parameters must be adjusted on an hourly basis the solution time for such an optimization software is limited to 15 minutes.

The problem of TTO includes nonlinear, combinatorial, and stochastic aspects. The pipe hydraulics describing the gas flow through the pipes are given by a system of nonlinear partial differential equations and the state equation of gas. Naturally, the problem is time-dependent as the hydraulic of pipes is transient. Blending different gas qualities also yields a nonlinear process. Moreover the fuel gas consumption of a compressor is given by a nonlinear function. On the other hand, there are discrete aspects. A valve can be open or close. Each compressor has its state, either it operates or it is shut down. Furthermore the modeling of delivery contracts can lead to combinatorial aspects.

Usually stochastic aspects have to be considered as well, e.g., sales quantities of gas dependent on weather conditions or on the liberalized gas market. Altogether, we receive a very complex mixed integer nonlinear optimization problem including stochastic aspects, and so far it is not possible to solve this problem.

In this thesis we focus on the time-dependent and discrete aspects of the problem, whereas we approximate the nonlinearities to obtain a mixed integer linear programming approach. Thus we neglect the stochastic aspects and develop a mathematical model for the remaining problem where an adequate handling of the nonlinearities is needed.

The work in the field of gas network optimization in the research group Optimization at the Technische Universität Darmstadt began in 2001. In a first step the stationary case of gas network optimization was treated where just one time step is considered [Möl04]. This thesis is based on [Möl04] and extends the developed strategies to the time-dependent case. For the transient case further work has to be done and additional aspects must be taken into account. Because of the time dependence, we need an appropriate modeling of the gas dynamics in the pipes. We also obtain more conditions than in the stationary case, i.e., there are min-up and min-down times as well as switching costs for compressors.

In this work we tailor a branch-and-cut algorithm to solve the mixed integer approach for the problem of TTO. Before we present different approaches for the problem of gas network optimization known from the literature, we give an outline of this thesis. In Chapter 2 we consider the mathematical basics concerning our problem. In the third chapter, we concentrate on approximation of nonlinear functions. Thereafter we describe the physical characteristics of a gas network comprising the gas dynamics in a pipe and the fuel consumption of a compressor. In Chapter 5 we present our mixed integer approach, whereas we approximate the nonlinearities by piece-wise linear functions using the concept of so called SOS conditions. In the following chapter we concentrate on the handling of these SOS conditions to model them implicitly in our branch-and-cut algorithm. A simulated annealing algorithm for the problem of TTO is specified in Chapter 7 to obtain a feasible solution yielding an upper bound in our branch-and-cut framework. In Chapter 8 we study switching polytopes resulting from minimum runtime and downtime conditions and from switching processes of a machine. Based on these studies, such conditions for the compressors are incorporated implicitly in our solution algorithm using a separation procedure. In Chapter 9 theoretical investigations can be found concerning the modeling of SOS conditions via additional binary variables. In Chapter 10 we conclude our work with computational results showing the applicability of our developed algorithm in practice. Finally, we give some concluding remarks and suggestions for further improvements and investigations.

1.2 Literature Survey

In the literature various approaches for gas network optimization can be found, but none of them covers all nonlinear, combinatorial, time-dependent, and stochastic aspects. In the following, we mention different solution methods and cite corresponding literature. Most papers concentrate on the stationary case. Therefore we begin with a summary of literature considering this easier case where just one time step is considered.

In [Zim75], [Gop79] and [Car98] dynamic programming is used. The first two restrict to a simplified network structure, namely a directed graph consisting of pipes and compressors without cycles, a so called "gunbarrel system". [Car98] extends the dynamic approach to cyclic gas networks. This paper also gives a general overview for the application of dynamical programming in gas transmission systems.

Because of the nonlinear elements the problem is often tackled by nonlinear optimization methods. In [Jen93] gradient techniques are used to optimize the steady-state of gas transport assuming fixed states of compressors and valves. [Krá93] concentrates on the modeling of compressor stations and investigates the configurations of different machines. Note that these methods just yield locally optimal solutions.

Heuristic approaches can also be found for the stationary case of gas network optimization like for example simulated annealing [WSD98], genetic algorithms [Car98] or a hybrid heuristic approach [BSRM05]. [WSD98] concentrates on the optimization of compressor stations, i. e., the authors look for optimal configurations and power settings for a large number of compressors arranged in series or in parallel. [Car98] compares a genetic algorithm implementation with a dynamic programming approach. Finally, [BSRM05] combines dynamic programming with tabu search in a two-stage iterative procedure. Notice that none of these heuristics yields a provable optimal solution.

Often iterative solution methods are used to solve the problem. There the nonlinearities are approximated, the resulting (mixed-integer) linear program is solved, and the process is continued until convergence is reached. In [PW84] we find such an approach. Here the problem is iteratively relinearized about the optimum of the linearized problem until convergence is obtained. Similarly, the optimization problems are solved in an iterative fashion in [Sek98, Sek00] using sequential linear programming. There the nonlinearities are linear approximated using Taylor expansion or piece-wise linear functions, and convergence is enforced via certain penalty factors. [SW00] also uses a stepwise solution process based on a modified simplex algorithm. The disadvantage of such iterative methods is that they cannot guarantee global convergence.

A mixed integer approach for the stationary case is also applied in [Hac02] and [Möl04, MMM06] whereas the nonlinearities are approximated via so called SOS conditions. In [Hac02], the author regards models for medium- and long-term optimization of gas transmission in networks. In these cases, less physical and technical details are required than for short-time optimization as considered in [Möl04, MMM06]. Moreover, [Hac02] restricts to SOS conditions of Type 2. [Möl04, MMM06] extend the SOS concepts to the general higher dimensional case. Suitable branching strategies are developed, and a separation algorithm combining different SOS conditions supports the solution algorithm. Note that these approaches using SOS conditions yield global optimal solutions in dependence on the approximation accuracy.

For the more challenging transient case less literature is known.

In [JC00], the authors solve the nonlinear optimization problem by control theory using gradient techniques, whereas they restrict on gunbarrel systems. They receive a local minimum, and suppose the uniqueness of their solutions.
As for the stationary case, nonlinear optimization methods are also applied for the time-dependent case. [Vos93] uses gradient methods whereas the author exploits properties of the system to obtain an efficient calculation of the gradient. In [ES03, ES05] suitable space and time discretizations for the partial differential equations are presented, and the resulting nonlinear optimization problem is solved using an SQP method. Both approaches assume predetermined states of the switchable segments in a gas network, and they can only guarantee local optimality.
[Nei04] applies a nonlinear approach to solve the transient gas network optimization problem assuming given states of compressors and valves. This approach is compared with a similar model, where the nonlinearities are piece-wise linear approximated using planes [Hei02]. The nonlinear approach maps more precisely the reality, but in case of the linear model the running times are much better.
A linear model for transient gas flow is also presented in [NW03] where the system of partial differential equations is substituted by a simple line-pack model of three sections for a pipe. After calibration of the parameters, this warehouse approach is compared with numerical simulations.
[Wes04] concentrates on stochastic aspects of TTO and uses a coarse approximation of the nonlinearities. An uncertain demand of the consumers is assumed and a two-stage stochastic optimization problem is formulated.
In [Sek00, KRS00] the problem of transient gas network optimization is tackled using sequential linear programming. The nonlinearities are approximated by Taylor expansion and polygons using the operating point. [KRS00] present the resulting mixed integer formulation and give computational results evaluated using a transient simulator.
Other mixed integer formulations are considered in [Hac02] and [Tom88] using the concept of SOS of Type 2. Medium- and long-term planning is discussed in [Hac02]. [Tom88] dwells on contractual conditions for suppliers and consumers, and considers long-term optimization.
Often simulation of gas transport in pipes is regarded. [NW03] considers the gas flow in a pipe and compare this simulation to a linearized approach. In [Bal05], a hierarchical modeling of the partial differential equations describing the gas flow in a pipe is presented, and the different models are compared. Especially, the instationary approaches are used to evaluate the stationary one in [Sek00] as well as in [Möl04, MMM06]. [BHK06] focuses on coupling conditions in gas networks.

As we see from this overview, none of the approaches fits to our intention in gas network optimization. Some methods just yield local optima, furthermore either combinatorial aspects are neglected or coarse approximations for the nonlinearities are applied. Recall that we concentrate on time dependence and combinatorial aspects. Moreover, we want to guarantee global optimality in dependence on the approximation accuracy of the nonlinear functions.

Chapter 2

Mathematical Background

In this chapter we give a brief introduction into the most important mathematical fields needed in this thesis. At first, we state formulations of optimization problems considered in this work. Then, we summarize some basics of graph theory. In the third section, we specify the term heuristic and meta-heuristic. In the following section, we concentrate on the fundamentals of polyhedral theory. Finally, we dwell on the general structure of a branch-and-cut algorithm. In all sections we cite literature which provides a deeper insight in the subject.

2.1 Optimization Problems

We want to formulate the problem of transient gas network optimization as a linear *mixed integer program* (MIP). A general definition can be found for example in [NW88, Wol98].
A MIP can be written in the following form

$$\begin{array}{ll} \min & c^T x + h^T y \\ & Ax + Gy \leq b \\ & x \in \mathbb{Z}^n,\ y \in \mathbb{R}^p, \end{array} \tag{2.1}$$

where $c \in \mathbb{R}^n$, $h \in \mathbb{R}^p$ and $b \in \mathbb{R}^m$ are vectors, and $A \in \mathbb{R}^{m\times n}$ and $G \in \mathbb{R}^{m\times p}$ are matrices. It is called ‘mixed’ because it contains integer variables x as well as continuous variables y. If there are no continuous variables, i.e., $p = 0$, (2.1) is called (pure) *integer program*. If $n = 0$, i.e., there are no integer variables, it is a *linear program*. The linear term $c^T x + h^T y$ is called *objective function*, and $Ax + Gy \leq b$ defines the *constraints*. We say that $(x, y) \in \mathbb{Z}^n \times \mathbb{R}^p$ is a *feasible solution*, if it satisfies all constraints. Further on, a feasible solution (x_o, y_o) is called an *optimal solution*, if $c^T x_o + h^T y_o \leq c^T x + h^T y$ for all feasible solutions (x, y).
At present, among the most successful algorithms to solve a MIP are the so called *branch-and-cut*

algorithms, see Section 2.5, which are based on LP relaxation. That is also the approach which we follow in this thesis. We formulate the problem of TTO as an MIP, see Chapter 5. Then we develop a branch-and-cut algorithm to solve it.

Originally, the gas network optimization problem is a nonlinear optimization problem. Here, we use the formulation given in [MF00]. We need this formulation in connection with the simulated annealing heuristic, see Chapter 7.
Let $f : \mathbb{R}^n \to \mathbb{R}$, $h_j : \mathbb{R}^n \to \mathbb{R}$ for $j \in \{1, ..., q\}$ and $g_j : \mathbb{R}^n \to \mathbb{R}$ for $j \in \{q+1, ..., m\}$ be functions. Then

$$\begin{array}{lll} \min & f(x) & \\ & h_j(x) = 0 & j = 1, \ldots, q \\ & g_j(x) \leq 0 & j = q+1, \ldots, m \end{array} \tag{2.2}$$

is called a *nonlinear optimization problem*.

Originally, the simulated annealing algorithm was developed for solving large combinatorial optimization problems. Therefore we specify this kind of problem, see for example [NW88, Wol98]. Let $N = \{1, \ldots, n\}$ be a finite set and let $c = (c_1, \ldots, c_n)$ be an n-vector. Further let $\mathscr{F} \subseteq \mathscr{P}(N)$ be a subset of the power set of N, denoting the set of feasible subsets of N. We define $c(F) = \sum_{j \in F} c_j$ for $F \subseteq N$. The *combinatorial optimization problem* is given by

$$\min_{F \in \mathscr{F}} c(F), \tag{2.3}$$

hence finding a minimum weight feasible subset.

2.2 Graphs

In this section we give definitions and results of graph theory needed in this thesis. We model the gas network as directed graph, see Section 5.1. In connection with our polyhedral studies for SOS conditions with binary variables, we use the decomposition of a connected graph in so called cycle with ears and paths, see Chapter 9. In our descriptions, we follow the books of Diestel [Die00] and of Lovász and Plummer [LP86] and the article of Whitney [Whi32].

A *graph* is a pair $G = (V, E)$ of sets, where V is the *node* set and $E \subseteq V \times V$ is the *edge* set. All graphs we consider are finite. We denote an edge $e \in E$ with endnodes u and v by uv. We will not allow loops, i.e., edges of the form uu. Given an edge uv in a graph G, edge uv is said to be *incident* with nodes u and v, and nodes u and v are said to be *adjacent*. Two edges are *adjacent* if they have a common node.
If $V' \subseteq V$ and $E' \subseteq E$, then $G' = (V', E')$ is called a *subgraph* of G. By $G[V']$ we denote the graph with node set V', whose edges are exactly those edges of G joining two nodes of V', and

call it the subgraph of G *induced by* V'.
An alternating sequence of nodes and edges of the form $v_0, v_0v_1, v_1, v_1v_2, v_2, ..., v_{k-1}, v_{k-1}v_k, v_k$, where all nodes are distinct, is called a *path* (or a *chain*). The *length* of a path is defined as the number of edges in it. A *suspended chain* is a chain containing two or more edges such that no node of the chain other than the first and the last is incident to any other edge in the graph, and these two endnodes are each on at least two other edges. A path $v_0, v_0v_1, v_1, v_1v_2, v_2, ..., v_{k-1}, v_{k-1}v_k$ with $v_0 = v_k$ and $k \geq 2$ is called a *cycle*.
A graph is *connected* if any two of its nodes are joined by a path. A *component* of G is a maximal connected subgraph of G. A graph is called *cyclicly connected* if any two of its nodes are contained in a cycle.
A set of nodes S in a connected graph G is a *cutset* if $G[V \setminus S]$ is not connected. If S is a cutset of G consisting of a single node v, the node v is called a *cutpoint* (or *cutvertex*) of G. A connected graph containing no cutpoint is called a *non-separable* or *2-(node)connected* graph. Geometrically, G is a non-separable graph if there do not exist two graphs H_1 and H_2, each containing at least one edge, which form G if they are joined at a node. If G is not non-separable, it is said to be *separable*. Clearly, a connected graph is non-separable if we cannot divide it into two graphs, each containing at least one edge, at a single node. For example a graph consisting of a single edge is non-separable, but a graph consisting of two edges uv and vw, $u \neq w$, is separable.
The following theorems can be found in [Whi32].
At first, we give a characterization of a cut vertex.

Theorem 2.2.1 *Let G be a connected graph. A necessary and sufficient condition that the node c be a cut vertex of G is that there exist two nodes u and v in G, each distinct from c, such that every path from u to v passes through c.*

We consider a further characterization of a non-separable graph.

Theorem 2.2.2 *Let G be a graph containing at least two edges. A necessary and sufficient condition that G be non-separable is that it be cyclicly connected.*

Now we look at decomposition of graphs. As mentioned above, a separable graph can be broken into two graphs having just a single node in common. Repeating this procedure we decompose the graph G in non-separable parts and say that G is separated into its *components*. Here we mention two results in this context.

Theorem 2.2.3 *Every non-separable subgraph of G is contained wholly in one of the components of G.*

Theorem 2.2.4 *A graph G may be decomposed into its components in a unique manner.*

At last we consider a construction method for non-separable graphs.

Theorem 2.2.5 *We can build up any non-separable graph containing at least two edges by taking first a cycle, then adding successively edges or suspended chains, so that at any stage of the construction we have a non-separable graph.*

This method is useful for proving statements for non-separable graphs inductively. This decomposition is also known as *ear decomposition* or *cycle with ears* (see [LP86]).
If we regard a connected graph and separate it into its components, hence into its non-separable parts, then all components with two or more edges are cycles or cycles with ears. The components consisting of just one edge can be pieced together to (several) paths of maximal length. So we can decompose a connected graph in non-separables sets (cycle with ears) and paths.
We use this form of decomposition in the proof for the complete linear description of the SOS 3 polytope, see Theorem 9.3.5.

Finally, we give the definition of a directed graph needed for the modeling of a gas network. A *directed* graph (or *digraph*) $G = (V, E)$ consists of a set of nodes V and a set of *arcs* E. An arc $e = (u, v) \in V \times V$ is an ordered tuple. u is called the *tail* of e, and v is the *head* of it. Considering a node $v \in V$, $\delta^+(v) = \{(u, w) \in E |\ u = v\}$ defines the the set of all *outgoing* arcs of v. Similarly, $\delta^-(v) = \{(u, w) \in E |\ w = v\}$ denotes the set of all ingoing arcs in v.

2.3 Heuristics

For many optimization problems and real world applications it is not only hard to find optimal solutions, even the specification of a feasible solution is very difficult. That is also the case for our transient gas network optimization problem. However, feasible solutions are necessary in the frame of a branch-and-cut algorithm for the bounding process, see Section 2.5. Therefore methods are needed that find good solutions in acceptable running times. Such methods are called *heuristics*. In [Ree93] we find the following definition.

Definition 2.3.1 [Ree93] *A* heuristic *is a technique which seeks good (i.e., near-optimal) solutions at a reasonable computational cost without being able to guarantee either feasibility or optimality, or even in many cases to state how close to optimality a particular feasible solution is.*

A special class of these methods are the so called *meta-heuristics*. They are superior strategies which control, modify and combine different methods and build new hybrid concepts. The following definition can be found in [OK96].

Definition 2.3.2 [OK96] *A* meta-heuristic *is an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search spaces using learning strategies to structure information in order to find efficiently near-optimal solutions.*

Simulated annealing, tabu search, and genetic algorithms are examples for meta-heuristics. Under certain assumptions theoretical convergence can be proven for some meta-heuristics (see [OK96]). But for most practical applications these assumptions cannot be fulfilled. Thus meta-heuristics do not guarantee optimality but they yield good feasible solutions.

We used the approach of simulated annealing for our TTO problem (see [Mah05, MMMar]). In Chapter 7 a detailed description of the algorithm can be found. The basic idea of the simulated annealing algorithm is given by the *local search algorithms*. These methods are based on an iterative improvement of the objective function value. Starting from a feasible initial solution S the method tries to find a better solution in the neighborhood $N(S)$ of S. $N(S)$ is obtained by a so called move-generation mechanism (or transformation). If a better solution S' is found, the algorithm continues searching in the neighborhood $N(S')$, otherwise it stops. Here we give a basic version of a local search method for a combinatorial optimization problem (2.3), see [OK96].

Algorithm 1 Local Search

```
1: Get an initial solution $S$ and compute its objective $c(S)$.
2: while there is an untested neighbor $S' \in N(S)$ do
3:     Generate sequentially a trial $S' \in N(S)$ and compute $c(S')$.
4:     If $c(S') < c(S)$ then set $S = S'$. Otherwise retain $S$. Goto 2.
5: end while
6: Terminate the search and return $S$ as the best found solution.
```

Local search is applicable to many kind of problems since just the specification of an initial solution, a cost function, and a generation rule for the neighborhood are needed. So it is a very flexible and robust method. The main disadvantage is that it often terminates in a local optimum. Moreover the resulting solution highly depends on the initial solution and the neighborhood structure.

2.4 Polyhedral Theory

In this section we concentrate on fundamentals of polyhedral theory. We need this basics for the studies of the switching polytopes in Chapter 8 and the SOS polytopes in Chapter 9. For our descriptions, we refer to the book [NW88].

We begin with the definition of a polyhedron.

Definition 2.4.1 *A* polyhedron $P \subseteq \mathbb{R}^n$ *is the set of points that satisfy a finite number of linear inequalities, that is,* $P = \{x \in \mathbb{R}^n | \ Ax \leq b\}$*, where* A *is an* $m \times n$ *matrix and* b *is an* m*-vector.*
A polyhedron is said to be rational *if there exists an* $m' \times n$ *matrix* A' *and an* m'*-vector* b' *with rational coefficients such that* $P = \{x \in \mathbb{R}^n | \ A'x \leq b'\}$*.*
A polyhedron $P \subseteq \mathbb{R}^n$ *is* bounded *if there exists an* $\omega \geq 0$ *such that* $P \subseteq \{x \in \mathbb{R}^n | \ -\omega \leq x_j \leq \omega \ \text{ for } \ j = 1, \ldots, n\}$*. A bounded polyhedron is called a* polytope.

In our studies, we just consider polytopes. Further on, we use the dimension of a polyhedron.

Definition 2.4.2 *A polyhedron* P *is of* dimension k*, denoted by dim* $(P) = k$*, if the maximum number of affinely independent points in* P *is* $k+1$*.*
A polyhedron $P \subseteq \mathbb{R}^n$ *is* full-dimensional *if dim* $(P) = n$*.*

The switching and SOS polytopes investigated in this thesis are not full-dimensional. If a polyhedron $P = \{x \in \mathbb{R}^n | \ Ax \leq b\}$ is not full-dimensional, then at least one of the inequalities $a^i x \leq b_i$ is satisfied at equality by all points of P. More precisely, if $(A^{=}_{P}, b^{=}_{P})$ are the rows of (A, b) that are satisfied at equality by all points of P (it is also called the *equality* set of the representation (A, b) of P), then the dimension of the polyhedron can be calculated via the following formula.

Proposition 2.4.3 *If* $P \subseteq \mathbb{R}^n$*, then dim* $(P) + rank(A^{=}_{P}, b^{=}_{P}) = n$*.*

Here, rank$(A^{=}_{P}, b^{=}_{P})$ denotes the rank of the matrix $(A^{=}_{P}, b^{=}_{P})$.
Considering a polyhedron P, we are interested in a minimal description of it, i.e., we want to know which are the inequalities that are necessary for describing it. In this context we need the terms valid inequality and face.

Definition 2.4.4 *The inequality* $ax \leq \alpha$ *is called a* valid inequality *for* P *if it is satisfied by all points in* P*.*

Definition 2.4.5 *A set* $F \subseteq P$ *is called a* face *of* P*, if there exists a valid inequality* $ax \leq \alpha$ *for* P *with* $F = P \cap \{x \in \mathbb{R}^n | \ ax = \alpha\}$*. We say that* $ax \leq \alpha$ induces F*. A face is said to be* proper *if* $\emptyset \neq F \neq P$*. When* F *is nonempty, we say that* $ax \leq \alpha$ supports P*.*

If F is a proper face of P, then dim $(F) <$ dim (P) holds, see Proposition 2.4.3. Now, we can characterize the inequalities that are necessary in the description of P.

Definition 2.4.6 *A face* F *of a polyhedron* P *is a* facet *of* P *if dim* $(F) =$ *dim* $(P) - 1$*.*

Proposition 2.4.7 *For each facet F of P, one of the inequalities representing F is necessary in the description of P.*

On the other hand, the facets are also sufficient for the description of P, i.e., all faces with dimension less than dim $(P) - 1$ are irrelevant to the description of P.
Now we look at the proper faces of lowest dimension.

Definition 2.4.8 *$x \in P$ is an* extreme point *(or* vertex*) of P if there do not exist $x^1, x^2 \in P$, $x^1 \neq x^2$, such that $x = \frac{1}{2}x^1 + \frac{1}{2}x^2$.*

We obtain the following characterization of an extreme point.

Proposition 2.4.9 *x is an extreme point of P if and only if x is a zero-dimensional face of P.*

Hence, if $(A_x^=, b_x^=)$ gives the rows of (A, b) that are satisfied at equality by the vertex x of P, then rank$(A_x^=, b_x^=) = n$.

The polytopes we consider in this thesis result from integer programs in case of the so called switching polytope (see Chapter 8) and from mixed integer programs for the SOS polytopes (see Chapter 9). The set $S \subseteq \mathbb{Z}^n \times \mathbb{R}^p$ of feasible points for these problems is described implicitly via a linear inequality system resulting in a set of (mixed) integer solutions of the form $S = \{x \in \mathbb{Z}^n,\ y \in \mathbb{R}^p |\ Ax + Gy \leq b\}$. Now, we are interested in finding a linear inequality description of this set S.

In Chapter 8, we specify a complete linear inequality description for the switching polytope. For proving this fact, we need the concept of totally dual integrality. Therefore, we conclude this section with the terminology of integral polyhedron, see for example [Sch86, NW88].
At first we introduce the definitions of an integral polyhedron and of the property totally dual integral for a rational system.

Definition 2.4.10 *A rational polyhedron P is* integral *if P is the convex hull of its integral vectors, i.e., $P = \mathrm{conv}(P \cap \mathbb{Z}^n)$.*

Definition 2.4.11 *Let $A \in \mathbb{Q}^{m\times n}$ and $b \in \mathbb{Q}^m$. The system $Ax \leq b$ is said to be* totally dual integral *(TDI), if for each integral vector $c \in \mathbb{Z}^m$ for which $\min\{b^\top y \mid A^\top y = c,\ y \geq 0\}$ is finite , there exists an integral optimum solution $y^\star \in \mathbb{Z}^m$, i.e., $A^\top y^\star = c$, $y^\star \geq 0$ and $b^\top y^\star = \min\{b^\top y \mid A^\top y = c,\ y \geq 0\}$.*

The following theorem yields the connection between a TDI-system and the integrality of the corresponding polyhedron.

Theorem 2.4.12 *If* $Ax \leq b$ *is a TDI-system and* b *is integral, the polyhedron* $\{x \mid Ax \leq b\}$ *is integral.*

2.5 Branch-and-Cut Algorithm

As already mentioned, branch-and-cut algorithms are among the most successful algorithms to solve mixed integer programs. For the gas network optimization problem, we design a branch-and-cut algorithm to solve it. In this section we sketch the structure of such an algorithm, see also [NW88, Wol98]. In principle, it combines two methods. The branch-and-bound algorithm which is based on a 'divide and conquer' approach, and a cutting plane phase which helps to tighten the dual bound.

We consider a MIP of the form (2.1). Let $S := \{(x,y) \in \mathbb{Z}^n \times \mathbb{R}^p |\ Ax + Gy \leq b\}$ be the set of feasible solutions. Now we look at the problem

$$\begin{aligned} z = \min\ & c^T x + h^T y \\ & (x,y) \in S. \end{aligned} \tag{2.4}$$

The idea is to break the problem into smaller problems that are possibly easier to solve, and then to put the solution information together to solve the original problem. Thus we split S into subsets $S_i \subseteq S$, $i = 1, \dots, k$, with $S = \cup_{i=1}^k S_i$ and try to solve the corresponding subproblems. Note that if $z^i = \min\{c^T x + h^T y \mid (x,y) \in S_i\}$, for $i = 1, \dots, k$, then we receive $z = \min_i z^i$ for the original problem.

In general, the subproblems might be as difficult to solve as the original problem. Therefore we continue the splitting process and create subproblems of the subproblems, and so on. The typical representation for such an 'divide and conquer' approach yields an enumeration tree, the so called *branch-and-bound tree*. The root node of this tree is given by the original problem. If a problem is split into two or more subproblems, we call it *father* and its subproblems *sons*. The unsolved problems are the leaves of the tree.

Since complete enumeration is inefficient, we exploit bounds on the values z^i of the subproblems to investigate the branch-and-bound tree intelligently. The idea is to determine lower and upper bounds for the values z^i and to use these informations to prune branches from the tree. This is the so called *bounding process*. Thereto we give the following observation.

Proposition 2.5.1 *Let* $S = \cup_{i=1}^k S_i$ *be a decomposition of* S *into smaller sets and let* $z^i = \min\{c^T x + h^T y \mid (x,y) \in S_i\}$, *for* $i = 1, \dots, k$. *Further on, let* $\overline{z}^i$ *be an upper bound on* z^i

and $\underline{z}^i$ *be a lower bound on it. Then* $\overline{z} = \min_i \overline{z}^i$ *is an upper bound for* z *and* $\underline{z} = \min_i \underline{z}^i$ *is a lower bound for it.*

There exist three criteria to eliminate branches from the branch-and-bound tree. The first one is pruning by optimality. If the lower and upper bounds of a subproblem are equal, $\underline{z}^i = \overline{z}^i$, the corresponding subset S_i must not be considered further, and it can be pruned by optimality. The second criterion is pruning by bound. In that case, there is a subproblem where the lower bound $\underline{z}^i$ is greater than the best upper bound of the original problem found so far, i.e., $\underline{z}^i \geq \overline{z}$. As the optimal value has value at most $\overline{z}$, no optimal solution can lie in the set S_i. Therefore S_i can be pruned by bound. Finally, the last criterion is pruning by infeasibility. Obviously, if $S_i = \emptyset$, this node in the tree can be pruned by infeasibility. Exploiting these criteria, we can handle a part of the enumeration process implicitly.

Now the question arises how to obtain upper and lower bounds for the problems. Notice that each feasible solution provides an upper bound. Having good feasible solutions, and hence good upper bounds, is crucial for the bounding process to eliminate nodes from the branch-and-bound tree. Often, problem dependent heuristics are developed to determine good feasible solutions. In case of the TTO problem, we choose the concept of simulated annealing to produce a good solution, see Chapter 7.

To obtain a lower bound, we consider a relaxation of the problem. Therefore we choose a set $\tilde{S}_i \subseteq \mathbb{Z}^n \times \mathbb{R}^p$ with $S_i \subseteq \tilde{S}_i$ and optimize the objective function over $\tilde{S}_i$. For example, this can be the *LP relaxation* where the integrality conditions in (2.4) are neglected. This is also the approach we use in this thesis. In case of the gas network optimization problem, we do not just drop the integrality requirements, but we also disregard the SOS conditions for the linearized functions and incorporate them implicitly in the branching scheme, see Chapter 3 and 6. In the course of the algorithm it may happen that the optimal solution of the LP relaxation of a subproblem is also a feasible solution for the original problem, i.e., it lies in S. In that case we receive an upper bound that is possibly better than the incumbent, and thus helps us to tighten the tree.

In the description of the algorithm there are two steps where we have to make a decision. The first one, is which node of the tree to process next, i.e., which subproblem should be considered. The second question is how to split a problem into subproblems. Assuming the LP relaxation to calculate lower bounds in the nodes, there is a typical way to split a problem into two subproblems. Regarding the optimal solution $(x^\star, y^\star)$ of the LP realaxation, we choose a fractional variable $x_i^\star$ that must be integer and create two subproblems by adding $x_i \geq \lceil x_i^\star \rceil$ or $x_i \leq \lfloor x_i^\star \rfloor$ to the formulation. But considering this approach, we do not know which fractional integer variable to choose. We refer to [LP79] for selection strategies within the scope of a branch-and-bound algorithm.

The solution algorithm described till now is called *branch-and-bound algorithm*, since we do not use cuts. To obtain a *branch-and-cut algorithm*, we have to incorporate the cutting plane phase in the branch-and-bound algorithm. The idea is to tighten the lower bounds throughout the branch-and-bound tree by adding so called *cutting planes*. In the following we describe this approach. Considering a node of the branch-and-bound tree, i.e., a subproblem of the original problem, we solve the LP relaxation and receive an optimal solution $(x^\star, y^\star)$ (which gives us a lower bound at this node). If $x^\star$ is integral, we found a feasible solution for (2.4) which possibly improves the best upper bound in the tree. Otherwise, we try to find a valid inequality for the polyhedron $\operatorname{conv}\{(x,y) \in \mathbb{Z}^n \times \mathbb{R}^p | \; Ax + Gy \leq b\} = \operatorname{conv}(S)$ that is violated by the LP solution $(x^\star, y^\star)$. Hence, we look for a valid inequality $d^\top x + e^\top y \leq \alpha$ for the original problem such that the violation of it by the LP solution, $d^\top x^\star + e^\top y^\star > \alpha$, is preferably big. We concentrate on facet-defining inequalities of the polyhedron or some subpolyhedron to receive faces of high dimension. The problem of finding such an inequality is called the *separation problem*. If we are able to specify a violated inequality, we can strengthen the LP relaxation and the lower bound at the node is possibly improved. We continue this process until $(x^\star, y^\star)$ is a feasible solution or if no more violated inequalities are determined. Altogether, this is the so called *cutting plan phase*. After termination of this phase, we continue with the branch-and-bound process, i.e., we split a problem into subproblems.

In case of the problem of TTO, we incorporate two kinds of cutting planes. The first class treats combinations of SOS conditions, see Section 6.4. And the second one comes from switching processes and runtime conditions for compressors, see Chapter 8. Both kinds of inequalities result from studies of subpolytopes of the gas network optimization formulation. For the switching polytopes, we are even able to determine facets and integrate them in the cutting plane phase. The following algorithm summarizes the described branch-and-cut algorithm.

Algorithm 2 Branch-and-Cut Algorithm

```
1:  Let $L$ be a list of unsolved problems. Initialize $L$ with (2.4).
2:  Set upper bound $\overline{z} = \infty$.
3:  repeat
4:      Choose a problem $P$ from $L$.
5:      repeat
6:          Solve LP relaxation of $P$. Let $(x^\star, y^\star)$ be an optimal solution.
7:          Let $\underline{z}_P = c^\top x^\star + h^\top y^\star$ be a lower bound for $P$.
8:          if $(x^\star, y^\star) \in \mathbb{Z}^n \times \mathbb{R}^p$ then
9:              if $\overline{z} > \underline{z}_P$ then
10:                 Let $\overline{z} = \underline{z}_P$.
11:                 Update best feasible solution $(\tilde{x}, \tilde{y}) = (x^\star, y^\star)$.
12:                 Delete from $L$ all problems $P'$ with $\underline{z}_{P'} \geq \overline{z}$.
13:             end if
14:             Goto 19.
15:         end if
16:         Find violated inequalities and add them to the LP.
17:     until No more violated inequality can be found.
18:     Split $P$ into subproblems and add them to list $L$.
19: until $L = \emptyset$
20: If $\overline{z} < \infty$, $(\tilde{x}, \tilde{y})$ is an optimal solution for (2.4). Otherwise the problem is infeasible.
```

Chapter 3

Approximation of Nonlinear Functions

The problem arising from TTO contains several nonlinear elements such as the fuel gas consumption of compressors and the system of partial differential equations describing the gas dynamic in pipes. Since we want to formulate this problem as an MIP of the form (2.1), we need to linearize these components.

In this chapter we give a brief introduction in how to approximate nonlinear functions by piece-wise linear ones. A lot of literature can be found concerning the approximation of nonlinear functions by piece-wise linear ones [Bea79, BF76, BT70, Dan63, LW01, KdFJN04, MM57, NW88, Pad00, Tom81, Wil98, Möl04, MMM06].

Most approaches focus on nonlinear separable functions. A separable function $h : \mathbb{R}^n \to \mathbb{R}$ is a function that can be written as $h(x) = \sum_{i=1}^{n} h^i(x_i)$ with one-dimensional functions $h^i : \mathbb{R} \to \mathbb{R}$. Hence literature concentrates on linearization concepts for one-dimensional functions. There exist three different linearization approaches. The first one uses so called *Special Ordered Sets of Type 2*, briefly SOS Type 2. These are sets with a certain property which is modeled implicitly during the branch-and-bound phase [BT70, BF76, Bea79]. The two further approaches introduce additional binary variables. One is the so called *convex combination* or *lambda method*, see [Dan63, NW88, Wil98, LW01]. Here the SOS Type 2 condition is modeled explicitly using additional binary variables. The second binary approach is known as *incremental* or *delta method* [Dan63, MM57, Wil98].

For the approximation of non-separable functions, i.e., higher-dimensional functions, less literature can be found. [Tom81, Möl04, MMM06] yield an extension of the SOS concepts to general n-dimensional functions. A generalization of the lambda and delta method can be found in [Wil98, LW01].

In the following we present the three approaches for the piece-wise linear approximation of a one-dimensional nonlinear function. Thereafter, we extend these ideas to functions in higher dimensions.

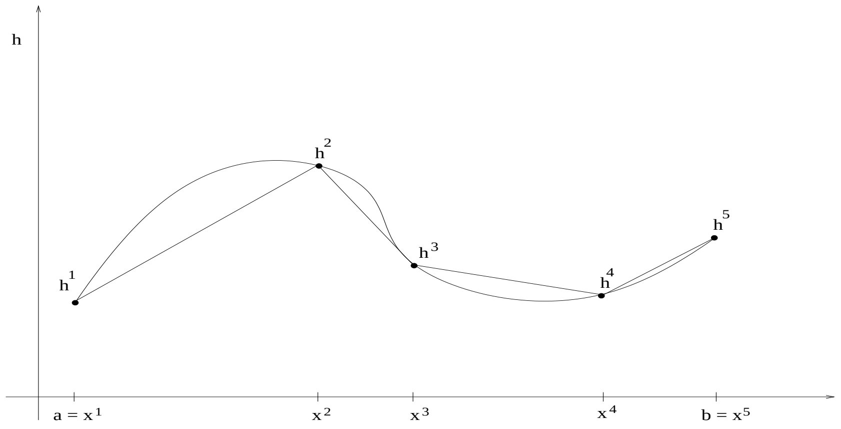

Figure 3.1: Approximation of a nonlinear function

3.1 Approximation of a One-Dimensional Function

We consider a nonlinear function of the form $h : D \to \mathbb{R}$ defined on an interval $D := [a, b]$ with $a, b \in \mathbb{R}$. The idea of approximating such a function goes back to the traditional way known from pertinent textbooks, see for example [Dan63, NW88]. By dividing the interval in parts, we receive grid points $a = x^1 \leq x^2 \leq \cdots \leq x^k = b$ with corresponding function values $h(x^i)$, $i \in \Lambda := \{1, \ldots, k\}$. We want to approximate h by the piece-wise linear function that is indicated in Figure 3.1. Therefore, we introduce a variable λ^i for each grid point $i \in \Lambda$ and approximate the function value $h(x)$ for $x \in D$ by the following convex combination.

$$\sum_{i \in \Lambda} \lambda^i = 1 \tag{3.1}$$

$$h(x) \approx \sum_{i \in \Lambda} h(x^i)\ \lambda^i \tag{3.2}$$

$$x = \sum_{i \in \Lambda} x^i\ \lambda^i \tag{3.3}$$

$$\lambda^i \geq 0 \qquad \text{for all} \quad i \in \Lambda. \tag{3.4}$$

Note that this formulation is not sufficient to linearize h by means of the line segments. We also need the additional condition that at most two λ-variables are positive, and if two are positive they must be adjacent. In the literature this condition is called the *Special Ordered Set of Type 2 condition*, briefly SOS 2 condition, and equation (3.1) is then regarded as an SOS Type 2 constraint [BT70, BF76, Bea79].

In the first approximation approach for the nonlinear function h this condition for the λ-variables is modeled implicitly by incorporating it in the branch-and-bound phase [BT70, BF76, Bea79]. In Section 6.2 we describe corresponding branching strategies that guarantee the SOS 2 condition.

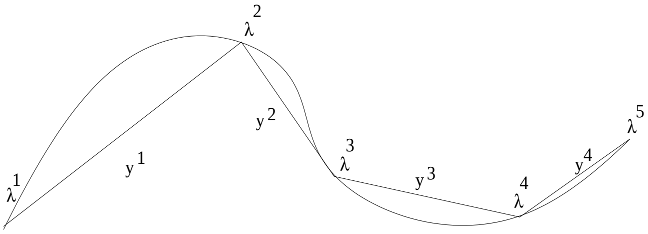

Figure 3.2: Variables for approximation of a function

The second approximation method for a one-dimensional function models this SOS 2 condition explicitly via additional binary variables. This method is known as *convex combination* or *lambda method* [Dan63, NW88, Wil98, LW01]. Again we consider the function h and the decomposition of the interval $D = [a, b]$ with the set of grid points Λ and the corresponding variables λ^i, $i \in \Lambda$. Let Y define the set of indices of the $k-1$ segments that define the partition of the interval. We introduce a binary variable y^j, $j \in Y$, for each segment indicating if this segment is chosen to approximate the nonlinear function, see Figure 3.2. Now we can formulate an MIP which yields an approximated function value $h(x)$.

$$\sum_{i\in\Lambda} \lambda^i = 1 \tag{3.5}$$

$$\sum_{j\in Y} y^j = 1 \tag{3.6}$$

$$\lambda^1 \le y^1 \tag{3.7}$$

$$\lambda^i \le y^{i-1} + y^i \qquad \text{for all} \quad i \in \Lambda \setminus \{1, k\} \tag{3.8}$$

$$\lambda^k \le y^{k-1} \tag{3.9}$$

$$h(x) \approx \sum_{i\in\Lambda} h(x^i)\, \lambda^i \tag{3.10}$$

$$x = \sum_{i\in\Lambda} x^i\, \lambda^i \tag{3.11}$$

$$\lambda^i \ge 0 \qquad \text{for all} \quad i \in \Lambda \tag{3.12}$$

$$y^j \in \{0, 1\} \qquad \text{for all} \quad j \in Y. \tag{3.13}$$

Because of the second constraint exactly one segment of the decomposition is selected to approximate h. Constraints (3.7) to (3.9) ensure that only those λ-variables can be positive which are adjacent to the chosen segment. Hence, these constraints guarantee the SOS 2 condition. In Section 9.2 we investigate the polytope resulting from this modeling of the SOS 2 condition.

The second binary approach is known as *delta* or *incremental method* [Dan63, Wil98, KdFJN04]. It also describes the linear interpolation of the grid points, and its binary formulation is as follows.

Let $l_i := x^{i+1} - x^i$ be the length of the interval $[x^i, x^{i+1}]$. We introduce a continuous variable δ_i for each subinterval $[x^i, x^{i+1}]$ of $[a, b]$, $i \in Y$. Then for $x \in [a, b]$ we obtain the approximation of $h(x)$ by

$$\begin{aligned} x &= a + \sum_{i \in Y} l_i \, \delta_i \\ h(x) &\approx h(a) + \sum_{i \in Y} (h(x^{i+1}) - h(x^i)) \, \delta_i \\ 0 &\leq \delta_i \leq 1 \quad \text{for all} \quad i \in Y, \end{aligned}$$

where the δ-variables have to satisfy the so called *filling condition* [Wil98]:

$$\text{If } \ \delta_i > 0 \ \text{ with } \ 2 \leq i \leq k-1, \ \text{then} \ \ \delta_j = 1 \ \text{ for } \ 1 \leq j < i.$$

This condition assures that if an interval is chosen, then all intervals to its left must also be used completely.
By means of additional binary variables w_i associated with the interval $[x^i, x^{i+1}]$, $i \in Y \setminus \{k-1\}$, we can model this filling condition via

$$\delta_{i+1} \leq w_i \quad \text{for} \quad i \in Y \setminus \{k-1\} \tag{3.14}$$

$$w_i \leq \delta_i \quad \text{for} \quad i \in Y \setminus \{k-1\}. \tag{3.15}$$

Because of constraint (3.15) $w_i = 1$ enforces $\delta_i = 1$. Together with (3.14) $w_{i-1} = 1$ follows. Hence, all intervals to the left are iteratively filled.

[Pad00] shows that the lambda method is computationally inferior to the delta method as its linear programming relaxation always produces worse bounds than the relaxation of the delta method. The author proves that the delta method is *locally ideal*, i.e., that regarding the polytope defined by the LP-relaxation of the delta formulation, the components of extreme points corresponding to binary variables are integer. In contrast, the polytope defined by the lambda formulation has extreme points with fractional components for the binary variables, and it strictly contains the polytope resulting from the delta method. [KdFJN04] prove that the LP-relaxations of the lambda model, of the delta model, and of the SOS Type 2 approach all yield the same optimal objective function value, if the piece-wise linear function is to be minimized. In [CGM03] and [dFJZZ05] approaches for discontinuous piece-wise linear functions are investigated, where the first paper considers MIP models and the second one an SOS approach.

3.2 Approximation of a Higher Dimensional Function

Now we extend these linearizing ideas to higher-dimensional functions. We consider a nonlinear function $h : D \rightarrow \mathbb{R}$ defined on a domain $D \subseteq \mathbb{R}^n$ with an arbitrary $n \in \mathbb{N}$. At first we determine

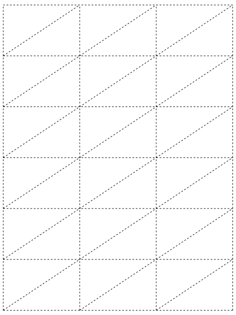

Figure 3.3: Triangulation of a rectangle

a triangulation of the domain D. This results in a set of n-dimensional grid points and a set of simplices. In the following Λ denotes the index set of the grid points and $Y = \{N^1, \dots, N^d\}$, with $N^i \subset \Lambda$, gives the set of simplices, where each of the d simplices is represented by the indices of its vertices. Note that the proposed triangulation needs not be uniform. Moreover, we could consider an arbitrary subdivision (in the one-dimensional case $|N^i| = n + 1$ holds, but in the two-dimensional case for example we could subdivide the domain D in triangles and rectangles). As in the one-dimensional case we introduce a variable λ^i for each grid point $i \in \Lambda$ and calculate the exact function value $h(x^i)$. We linearize h within each simplex by means of the exact function values at its vertices, see again (3.1) to (3.4).

As in the one-dimensional case the λ-variables must fulfill an additional condition, namely that the positive λ-variables belong to the vertices of one simplex. This is modeled via the so called *set condition*, an extension of the SOS 2 condition, which was introduced by [Möl04, MMM06]. The following definition can be found in [MMM06].

Definition 3.2.1 *Consider some subdivision* $Y = \{N^1, \dots, N^d\}$ *with respect to grid points* Λ. *We say that a vector* λ *satisfies the* set condition *with respect to* (3.1) *and the subdivision* Y, *if there exists some index* $r \in \{1, \dots, d\}$ *such that* $\{i \in \Lambda : \lambda^i > 0\} \subseteq N^r$. *We call* (3.1) SOS Type k constraint, *where* $k = \max_i |N^i|$.

Note that in case of a triangulation, (3.1) is an SOS Type k constraint with $k = n + 1$. Let us illustrate this generalization of SOS conditions by means of an example. We consider a two-dimensional function that is defined on a rectangle $[a, b] \times [c, d]$ with $a, b, c, d \in \mathbb{R}$. We determine a triangulation of this rectangle by subdividing the first interval $[a, b]$ into $l - 1$ parts and the second one into $m - 1$ parts with $l, m \geq 2$. The small rectangles that result from this subdivision are halved in triangles. Altogether we obtain lm grid points and $2(l - 1)(m - 1)$ triangles. See Figure 3.3 for an example with $l = 4$ and $m = 7$.

Now we linearize the nonlinear function within each triangle as described above. The set of grid points is given by $\Lambda = \{1, \ldots, kl\}$, and the set Y of simplices is defined by the $2(k-1)(l-1)$ triangles. In this case (3.1) is an SOS Type 3 constraint, since Y consists of triangles. The variables λ^i, $i \in \Lambda$, must fulfill the SOS 3 condition, i.e., at most three λ-variables may be positive and these positive variables must belong to one triangle.

In [Tom81, Möl04, MMM06] suitable branching strategies for SOS conditions in higher dimensions are developed. We describe these strategies in Chapter 6 and improve them by adequate preprocessing methods.

The *generalized lambda method* for the approximation of higher dimensional nonlinear functions models these extended SOS conditions explicitly using additional binary variables [Wil98, LW01]. We again consider the n-dimensional function h, and introduce a binary variable y^j for each simplex $N^j \in Y$ of the triangulation. Let $\tilde{Y} = \{1, \ldots, d\}$ denote the indices of these binary variables. The following mixed integer program yields the function value h approximated by the generalized lambda method.

$$
\begin{aligned}
\sum_{i \in \Lambda} \lambda^i &= 1 \\
\sum_{j \in \tilde{Y}} y^j &= 1 \\
\lambda^i &\leq \sum_{j \in \{l \in \tilde{Y} \mid \, i \in N^l\}} y^j \qquad \text{for all} \quad i \in \Lambda \qquad\qquad (3.16) \\
h(x) &\approx \sum_{i \in \Lambda} h(x^i)\, \lambda^i \\
x &= \sum_{i \in \Lambda} x^i \, \lambda^i \\
\lambda^i &\geq 0 \qquad\qquad\qquad \text{for all} \quad i \in \Lambda \\
y^j &\in \{0, 1\} \qquad\qquad \text{for all} \quad j \in \tilde{Y}.
\end{aligned}
$$

The second constraint guarantees that exactly one simplex of the triangulation is chosen to approximate the nonlinear function. Constraints (3.16) model the SOS Type k condition. In Section 9.3 we study the polytope which results from the modeling of the SOS 3 condition.

Finally, we present the *generalized delta method* which is a bit more complicated [Wil98]. Remember that when considering the delta method for one dimension the filling condition requires an ordering of the simplices. In the one-dimensional case such an ordering is automatically given since the set of simplices Y consists of a sequence of segments. For the delta method in higher dimensions we have to generalize this filling condition, but there does not exist an obvious ordering of the simplices. Therefore, [Wil98] makes the following assumption for the simplices Y of the triangulation which is called the *ordering assumption*.

We assume that the simplices in $Y = \{N^1, \ldots, N^d\}$ are ordered in such a way that $N^{i-1} \cap N^i \neq \emptyset$ holds for $i = 2, \ldots, d$. Furthermore, we require that for each simplex N^i, we can label its $k_i + 1$ vertices as $v_i^0, v_i^1, \ldots, v_i^{k_i}$ such that $v_{i-1}^{k_{i-1}} = v_i^0$ holds for $i = 2, \ldots, d$.

Note that in the one-dimensional case this ordering assumption is automatically fulfilled. In that case all simplices N^i are segments with two vertices $v_i^0 = x^i$ and $v_i^1 = x^{i+1}$, where $v_{i-1}^1 = v_i^0 = x^i$ holds for $i = 2, \ldots, d$. But in general a triangulation need not satisfy the ordering assumption.

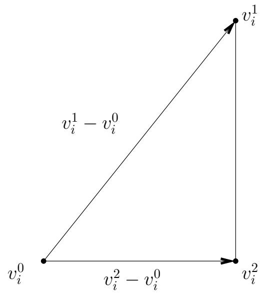

Figure 3.4: Point in simplex given by N^i

In the following we assume that the set of simplices Y of the triangulation of the domain D fulfill the ordering assumption. Choosing a simplex $N^i \in Y$, any point in this simplex can be written as

$$x = v_i^0 + \sum_{j=1}^{k_i} (v_i^j - v_i^0)\, \delta_i^j,$$

where $\sum_{j=1}^{k_i} \delta_i^j \leq 1$ and $\delta_i^j \geq 0$, see Figure 3.4 for an illustration. We see that any point in the ith simplex, given by N^i, can be represented as the point v_i^0 plus a conical combination of the rays $v_i^j - v_i^0$ with $j = 1, \ldots, k_i$. Furthermore, the point v_i^0 can be written as the point v_{i-1}^0 plus the ray $v_{i-1}^{k_{i-1}} - v_{i-1}^0$, since $v_{i-1}^{k_{i-1}} = v_i^0$ holds because of the ordering assumption. Applying this iteratively, v_i^0 is given by

$$v_i^0 = v_1^0 + \sum_{j=1}^{i-1} (v_j^{k_j} - v_j^0) = v_1^0 + \sum_{j=1}^{i-1} (v_{j+1}^0 - v_j^0).$$

Altogether any point in the ith simplex can be presented by v_1^0 plus the vectors from v_j^0 to v_{j+1}^0, $j = 1, \ldots, i-1$, plus a conical combination of the rays $v_i^j - v_i^0$.
Using these ideas we can give the generalized formulation of the delta method [Wil98]. For $x \in D$

we receive an approximated function value for $h(x)$ by

$$\begin{aligned} x &= v_1^0 + \sum_{i=1}^{d} \sum_{j=1}^{k_i} (v_i^j - v_i^0)\, \delta_i^j \\ h(x) &\approx h(v_1^0) + \sum_{i=1}^{d} \sum_{j=1}^{k_i} (h(v_i^j) - h(v_i^0))\, \delta_i^j \\ \sum_{j=1}^{k_i} \delta_i^j &\leq 1 \quad \text{for } i = 1, \ldots, d \\ \delta_i^j &\geq 0 \quad \text{for } i = 1, \ldots, d;\ j = 1, \ldots, k_i, \end{aligned}$$

where the δ-variables have to satisfy the so called *generalized filling condition* [Wil98]:

$$\text{For } i = 1, \ldots, d,\ j = 1, \ldots, k_i,\ \ \delta_i^j \text{ can be positive only if } \delta_{i-1}^{k_{i-1}} = 1.$$

Iteratively we can conclude that a variable δ_i^j can only be positive, if all variables $\delta_{l-1}^{k_{l-1}}$ equal one for $l = 2, \ldots, i$. This means that a point in the ith simplex given by N^i can only be considered if all simplices N^l in the ordering before are used up completely. Notice that in dimension one the generalized filling condition reduces to the filling condition for the one-dimensional delta method. To model the generalized filling condition we have to introduce binary variables w_i associated with simplex N^i for $i = 1, \ldots, d-1$. The following constraints enforce this condition

$$\begin{aligned} \sum_{j=1}^{k_i} \delta_{i+1}^j &\leq w_i \quad \text{for} \quad i = 1, \ldots, d-1 \\ w_i &\leq \delta_i^{k_i} \quad \text{for} \quad i = 1, \ldots d-1. \end{aligned}$$

If δ_i^j is positive, the first constraint enforces the binary variable w_{i-1} to one. With the second constraint $\delta_{i-1}^{k_{i-1}} = 1$ follows. Applying this repeatedly, we see that all simplices before N^i are used up.

[Wil98] shows the integrality of the polytope resulting from the LP-relaxation of the generalized delta formulation. Further on, the corresponding polytope of the generalized lambda method has extreme points with fractional components for the binary variables. Hence, the results of [Pad00] also hold for the n-dimensional case. Moreover, [Wil98] presents computational results on randomly generated instances where the sum of two-dimensional functions is to be maximized. These results show the superiority of the delta over the lambda method.

In [MMM06] the generalized delta and lambda method as well as the extended SOS approach are compared in context of an MIP approach for the stationary case of gas network optimization. Computational results for that application also show that the delta method provides better results than the lambda method. But the SOS approach yields by far the best running times, and therefore, no binary approach is competitive to it.

Chapter 4

Physical Characteristics of a Gas Network

In this chapter we gain insight into the physical and technical background of gas network optimization. The problem of TTO contains two main topics including physical characteristics. On the one hand there is the system of equations describing the transient processes in a pipe. On the other hand compressors operate subject to complex technical properties.
In the following sections we will discuss these themes in more detail and introduce the most important variables. A complete list of all variables - time and spatial dependent variables and constants - is given in the last section of this chapter (see Tables 4.1 and 4.2). The physical and technical descriptions are based on [Sek00, Sek01, RS01, Sek03].

4.1 Gas Dynamics in Pipes

The gas transport in a pipe is described by a nonlinear system of partial differential equations as well as the thermodynamic state equation of gas. For each pipe of the network, we obtain four equations. At first we consider these equations, i.e., the continuity equation, the momentum equation, the energy equation, and the state equation. After that we transform them, since we want to use the variables gas pressure p and gas (volume) flow q in our model. At last we present suitable discretizations (in space and time) for the simplified equations, which we can integrate in our model.

4.1.1 The Continuity Equation

At first we consider the continuity equation [Sek00]

$$\frac{\partial M}{\partial x} + A\,\frac{\partial \rho}{\partial t} = 0, \tag{4.1}$$

where M is the mass flow of gas, ρ is the gas density, and A is the area of the cross-section of the pipe. It states that the rate of change of the gas density in time corresponds to the mass flow out of or into the pipe across its boundaries. For example, if the amount of gas entering the pipe is bigger than the amount of gas leaving the pipe, then the density of the gas inside the pipe grows. If we standardize the mass flow by means of the norm density

$$M = \rho_0\, q$$

we receive the (norm) gas flow q, a technical variable commonly used in gas transport [Sek00]. Therefore q is the gas flow variable of our model.

4.1.2 The Momentum Equation

The momentum equation describes the sum of all forces on the gas particles. For cylindrical pipes it has the following form [Sek00, Sek03]

$$\frac{\partial p}{\partial x} + g\rho\frac{\partial h}{\partial x} + \frac{\lambda |v| v}{2D}\rho + \frac{1}{A}\frac{\partial M}{\partial t} + \frac{\partial(\rho v^2)}{\partial x} = 0. \tag{4.2}$$

Here the important variables p and v denote the pressure and the flow velocity of gas, respectively. Moreover, g is the acceleration constant due to gravity, $\frac{\partial h}{\partial x}$ is the slope of the pipe, λ is the pipe friction value, and D the diameter of the pipe.

Let us consider the terms of this equation. Obviously the first term is the pressure gradient. Thereafter, we find the force of gravity which is influenced by the slope of the pipe. The third term gives the friction force with the pipe wall. The friction factor λ in this term depends on the diameter and the roughness of the pipe. The following term signifies the change of flow rate in time. Finally, we have the so called impact pressure. The most important factor which influences the pressure loss is the friction force with the pipe wall.

4.1.3 The Energy Equation

The last partial differential equation is the energy equation [Sek00]

$$W(\rho A\,dx) = \frac{\partial}{\partial t}\left[(\rho A\,dx)\,(c_v T + \frac{v^2}{2} + g\,dh)\right] + \frac{\partial}{\partial x}\left[(\rho v A\,dx)\,(c_v T + \frac{p}{\rho} + \frac{v^2}{2} + g\,dh)\right].$$

Here, W denominates the heat addition (per mass flow and time) from the soil to the gas, c_v the specific heat (at constant gas volume), and T the temperature of the gas. Besides the inner energy of the gas, the energy equation deals with the heat exchange with the soil which is a very slow process.

The first two equations (continuity and momentum equation) as well as the state equation of gas are of prevailing significance for the simulation of most transient effects in pipeline networks. Since the German pipes are at least one meter beneath the ground with nearly constant temperature, in general (transient) simulation models do not take the energy equation into account (see for example [Sek00, Záw]). So we neglect the energy equation in this context and we assume the temperature T to be constant.

4.1.4 The State Equation of Gas

Finally, there is the state equation [Sek00]

$$\rho = \frac{\rho_0 z_0 T_0}{p_0} \frac{p}{zT},$$

where the index '0' stands for values under norm condition, and z is the compressibility factor (so called z-factor) characterizing the non-ideal behavior of gas. Real gas deviates from the ideal gas because of the interaction of molecules (clearly for ideal gas $z \equiv 1$ holds). This z-factor depends on the chemical composition of the gas as well as on pressure and temperature.

Since no exact theoretical determination of the z-factor exists, several empirical approximations were developed. In our model we use the following formula from the American Gas Association (AGA)

$$z(p, T) = 1 + 0.257 p_r - 0.533 \frac{p_r}{T_r},$$

where $p_r = \frac{p}{p_c}$ and $T_r = \frac{T}{T_c}$ are the relative pressure and temperature [Sek00] with pseudo-critical pressure p_c and pseudo-critical temperature T_c. It yields a good approximation for pressure up to 70 bar which is a realistic assumption in our gas network.

4.1.5 Simplification of the Equations

We want to bring the equations in the form needed for our model, where the variables used in practice are introduced. As mentioned above, we omit the energy equation and assume a constant temperature T. In the following we set the state equation of gas in the remaining equations and thus obtain a system of two partial differential equations.

At first the state equation and the approximated z-factor can be simplified because of the constant temperature

$$\rho = \frac{\rho_0 z_0 T_0}{p_0 T} \frac{p}{z} \quad \text{and} \quad z = z(p) = 1 + \alpha p.$$

Notice that the pseudo-critical pressure p_c and temperature T_c are also constants that $\alpha < 0$ and hence $0 < z < 1$.
We continue with the continuity equation (4.1). By introducing the mass flow $M = \rho_0\, q$ and the state equation we receive

$$\frac{\partial q}{\partial x} + A\, \frac{z_0 T_0}{p_0 T} \frac{\partial}{\partial t} \left(\frac{p}{z(p)} \right) = 0. \tag{4.3}$$

The momentum equation (4.2) requires a bit more work. At first we introduce $M = \rho_0\, q$ and the gas velocity $v = \frac{\rho_0}{A} \frac{q}{\rho}$ and obtain (see [Sek01])

$$\frac{\partial p}{\partial x} + g\rho \frac{\partial h}{\partial x} + \frac{\lambda}{2D} \frac{\rho_0^2}{A^2} \frac{|q|q}{\rho} + \frac{\rho_0}{A} \frac{\partial q}{\partial t} + \frac{\rho_0^2}{A^2} \frac{\partial}{\partial x} \left(\frac{q^2}{\rho} \right) = 0.$$

Insertion of the state equation yields

$$\frac{\partial p}{\partial x} + g \frac{\rho_0 z_0 T_0}{p_0 T} \frac{\partial h}{\partial x} \frac{p}{z(p)} + \frac{\lambda}{2D} \frac{\rho_0 p_0 T}{A^2 z_0 T_0} \frac{|q|q\, z(p)}{p} + \frac{\rho_0}{A} \frac{\partial q}{\partial t} + \frac{\rho_0 p_0 T}{A^2 z_0 T_0} \frac{\partial}{\partial x} \left(\frac{q^2 z(p)}{p} \right) = 0. \tag{4.4}$$

Until now all pipes in the data which we used for our calculations are horizontal, hence the second term of the equation vanishes.
The pipe friction value λ is an important factor of the friction term. In principle it depends on the diameter D and the roughness k of the pipe and on the Reynolds' number

$$R_e = \frac{4}{\pi\, \eta} \frac{M}{D}$$

indicating whether a flow is laminar or turbulent. Generally the gas flow we are considering is turbulent. Unfortunately the friction factor λ is given by an implicit formula from Prandtl-Colebrook

$$\frac{1}{\sqrt{\lambda}} = -2 \log_{10} \left(\frac{2.51}{R_e \sqrt{\lambda}} + \frac{k}{3.71\, D} \right).$$

But choosing an iterative method it can be calculated within a few iterations.

Now the two partial differential equations have the desired form in both the time and spatial dependent variables p and q.

4.1.6 Discretization of the Equations

We conclude this section about gas dynamics in pipes by presenting discretizations for the continuity equation (4.3) and the momentum equation (4.4). Note that as we want to include these discretizations into our mixed integer model, they must be as simple as possible. On the one hand, we need equations which describe the transient processes in the pipes. On the other hand, we cannot be as exact as necessary for simulation algorithms, since we want to optimize large gas networks, see [ES03, ES05].

For the space discretization, we use a simple grid consisting of the beginning and the end of the pipe, signified by ‘in’ and ‘out’ (Note that if we have a very long pipe, we could subdivide it and model it by several short pipes.). For the discretization in time, the planning horizon is divided equidistantly to receive a time grid. In our case we consider hourly time steps. Following [ES03, ES05], the two partial differential equations are discretized with implicit Euler schemes in space and time. Let us begin with the continuity equation (4.3). Discretizing this equation in space and time yields

$$\frac{q_{out}(t) - q_{in}(t)}{L} + A\frac{z_0 T_0}{p_0 T} \frac{\frac{p_{out}(t)}{z(p_{out}(t))} - \frac{p_{out}(t-\Delta t)}{z(p_{out}(t-\Delta t))}}{\Delta t} = 0, \tag{4.5}$$

where $q_{in}(t)$ and $q_{out}(t)$ denotes the flow at the beginning and the end of the pipe at time t, and L is the length of the pipe. $p_{out}(t)$ describes the pressure at the end node of the pipe at time t and $p_{out}(t - \Delta t)$ at the previous time step. Thus, Δt stands for the length of the subdivision of the planning horizon, which, as already mentioned, in our case is an hour. Note that the flow direction in a pipe is specified and no back-flow is allowed.

Now we come to the momentum equation (4.4). We again discretize in space and time and obtain

$$\begin{aligned}\frac{p_{out}(t) - p_{in}(t)}{L} &+ g\frac{\rho_0 z_0 T_0}{p_0 T}\frac{\partial h}{\partial x}\frac{p_{out}(t)}{z(p_{out}(t))} + \frac{\lambda}{2D}\frac{\rho_0 p_0 T}{A^2 z_0 T_0}\frac{q_{out}^2(t) z(p_{out}(t))}{p_{out}(t)} \\ &+ \frac{\rho_0}{A}\frac{q_{out}(t) - q_{out}(t-\Delta t)}{\Delta t} + \frac{\rho_0 p_0 T}{A^2 z_0 T_0}\frac{\frac{q_{out}^2(t) z(p_{out}(t))}{p_{out}(t)} - \frac{q_{in}^2(t) z(p_{in}(t))}{p_{in}(t)}}{L} = 0, \end{aligned} \tag{4.6}$$

where we follow the same notations as above. Hence, $p_{in}(t)$ denotes the pressure at the beginning of the pipe at time t and $q_{out}(t - \Delta t)$ signifies the flow at the end of the pipe at the previous time step. As we assume no back-flow in a pipe, we can neglect the absolute value of the flow variable in the third term of the equation.

Note that these are coarse discretizations for the partial differential equations. In [Kol] these discretizations are evaluated via numerical simulation. The computational results therein show that these simple discretizations are sufficient for our optimization model. Notice that we cannot integrate these discretizations in our model directly. Recall that we want to formulate the gas network optimization problem as a mixed integer program, thus we can only accept linear constraints. If we look at equations (4.5) and (4.6), we see that they contain several nonlinear terms. In Chapter 5 we consider how to handle these nonlinearities. We develop appropriate approximations to receive linear conditions describing the transient processes in pipes.

4.2 Compressors

In this section we concentrate on the technical and physical scenario of a compressor. We model a special kind of compressor, the so called turbocompressor, driven by a gas turbine. It is the most commonly used machine in the network of the E.ON Ruhrgas AG.
At first, we describe the technical function in detail. As we will see, the operation of a compressor is quite complex and involves a lot of technical and physical variables. To have a clear structure we neglect mechanical losses. Moreover most of the variables are given by empirical data tables. Therefore, simplifications are necessary and we obtain an idealized compressor which we include in our model, see [Sek01, RS01].

Each compressor has a favorable working point, a so called design point, and it can operate in the neighborhood of this point. This neighborhood is often described by a characteristic diagram. The characteristic diagram of a turbocompressor comprises the parameters flowrate Q, head H_{ad}, revolutions (or speed) n, and adiabatic efficiency η_{ad}. The flowrate is given by means of the gas flow q with

$$Q = 3.6^{-1} \frac{p_0 T_{in} z_{in}}{p_{in} T_0}\, q,$$

where the index ‘in’ signifies the input of the compressor. The value of the head

$$H_{ad} = \frac{\kappa}{\kappa - 1}\, \frac{z_{in}\, R\, T_{in}}{g\, m} \left[\left(\frac{p_{out}}{p_{in}} \right)^{\frac{\kappa-1}{\kappa}} - 1 \right] \tag{4.7}$$

can be interpreted as the (fictive) height, on which the gas have to be pumped, to attain the same power. Here κ is the isentropic exponent, R is the universal gas constant, and m is the molecular weight of gas. The index ‘out’ stands for the output of the compressor. Naturally, H_{ad} depends on the pressures at the input and output of the compressor, more precisely on the pressure quotient.
The adiabatic efficiency η_{ad} is a quotient

$$\eta_{ad} = \frac{N_{gas}}{N_{coupl}} \tag{4.8}$$

of the power

$$N_{gas} = \frac{g}{3600} \rho_0\, q\, H_{ad}, \tag{4.9}$$

which is necessary to compress the gas, and the power N_{coupl} which has to be generated by the drive unit (here the gas turbine), see below.
Each (permissible) working point of a turbocompressor is specified by a measured (or extrapolated) quadruple of this four parameters. The empirical data (measured points) can be found in tables of

the following shape

$$\begin{array}{ccccccccc}
 & & n_1 & & n_2 & \dots & \dots & & n_j \\
\eta_1 & (H_{ad1,1}, & Q_{1,1}) & (H_{ad1,2}, & Q_{1,2}) & \dots & \dots & (H_{ad1,j}, & Q_{1,j}) \\
\eta_2 & (H_{ad2,1}, & Q_{2,1}) & (H_{ad2,2}, & Q_{2,2}) & \dots & \dots & (H_{ad2,j}, & Q_{2,j}) \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\eta_i & (H_{adi,1}, & Q_{i,1}) & (H_{adi,2}, & Q_{i,2}) & \dots & \dots & (H_{adi,j}, & Q_{i,j}).
\end{array}$$

Hence, for each pair of efficiency η_{ad} and revolutions n, we obtain different values for the head and the flowrate. Using the formulas of H_{ad} and Q we receive the connection to the variables pressure p and gas flow q in our model.

The permissible working range is limited by four curves: the maximum and minimum revolutions (or speed), the surge limit, and the choke line. While the compressor is destroyed by operation beyond max./min. speed or surge limit, the choke line can be exceeded. But such an exceeding should be avoided because of very poor efficiency.

Now we consider the operation mode of the gas turbine which has to generate the shaft power N_{coupl} at the coupler. The gas turbine also possesses a characteristic diagram defined by (amongst others) the parameters shaft speed n_{driv}, power at the coupler N_{coupl}, drive power N_{driv}, air flow, and compressor inlet temperature T. The manufacturer of the turbines delivers several diagrams in the coordinates shaft speed n_{driv}, power N_{coupl}, and specific fuel consumption b (sometimes also air flow) for varying compressor inlet temperatures. These diagrams are of the form

	N_1	N_2	$\dots$	N_j
	b_1	b_2	$\dots$	b_j
	n_1	n_2	$\dots$	n_j
T_1	$N_{1,1}$	$N_{1,2}$	$\dots$	$N_{1,j}$
T_2	$N_{2,1}$	$N_{2,2}$	$\dots$	$N_{2,j}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
T_i	$N_{i,1}$	$N_{i,2}$	$\dots$	$N_{i,j}$.

Here for (discrete) values of the shaft power N_{coupl}, the (specific) fuel consumption and the shaft speed can be found. The dependence on the inlet temperature is given by the second part of the diagram, where the interrelationship of the three parameters inlet temperature, shaft power, and speed are shown.

The specific fuel consumption is the ratio of input to output power

$$b = \frac{N_{driv}}{N_{coupl}} = \frac{1}{\eta_{driv}}$$

and equals the reciprocal of the turbines efficiency.

We are interested in the hourly fuel consumption f which can be calculated by the formula

$$f = \frac{N_{driv}}{1000 H_n} = \frac{N_{coupl}\, b}{1000 H_n} \tag{4.10}$$

where the constant H_n is the net calorific value of the gas.

As mentioned above, we use an idealized form of the turbocompressor which we include in our model. Based on empirical data, approximated mean values for the adiabatic efficiency η_{ad} and the specific fuel consumption b can be calculated. Furthermore, we incorporate fixed limiting values N_{min} and N_{max} for the shaft power. All other parameters and limits are not taken into account.
So we obtain constant values for η_{ad} and b as well as constant bounds for N_{coupl} in the equations (4.7) till (4.10). By combination of these formulars we can reduce the number of equations. We receive equation (4.10) and the nonlinear equation

$$f = \frac{b\rho_0 z_{in} R T_{in}}{36 \cdot 10^5 m H_n \eta_{ad}} \frac{\kappa}{\kappa - 1} \left[\left(\frac{p_{out}}{p_{in}} \right)^{\frac{\kappa-1}{\kappa}} - 1 \right] q \qquad (4.11)$$

for the fuel gas consumption including the model variables pressures at the beginning p_{in} and at the end p_{out} of the compressor as well as gas flow q through it. We include this nonlinear function $f = f(p_{in}, p_{out}, q)$ in our MIP using an adequate linearization, see Chapter 5.
Finally let us remark that this nonlinear function $f(p_{in}, p_{out}, q)$ is neither convex nor concave. For constant flow q and ingoing pressure p_{in} the function is concave. Assuming constant flow and outgoing pressure it is convex. This is an important fact, since fuel consumption of the compressors is an essential part of the objective function. Figure 4.1 shows this function for constant flow q.

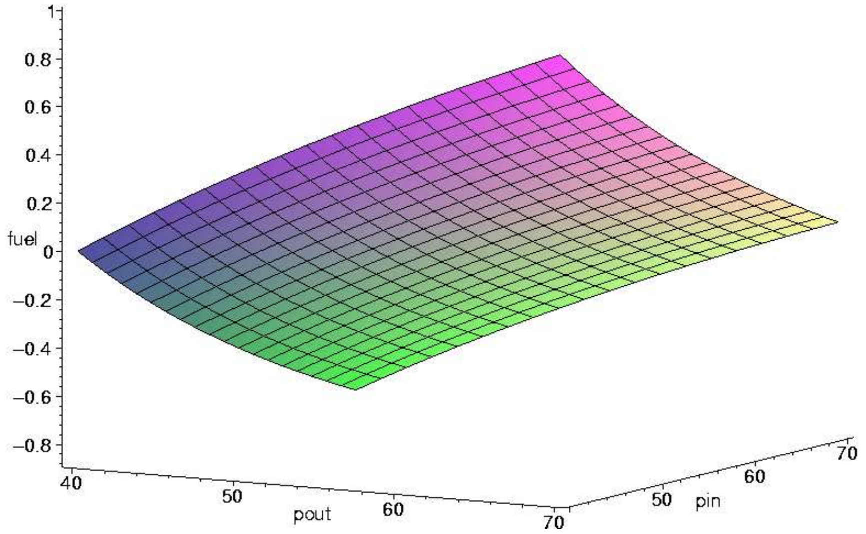

Figure 4.1: Function for fuel gas consumption with constant flow

4.3 Variables and Constants

We conclude this chapter with a list of all dependent variables and constants appearing in the description of the physical characteristics in a gas network. At first we table the dependent variables. There are variables dependent on time and space as well as variables dependent just on space or time. The letter x (unit m) denotes the space and t (unit s) indicates the time. Note that because of simplifications some variables become independent (e.g. T, η_{ad}).

Table 4.1: Variables in the physical description

$q = q(x,t)$	gas flow	$[m^3 s^{-1}]$
$p = p(x,t)$	pressure of gas	$[Pa]$
$\rho = \rho(x,t)$	gas density	$[kg\ m^{-3}]$
$v = v(x,t)$	flow velocity of gas	$[m\ s^{-1}]$
$M = M(x,t)$	mass flow of gas	$[kg\ s^{-1}]$
$\lambda = \lambda(x,t)$	friction coefficient of pipe	$[1]$
$T = T(x,t)$	gas temperature	$[K]$
$z = z(x,t)$	compressibility factor (z-factor)	$[1]$
$p_r = p_r(x,t)$	relative pressure of gas	$[1]$
$T_r = T_r(x,t)$	relative temperature of gas	$[1]$
$R_e = R_e(x,t)$	Reynolds number	$[1]$
$A = A(x)$	cross section of pipe	$[m^2]$
$h = h(x)$	height of pipe	$[m]$
$W = W(t)$	heat addition to gas	$[kJ\ kg^{-1} s^{-1}]$
$Q = Q(t)$	flowrate	$[m^3 s^{-1}]$
$H_{ad} = H_{ad}(t)$	adiabatic head	$[m]$
$n = n(t)$	revolutions/speed of compressor	$[min^{-1}]$
$\eta_{ad} = \eta_{ad}(t)$	adiabatic efficiency of compressor	$[1]$
$T_{in} = T_{in}(t)$	gas temperature at input of compressor	$[K]$
$z_{in} = z_{in}(t)$	z-factor at input of compressor	$[1]$
$p_{in} = p_{in}(t)$	gas pressure at input of compressor	$[bar]$
$p_{out} = p_{out}(t)$	gas pressure at output of compressor	$[bar]$
$N_{gas} = N_{gas}(t)$	(theoretical) power of compressor	$[kW]$
$N_{coupl} = N_{coupl}(t)$	shaft power	$[kW]$
$N_{driv} = N_{driv}(t)$	drive power of turbine	$[kW]$
$n_{driv} = n_{driv}(t)$	shaft speed of turbine	$[min^{-1}]$
$b = b(t)$	specific fuel consumption	$[1]$
$\eta_{driv} = \eta_{driv}(t)$	adiabatic efficiency of turbine	$[1]$
$f = f(t)$	fuel gas consumption of compressor	$[m^3 h^{-1}]$

In the following, we compile the list of general physical constants as well as gas and pipe constants.

Table 4.2: Physical, gas, and pipe constants

g	acceleration of gravity	$[m\ s^{-2}]$
D	diameter of pipe	$[m]$
L	length of pipe	$[m]$
c_v	specific heat	$[kJ\ kg^{-1}K^{-1}]$
ρ_0	norm density	$[kg\ m^{-3}]$
z_0	norm compressibility factor	$[1]$
T_0	norm temperature	$[K]$
p_0	norm pressure	$[bar]$
p_c	pseudo-critical pressure	$[bar]$
T_c	pseudo-critical temperature	$[K]$
η	dynamic viscosity of gas	$[kg\ m^{-1}s^{-1}]$
k	roughness of pipe	$[m]$
κ	isentropic exponent	$[1]$
R	universal gas constant	$[kJ\ kmol^{-1}K^{-1}]$
m	molecular weight of gas	$[kg\ kmol^{-1}]$
H_n	net calorific value of gas	$[kW\ m^{-3}]$

Finally, we give the physical units used in the description and the units of the variables assumed in our implementation.

Table 4.3: Units of physical variables and of variables used in program

	physical unit	unit used in program
pressure variable p	$[Pa]$	$[bar]$
flow variable q	$[m^3/s]$	$[1000m^3/h]$
length L of pipe	$[m]$	$[km]$

Chapter 5

The Model

The problem of TTO is to optimize the gas flow in the network such that the demands of the consumers are fulfilled and the fuel gas consumption and switching costs of compressors are minimized.

In this chapter we model this problem as a mixed integer program. We introduce the elements of a gas network and give a mathematical formulation of their properties. Besides trivial inequalities like bounds for the diverse variables, we put emphasis on combinatorial constraints including so called switching variables on the one hand. On the other hand, we have to model nonlinear components resulting from fuel gas consumption of compressors and gas dynamics in pipes. These nonlinearities are approximated by piece-wise linear functions. As these approximations play a decisive role, the basic ideas can be found in Chapter 3.

In the following we present the graph structure of a gas network and we introduce the variables of the model. Then we concentrate on time-independent constraints, i.e., constraints that also appear in the stationary case, describing the components of a gas network. After this we consider the discretized equations resulting from gas dynamics in pipes and show our linearizing ideas. Besides, further time-dependent conditions are needed concerning switching modes of compressors and states of the gas network. Finally we specify the objective function.

5.1 Graph Structure and Variables

A gas network basically consists of pipes, compressors and valves. Moreover there are gas delivering points and consumers. A gas network can be modeled via a directed graph $G = (V, E)$ as follows. The set of edges, in this contest also called segments, can be divided into the set E_P of pipes, the set E_C of compressors, and the set E_V of valves. Further on, we have special kinds of valves $E_R \subset E_V$, so called control valves, having the additional property to reduce gas pressure.

Connections, a subset of the pipes $E_A \subset E_P$, can be viewed as short pipes without pressure loss. The set of nodes V consists of the set of sources (gas suppliers) $S \subset V$, the set of sinks (consumers) $U \subset V$, and innodes (intersection of points of segments). We assume the graph to be directed since we do not allow back-flow in pipes.

Now let us come to the variables of the model. The most important variables are gas flow and pressure variables, fuel gas of compressors and decision variables for the controllable elements.
For each segment except the pipes we obtain one flow variable describing the gas volume flow in it. Whereas for each pipe there are two flow variables signifying the flow at the beginning and at the end of a pipe. In our model pressure is a nodal variable. Then there is a fuel gas variable for each compressor reflecting the fuel gas consumption of the machine. Finally, the decision variables indicate if a compressor or a valve is on or off.
Let $T \in \mathbb{N}$ denote the number of time periods (in our case hours), into which the planning horizon is divided. All variables receive a time index t for each time step $t \in \mathbb{T} := \{1, \ldots, T\}$. In the following, we specify all variables of the model where we begin with continuous variables and end with the binary ones.

Table 5.1: Variables of the model

q_e^t	gas flow in compressor, valve or connection	$e \in E_C \cup E_V \cup E_A$
$q_{e,v}^t$	gas flow at the beginning of the pipe	$e = vw \in E_P \setminus E_A$
$q_{e,w}^t$	gas flow at the end of the pipe	$e = vw \in E_P \setminus E_A$
p_v^t	gas pressure in node	$v \in V$
q_v^t	gas delivered from source	$v \in S$
q_v^t	gas consumed at sink	$v \in U$
f_e^t	fuel gas of compressor	$e \in E_C$
N_e^t	power of compressor	$e \in E_C$
$p_{e,v}^{t,h}, p_{e,w}^{t,h}$	auxiliary variables for compressor	$e = vw \in E_C$
$s_e^t \in \{0,1\}$	switching variable of compressor or valve	$e \in E_C \cup E_V$
$s_{e,up}^t \in \{0,1\}$	start-up variable of compressor	$e \in E_C$
$s_{e,down}^t \in \{0,1\}$	shut-down variable of compressor	$e \in E_C$

Note that all continuous variables are nonnegative and due to technical constraints they all have constant upper and lower bounds.

Concluding this section, we list constants of the model. They are all positive.

Table 5.2: Constants of the model

q_e^{min}	minimal gas flow in pipe, compressor or valve	$e \in E_P \cup E_C \cup E_V$
q_e^{max}	maximal gas flow in pipe, compressor or valve	$e \in E_P \cup E_C \cup E_V$
p_v^{min}	minimal gas pressure in node	$v \in V$
p_v^{max}	maximal gas pressure in node	$v \in V$
$q_{v,t}^{min}$	minimal gas flow delivered from source	$v \in S,\ t \in \mathbb{T}$
$q_{v,t}^{max}$	maximal gas flow delivered from source	$v \in S,\ t \in \mathbb{T}$
$q_{v,t}^{min}$	minimal gas flow consumed at sink	$v \in U,\ t \in \mathbb{T}$
$q_{v,t}^{max}$	maximal gas flow consumed at sink	$v \in U,\ t \in \mathbb{T}$
$M_e, \tilde{M}_e$	constants for valve	$e \in E_V$
dp_e^{max}	maximal pressure difference for bypass valve	$e \in E_V \setminus E_R$
dp_e^{max}	maximal pressure reduction of control valve	$e \in E_R$
dp_e^{min}	minimal pressure reduction of control valve	$e \in E_R$
N_e^{max}	maximal power of compressor	$e \in E_C$
N_e^{min}	minimal power of compressor	$e \in E_C$
f_e^{max}	maximal fuel gas of compressor	$e \in E_C$
$C_{e,up}^t$	start-up cost for compressor in time step $t \in \mathbb{T} \setminus \{1\}$	$e \in E_C$
$C_{e,down}^t$	shut-down cost for compressor in time step $t \in \mathbb{T} \setminus \{1\}$	$e \in E_C$

5.2 The Components of a Gas Network

In this section we consider the principal components of a gas transmission system. We describe the properties of the particular elements and show how we integrate them in our model, whereby we restrict to the time-independent inequalities that is the constraints that occur also in the stationary case of gas network optimization.

In principal there are three kinds of constraints: combinatorial constraints including switching variables, linearization constraints for the fuel gas consumption, and simply bounds for the variables. We only give a brief overview since the stationary case was discussed in detail by [Möl04]. Now we concentrate on function and modeling of the specific network components. Naturally, the following conditions must be considered for each time step $t \in \mathbb{T}$ of the planning horizon.

5.2.1 Properties of a Pipe

Obviously, the most common segment of a transmission system is a *pipe*. For example the length of the pipes in the network driven by the German E.ON Ruhrgas AG is about 10.000 kilometers [RA].

For the stationary description of a pipe only few conditions are needed. The most important pipe constraints arise because of the physical characteristics continuity and momentum which have

effect on gas flow and pressure. We will focus our attention on these time-dependent properties in Section 5.3.
So we just have to take flow bounds into account. Hence we receive

$$q_e^{min} \leq q_{e,v}^t, q_{e,w}^t \leq q_e^{max}, \tag{5.1}$$

where $e = vw \in E_P \setminus E_A$.

5.2.2 Properties of a Connection

For a *connection*, gas dynamics can be neglected. Thus it is a transport element with constant flow and without pressure loss in it.
For a connection $e = vw \in E_A$, this property is modeled by the simple equation

$$p_v^t = p_w^t. \tag{5.2}$$

Further on, we just have to pay attention to bounds of the gas flow

$$q_e^{min} \leq q_e^t \leq q_e^{max}. \tag{5.3}$$

5.2.3 Properties of a Valve

A *valve* is a controllable element of the gas network. It can be open or closed so it can be controlled whether there is gas flow or not. This effect is modeled via a switching variable $s_e^t \in \{0, 1\}$, $e \in E_V \setminus E_R$. Beyond we assume that there is no pressure loss regarding an open valve (except control valves, see next subsection).
A valve can be placed anywhere in a transmission system. Mainly it can be found in connection with a compressor. If a compressor is shut down the gas needs an alternative way to flow. For this reason a valve is built-in parallel to a compressor, a so called *bypass valve*. Below we will see that there is a difference between the constraints of an ordinary and a bypass valve. The technical symbol of a valve is shown in Figure 5.1.

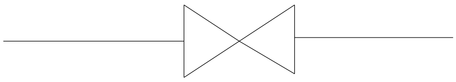

Figure 5.1: Symbol of a valve

Both types of valves have bounds for the gas flow q_e^t, $e \in E_V \setminus E_R$. These flow bounds are affected by the switching variable s_e^t since there cannot be any flow if a valve is closed, so we obtain

$$q_e^{min} s_e^t \leq q_e^t \leq q_e^{max} s_e^t. \tag{5.4}$$

Now we have to distinguish between valves and bypass valves.
For an ordinary valve we want constraints which assure that the pressure in the incident nodes is equal if the valve is open and that the corresponding pressures can be arbitrary if the valve is closed. So for $e = vw \in E_V \setminus E_R$ we receive two inequalities

$$M_e\, s_e^t - p_v^t + p_w^t \leq M_e \quad \text{and} \quad \tilde{M}_e\, s_e^t + p_v^t - p_w^t \leq \tilde{M}_e, \tag{5.5}$$

whereby M_e and $\tilde{M}_e$ are appropriate constants depending on pressure bounds of the nodes.
Considering a bypass valve we have similar constraints

$$dp_e^{max} s_e^t - p_v^t + p_w^t \leq dp_e^{max} \quad \text{and} \quad p_v^t - p_w^t \leq 0, \tag{5.6}$$

whereas dp_e^{max} signifies the maximal pressure difference between beginning and end node. If the bypass valve is open these inequalities ensure equality of pressure in the nodes. If the valve is closed, the compressor must work. If so the second inequality yields a pressure increase which is limited by the first inequality because of technical conditions.
It is clear that further constraints modeling the combination of a compressor and its bypass valve are required. They are outlined in the subsection of compressor properties.

5.2.4 Properties of a Control Valve

A *control valve* is a special kind of valve. It has the additional property that it reduces the pressure. Such a regulator is often installed before a consumer because the distribution company has to meet stringent pressure conditions. Notice that a control valve is never used as a bypass for a compressor. Figure 5.2 shows its technical symbol.

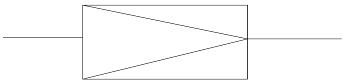

Figure 5.2: Symbol of a control valve

Like for a valve the gas flow in a control valve $e \in E_R$ has operating dependent bounds

$$q_e^{min} s_e^t \leq q_e^t \leq q_e^{max} s_e^t. \tag{5.7}$$

In contrast to a valve there is device-controlled pressure drop instead of pressure equality in the adjacent nodes of an opened control valve. So we make minor modifications in inequalities (5.5) and obtain

$$(M_e + dp_e^{min}) s_e^t - p_v^t + p_w^t \leq M_e \quad \text{and} \quad (\tilde{M}_e - dp_e^{max}) s_e^t + p_v^t - p_w^t \leq \tilde{M}_e \tag{5.8}$$

for $e = vw \in E_R$ with technical bounds dp_e^{min}, dp_e^{max}.

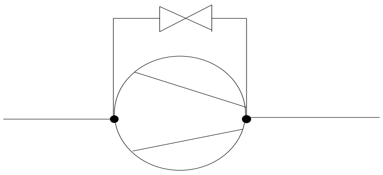

Figure 5.3: A compressor and its bypass valve

5.2.5 Properties of a Compressor

Now we discuss the model of a compressor. A *compressor* compensates for the pressure loss in pipes. It is driven by a gas turbine therefore some fraction of gas flowing through it is lost during compression process. For the precise physical background we refer to Section 4.2. Remember that fuel gas consumption is one of the main cost in the objective function.

Clearly, compressors operate subject to a fairly complex system of constraints. For this reason we divide the constraints into two parts. First we regard some basic inequalities. Thereafter we concentrate on the nonlinear function describing the fuel gas consumption of a compressor.

Just as a valve, a compressor $e \in E_C$ has a switching variables s_e^t indicating if a compressor is on or off. At first there are bounds for the gas flow through a compressor

$$q_e^{min} s_e^t \leq q_e^t \leq q_e^{max} s_e^t. \tag{5.9}$$

After that we consider a compressor in connection with its bypass valve. Figure 5.3 is a sketch of this situation. Let $s_{v(e)}^t$ be the switching variable of the respective bypass valve $v(e) \in E_V \setminus E_R$. The next constraint is of combinatorial nature

$$s_e^t + s_{v(e)}^t = 1. \tag{5.10}$$

It states that either the compressor is on or the valve is open.

At last we focus on basic inequalities including fuel gas f_e^t and power N_e^t of a compressor. These two variables are coupled since the fuel gas consumption of a compressor depends on its power. So we receive a coupling constraint

$$N_e^t = \frac{1000 H_n}{b_e} f_e^t, \tag{5.11}$$

whereby the constant H_n stands for (net) calorific value of gas and the constant b_e for specific fuel consumption of the gas turbine (see also equation (4.10)).

Due to technical requirements the power of a compressor is bounded by maximal and minimal power rate, N_e^{max} and N_e^{min}, resulting in

$$N_e^t \leq N_e^{max} s_e^t \quad \text{and} \quad N_e^t \geq N_e^{min} s_e^t. \tag{5.12}$$

Finally we add the inequality

$$f_e^t \leq f_e^{max} s_e^t \tag{5.13}$$

with a theoretical calculated constant f_e^{max} to bound the fuel gas even though an indirect bound is given by equation (5.11). We observed that this additional constraint accelerates running time (see [Möl04]).

Now we investigate the more extensive constraints approximating the fuel consumption f_e^t. For a better understanding we neglect the time index t in the following. As we worked out in Section 4.2 fuel gas of a compressor $e = vw \in E_C$ is given by a nonlinear function. It depends on the flow q_e through the compressors, on the pressure p_v at the beginning of the compressor and on the pressure p_w that is generated by it. We approximate this function $f_e = f_e(q_e, p_v, p_w)$ (see equation (4.11)) by piece-wise linearization, the idea is as follows (see also Chapter 3).

We decompose the domain $[q_e^{min}, q_e^{max}] \times [p_v^{min}, p_v^{max}] \times [p_w^{min}, p_w^{max}]$ of the function $f_e = f_e(q_e, p_v, p_w)$ in tetrahedra that is we determine a grid with three-dimensional grid points (q_e^i, p_v^i, p_w^i). Then we linearize the function within each tetrahedron by means of convex combination. Finally, we ensure that during optimization exactly one tetrahedron can be chosen in order to linearize f_e.

The implementation of this idea is done in the following way.

We associate a weighting $\lambda_e^i \in [0, 1]$ with each grid point (q_e^i, p_v^i, p_w^i). Let Λ_e define the set of indices of grid points and Y_e the set of indices of all tetrahedra for compressor $e = vw$. For each grid point $i \in \Lambda_e$, we calculate the exact value of the nonlinear function $f_e^i = f_e(q_e^i, p_v^i, p_w^i)$.

We linearize f_e by convex combination of grid points therefore we utilize the λ-variables. We have to assure that the positive λ-variables belong to vertices of exactly one tetrahedron $j \in Y_e$. This is modeled by the so called *SOS Type 4* condition (see [MMM06] and Chapter 3). Let us denote by N_e^j the indices of λ-variables associated with the vertices of tetrahedron $j \in Y_e$. We say that a vector $\lambda_e \in \mathbb{R}^{\Lambda_e}$ satisfies the SOS Type 4 condition if there exists some index $j \in Y_e$ such that $\{i \in \Lambda_e \mid \lambda_e^i > 0\} \subseteq N_e^j$. We incorporate this condition implicitly in our branch-and-cut algorithm (see Section 6.2) and do not explicitly model it via additional binary variables.

Having introduced the variables we just write down the linearizing constraints.

At first we have the equation

$$\sum_{i \in \Lambda_e} \lambda_e^i = s_e, \tag{5.14}$$

since the approximated function value is given by convex combination of grid point values if the compressor is operated.

In the following two constraints we describe pressures at the beginning and the end of the compressor as convex combinations of grid points. Here we have to take into account that pressure variables are node dependent, that is, p_v and p_w can be positive even if the compressor is off (Remember that in this case there is pressure equality in the adjacent nodes because of the opened bypass valve). For this reason we introduce nonnegative auxiliary variables $p_{e,v}^h$ and $p_{e,w}^h$ and add appropriate bound constraints. Altogether we receive

$$p_v = \sum_{i \in \Lambda_e} p_v^i \ \lambda_e^i + p_{e,v}^h \qquad \text{and} \quad p_v^{max}\,(s_e - 1) + p_{e,v}^h \leq 0, \tag{5.15}$$

$$p_w = \sum_{i \in \Lambda_e} p_w^i \ \lambda_e^i + p_{e,w}^h \qquad \text{and} \quad p_w^{max}\,(s_e - 1) + p_{e,w}^h \leq 0, \tag{5.16}$$

where p_v^{max} and p_w^{max} are upper bounds for the respective pressure.
At last convex combinations according to gas flow q_e and fuel gas consumption f_e have to be considered yielding

$$q_e = \sum_{i \in \Lambda_e} q_e^i \, \lambda_e^i \quad \text{and} \quad f_e = \sum_{i \in \Lambda_e} f_e^i \, \lambda_e^i. \tag{5.17}$$

5.2.6 Properties of Nodes

As aforementioned there are three kinds of nodes: sources, sinks, and innodes. At a *source* the transmission system is supplied with gas. The gas is taken from the consumers at *sinks*. As the name implies *innodes* can be found inside the network connecting different segments.
For the modeling of a node just one constraint is required. In each node the first law of Kirchhoff must hold. This physical law ensures flow balance in a node meaning that the sum of ingoing gas flows must be equal the sum of outgoing gas flows. For formalizing this consider a node $v \in V$. Let $\delta^+(v)$ be the set of outgoing segments of node v and let $\delta^-(v)$ be the set of ingoing segments. We obtain

$$\begin{aligned} \sum_{e \in \delta^+(v) \setminus (E_P \setminus E_A)} q_e^t + \sum_{e \in \delta^+(v) \cap (E_P \setminus E_A)} q_{e,v}^t = \sum_{e \in \delta^-(v) \setminus (E_P \setminus E_A)} q_e^t + \sum_{e \in \delta^-(v) \cap (E_P \setminus E_A)} q_{e,v}^t \\ - \sum_{e \in \delta^-(v) \cap E_C} f_e^t \end{aligned} \tag{5.18}$$

In this equation we have to differentiate between pipes and remaining segments. There exist two flow variables for pipes depending on space, since in opposition to all other types of segments the gas flow is not modeled constant in the whole pipe. So it becomes clear that we choose $q_{e,v}^t$ for a pipe. Note that also the fuel gas consumption must be regarded in this equation. It is subtracted from the gas flow in the end node of the compressor.
To be strict the flow variable q_v^t of a node $v \in S \cup U$ should be integrated in this equation involving a case differentiation which we did not for transpary reasons.
Besides these equations we have lower and upper bounds for the pressure in a node $v \in V$ yielding

$$p_v^{min} \leq p_v^t \leq p_v^{max}. \tag{5.19}$$

These pressure bounds arise because of technical constraints.
Finally, in case of a sink or a source $v \in S \cup U$, we also receive bounds for the gas flow

$$q_{v,t}^{min} \leq q_v^t \leq q_{v,t}^{max}. \tag{5.20}$$

These flow bounds in principal depend on consumer needs and delivery contracts for the respective sources.

5.3 Linearization of the discretized Partial Differential Equations

In this section we show how we integrate the discretized partial differential equations in our model. There are the continuity and the momentum equation which describe the gas flow in a pipe. Remember that they contain nonlinear terms. As we want a mixed integer program we have to linearize them in an adequate manner. Here we follow the same idea as in the case of the fuel gas consumption of a compressor (see also Chapter 3).

We begin with the discretized continuity equation. Let $e = vw \in E_P \setminus E_A$ be a pipe, then equation (4.5) must be considered for each time step $t \in \mathbb{T} \setminus \{1\}$. We adapt the notation to this chapter and switch from the physical units to the units that we use in our program (see Table 4.3), where Δt corresponds to one hour. So we obtain

$$\frac{q_{e,w}^t - q_{e,v}^t}{L} + A\frac{z_0 T_0}{p_0 T}\left(\frac{p_w^t}{z(p_w^t)} - \frac{p_w^{t-1}}{z(p_w^{t-1})}\right) = 0,$$

where $z(p) = 1 + 0.257\frac{p}{p_c} - 0.533\frac{T_c}{p_c T}\,p$. This equation consists of two linear terms that depend on flow and of two nonlinear terms in pressure variables. Note that the two nonlinear terms are one-dimensional and they have the same structure as they differ just in the time step. Therefore we introduce $|\ \mathbb{T}\ |$ one-dimensional functions of the form

$$conti(p) = A\frac{z_0 T_0}{p_0 T}\frac{p}{z(p)}, \tag{5.21}$$

which depend on the pressure at the end of the pipe in time step t. This function $conti(p)$ is defined on an interval given by the pressure bounds p^{min} and p^{max} in the end node of the considered pipe. For a better understanding we neglect the pipe index e and the time index t. The linearizing idea of this one-dimensional function is quite easy (see also Chapter 3). We decompose the interval $[p^{min}, p^{max}]$ in parts. So we obtain a set of grid points p^i, $i \in \Lambda_{conti}$. We calculate the exact function value $conti(p^i)$ in each grid point $i \in \Lambda_{conti}$. Now we want to approximate $conti(p)$ by the piece-wise linear function that is indicated in Figure 5.4. Hence we introduce a variable λ^i for each grid point and write down the linearizing equations.

$$\sum_{i \in \Lambda_{conti}} \lambda^i = 1 \tag{5.22}$$

$$conti(p) \approx \sum_{i \in \Lambda_{conti}} conti(p^i)\ \lambda^i \tag{5.23}$$

$$p = \sum_{i \in \Lambda_{conti}} p^i\ \lambda^i \tag{5.24}$$

$$\lambda^i \geq 0 \quad i \in \Lambda_{conti}$$

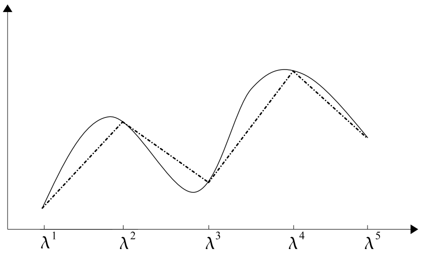

Figure 5.4: Piece-wise linearized one-dimensional function

Observe that the additional condition is needed that at most two λ-variables are positive and if so they must be consecutive. This is called SOS Type 2 condition and equation (5.22) is an SOS Type 2 constraint (see also Section 3.1). This SOS Type 2 condition is modeled implicitly in our branch-and-cut algorithm, see Section 6.2.
We linearize all $\mid \mathbb{T} \mid$ $conti$-functions that occur in the continuity equations for one pipe in this way. Finally, we can write the continuity equations of pipe $e = vw$ in the linear form

$$\frac{q_{e,w}^t - q_{e,v}^t}{L} + conti(p_w^t) - conti(p_w^{t-1}) = 0$$

for $t \in \mathbb{T} \setminus \{1\}$.
Note that considering all pipes we receive altogether $(|E_P \setminus E_A| \cdot \mid \mathbb{T} \mid)$ one-dimensional functions that have to be linearized.

Now we consider the discretized momentum equation. Again we take a pipe $e = vw \in E_P \setminus E_A$ and regard equation (4.6) for each time step $t \in \mathbb{T} \setminus \{1\}$. In our case all pipes are horizontal thus we can neglect the second summand since $\frac{\partial h}{\partial x} = 0$. Adapting the notation and switching to the units in our program yield

$$\begin{aligned} 36^2\, 10^3\, \frac{p_w^t - p_v^t}{L} + 10^3 \frac{\lambda}{2D} \frac{\rho_0 p_0 T}{A^2 z_0 T_0} \frac{(q_{e,w}^t)^2\, z(p_w^t)}{p_w^t} + \frac{\rho_0}{A}(q_{e,w}^t - q_{e,w}^{t-1}) \\ + \frac{\rho_0 p_0 T}{A^2 z_0 T_0} \frac{\frac{(q_{e,w}^t)^2\, z(p_w^t)}{p_w^t} - \frac{(q_{e,v}^t)^2\, z(p_v^t)}{p_v^t}}{L} = 0. \end{aligned}$$

The first and the third summand of this equation are linear. So there remain two nonlinear summands in this equation. Note that we can combine them in the following way

$$\begin{aligned} 36^2\, 10^3\, \frac{p_w^t - p_v^t}{L} + \frac{\rho_0}{A}(q_{e,w}^t - q_{e,w}^{t-1}) \\ + \left(10^3 \frac{\lambda}{2D} + \frac{1}{L}\right) \frac{\rho_0 p_0 T}{A^2 z_0 T_0} \frac{(q_{e,w}^t)^2\, z(p_w^t)}{p_w^t} - \frac{\rho_0 p_0 T}{A^2 z_0 T_0 L} \frac{(q_{e,v}^t)^2\, z(p_v^t)}{p_v^t} = 0. \end{aligned}$$

There are two nonlinear terms of dimension two in this equation which have to be approximated. For the first term we introduce $\mid \mathbb{T} \mid - 1$ two-dimensional functions of the form

$$friction(p, q) = \left(10^3 \frac{\lambda}{2D} + \frac{1}{L}\right) \frac{\rho_0 p_0 T}{A^2 z_0 T_0} \frac{q^2 \, z(p)}{p} \tag{5.25}$$

which depend on pressure and flow at the end of the pipe in time step t. We call this function ‘friction’ as it mainly represents the friction force of the gas with the pipe walls. It is defined on a rectangle given by the pressure bounds p^{min} and p^{max} at the end of the pipe and the flow bounds q^{min} and q^{max} of the pipe.

Analogously, we handle the second nonlinear term. Therefore we need $\mid \mathbb{T} \mid - 1$ two-dimensional functions of the form

$$impact(p, q) = \frac{\rho_0 p_0 T}{A^2 z_0 T_0 L} \frac{q^2 \, z(p)}{p} \tag{5.26}$$

which in contrast depend on pressure and flow at the beginning of the pipe in time step t. Here, we use the name ‘impact’ as the impact pressure in the pipe is described by it. The domain of this function is again a rectangle given by pressure and flow bounds.

Notice that these two functions have the same structure. They only vary in a constant factor and in the space. And as pressure and flow bounds of most pipes do not differ between beginning and end of the segment, the domains of these functions are equal.

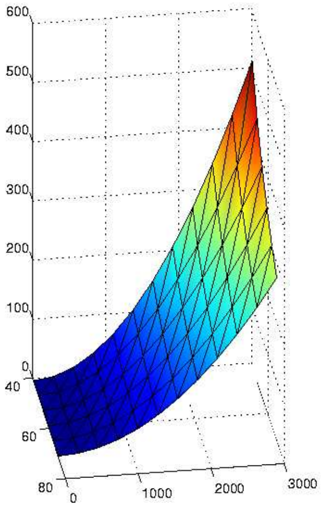

Figure 5.5: Approximation of the $friction$-function

We illustrate the linearization of such a two-dimensional function by means of the $friction$-term, where we neglect pipe and time indices. The method is the same as in the one- and three-dimensional case, see also Section 3.2. First we decompose the domain $[p^{min}, p^{max}] \times [q^{min}, q^{max}]$

in triangles, this defines a two-dimensional grid. Then we linearize the function within each triangle, namely we obtain an approximated function value if we take the convex combination of the exact function values in the vertices of the triangle, see Figure 5.5. Let us translate the idea in the mathematical language. For each grid point (p^i, q^i), $i \in \Lambda_{friction}$, we calculate the exact function value $friction(p^i, q^i)$. Then we introduce a variable λ^i for each grid point $i \in \Lambda_{friction}$. These λ-variables are necessary for the representation of the convex combination. Now we can write down the formulation.

$$\sum_{i \in \Lambda_{friction}} \lambda^i = 1 \tag{5.27}$$

$$friction(p, q) \approx \sum_{i \in \Lambda_{friction}} friction(p^i, q^i)\ \lambda^i \tag{5.28}$$

$$p = \sum_{i \in \Lambda_{friction}} p^i\ \lambda^i \tag{5.29}$$

$$q = \sum_{i \in \Lambda_{friction}} q^i\ \lambda^i \tag{5.30}$$

$$\lambda^i \geq 0 \quad i \in \Lambda_{friction}$$

The second equation yields an approximated value for the $friction$-function. For the λ-variables the SOS Type 3 condition must be fulfilled, i.e., at most three λ's can be positive and these positive variables must belong to one triangle. As in the case of the fuel function of a compressor and the $conti$-function of a pipe this condition is implicitly incorporated in our solution algorithm. Equation (5.27) is an SOS Type 3 constraint.

Approximating all $friction$- and $impact$-functions of a pipe that way, we obtain a piece-wise linearized momentum equation

$$\begin{aligned} & 36^2\ 10^3\ \frac{p^t_w - p^t_v}{L} + \frac{\rho_0}{A}(q^t_{e,w} - q^{t-1}_{e,w}) \\ & + friction(p^t_w, q^t_{e,w}) - impact(p^t_v, q^t_{e,v}) = 0 \end{aligned} \tag{5.31}$$

for pipe $e = vw$ in each time step $t \in \mathbb{T} \setminus \{1\}$.

Observe that altogether we receive $(2 \cdot |E_P \setminus E_A| \cdot (|\ \mathbb{T}\ | - 1))$ two-dimensional functions that must be considered in case of the momentum equation.

5.4 Further Transient Conditions

Besides the discretizations of the partial differential equations we have some other kinds of coupling constraints in the transient model.

First we have to adhere to minimum runtime and downtime for a compressor. Then we need conditions for modeling the switching process of compressors for integrating start-up costs and shut-down costs. These two kinds of constraints are not integrated directly in our model, instead they are included by means of a separation algorithm within the scope of our branch-and-cut algorithm, see Chapter 8 for a thorough investigation.

Then there are constraints concerning the network. Here, we need an initial state for the gas network based upon which optimization for the succeeding time steps is made. Sometimes, we have to respect fixations of a controllable segment for certain time steps. Finally a terminal constraint is required for the overall gas volume in the network.

5.4.1 Minimum Runtime and Downtime

Because of technical conditions a compressor can just be switched on after a certain minimum downtime. Similarly it may only be set into operation if a specified minimum runtime can be complied. These conditions can also be found for machines in other technical fields as for example power production or regenerative energy and are modeled via the following constraints (see for example [Sek00, NNR$^+$00, GNRS00, TKW00, HNNS04]).

As already mentioned there are switching variables s_e^t for each time step $t \in \mathbb{T}$ and for each compressor $e \in E_C$. Let $L_e, l_e \in \mathbb{N}$ denote the minimum runtime and downtime. Then the minimum runtime of compressor e is modeled by the inequalities

$$s_e^t - s_e^{t-1} \leq s_e^j \qquad \text{for} \quad t+1 \leq j \leq \min\{t + L_e - 1, T\}, \tag{5.32}$$

where $t \in \mathbb{T} \setminus \{1, T\}$. They assure that if the compressor is switched on, it must be operated at least L_e time steps (in our case hours) or at least until the end of the considered time horizon. Likewise the constraints

$$s_e^{t-1} - s_e^t \leq 1 - s_e^j \qquad \text{for} \quad t+1 \leq j \leq \min\{t + l_e - 1, T\}, \tag{5.33}$$

where $t \in \mathbb{T} \setminus \{1, T\}$ yield the minimum downtime of compressor e.

In [LLM04] a complete linear description of the polytope defined by these inequalities can be found as well as a separation algorithm is developed with running time $O(T)$.

5.4.2 Switching Processes

Switching a compressor on or off involves costs. Obviously these costs occur in the objective function and variables and constraints are needed for modeling the switching processes. Notice that such costs must also be considered in other practical applications, see [Sek00, NNR$^+$00, GNRS00, AC00, HNNS04].

For modeling start-up costs of compressor e we need variables $s_{e,up}^t$ for $t \in \mathbb{T} \setminus \{1\}$, where $s_{e,up}^t$

is 1 if and only if machine e is switched on in time period t. Similarly there are variables $s^t_{e,down}$, $t \in \mathbb{T} \setminus \{1\}$, indicating if compressor e is shut down in period t. The following conditions ensure the properties of these variables

$$s^t_e - s^{t-1}_e - s^t_{e,up} + s^t_{e,down} = 0 \qquad \text{for} \quad t \in \mathbb{T} \setminus \{1\} \tag{5.34}$$

$$s^t_{e,up} + s^t_{e,down} \leq 1 \qquad \text{for} \quad t \in \mathbb{T} \setminus \{1\} \tag{5.35}$$

see also [Sek00, AC00]. If compressor e is switched on in period t, i.e., $s^t_e = 1$ and $s^{t-1}_e = 0$, then (5.34) yields $s^t_{e,up} = 1$ and $s^t_{e,down} = 0$ since the variables in these equations are binary. Conversely, we receive $s^t_{e,down} = 1$ and $s^t_{e,up} = 0$ if the compressor switches from up in time step $t - 1$ to down in step t, i.e., $s^t_e = 0$ and $s^{t-1}_e = 1$. Inequalities (5.35) are needed if the running state of the machine does not change, i.e., $s^t_e = s^{t-1}_e$. In that case, (5.34) results in the equality of $s^t_{e,up}$ and $s^t_{e,down}$ and thus (5.35) forces them to be zero. Note that these inequalities can be neglected if start-up and shut-down costs are positive and we consider a minimization problem.

These constraints and the constraints of the previous subsection do not explicitly appear in our model. In [Mar05] the polytope given by the inequalities (5.32) to (5.35) is investigated and a complete linear description is presented. Moreover a separation algorithm is specified having running time $O(T)$ for using these facet-defining inequalities in a branch-and-cut framework. The benefits of this separation algorithm instead of the explicit consideration of the conditions is analyzed by means of the example of transient gas network optimization. Therefore we integrate this separation procedure in our branch-and-cut algorithm to fulfill the runtime and switching conditions. In Chapter 8 we give an overview of [Mar05] and comment on the main results.

5.4.3 Initial State

For time-dependent gas network optimization, an initial state of the gas network is required from which the optimization process starts. Such an initial state gives a complete description of the gas network at the beginning of the planning horizon. Hence, it is defined by values of all flow and pressure variables for $t = 1$.

So we need concrete values of the following variables.

$$\begin{array}{ll}
q^1_e & \text{for } e \in E_C \cup E_V \cup E_A \\
q^1_{e,v}, q^1_{e,w} & \text{for } e = vw \in E_P \setminus E_A \\
p^1_v & \text{for } v \in V \\
q^1_v & \text{for } v \in S \cup U
\end{array}$$

Obviously, the settings of these variables must comply the properties of the elements of the gas network. Thus they have to fulfill the constraints that also occur in the stationary case which are described in Section 5.2.

Note that by means of these flow and pressure values the remaining variables of time step $t = 1$ can be calculated. These are the fuel gas and the power of the compressors as well as the switching variables of valves and compressors, see Table 5.1.

5.4.4 Fixations of controllable Segments

A special kind of condition are fixations of controllable segments. This means that for a certain time step the switching variable of a valve or a compressor has a predefined value, so the corresponding segment must be on or off in this time step. Such fixations are necessary to model different situations.
For example, if a compressor must be shut down for a couple of time steps because of maintenance, its switching variable must have value 1. Or if a compressor was switched on just before the beginning of the considered time horizon, its switching variable must initially equal 1 to comply the minimum runtime of the machine.
Fixations of a valve can be considered, if we want the gas flow to follow a certain transportation direction, and thus to block a part of the gas network. This is useful for maintenance of pipes.

5.4.5 Terminal Condition

A so called terminal condition is required to guarantee operational availability after the considered time horizon. Otherwise the optimization process would result in very low pressure and flow values for the last time step (and the network is no more operational) since it is cheaper to pump dry the gas network than to transport the gas flow from the sources. There are two approaches that avoid such situations. In the first approach cyclical conditions for the pressure values in all nodes for the first and the last time step are assumed. The second one requires a lower bound on the total gas volume flow in the network at the end of the considered horizon. Since the gas volume flow in a pipe depends on the pressure values at the beginning and end node (see below), the cyclical pressure conditions imply the gas volume flow condition. As in [ES03, ES05] we follow the approach of setting a lower bound on the total gas volume flow in the network at time step $t = T$. Note that in this context we disregard the few amount of gas volume in non-pipe elements.
According to [Sek00], the gas volume flow in a pipe $e = vw \in E_P \setminus E_A$ for time step $t \in \mathbb{T}$ can be approximated by

$$V_e^t = \frac{LD^2\pi}{4}\frac{\rho_m^t}{\rho_0} = \frac{LD^2\pi}{4}\frac{z_0 T_0}{z_m T p_0} p_m^t,$$

where L is the length and D is the diameter of the pipe, the mean z-factor z_m is given by an appropriate constant value, and for the mean pressure p_m^t we take

$$p_m^t = \frac{p_v^t + p_w^t}{2}$$

the arithmetic mean. Remark that the gas volume flow depend on the pressure at beginning and end of the pipe.
To obtain the total gas volume for time step t we have to sum up V_e^t over all pipes $e \in E_P \setminus E_A$. We require that the total gas volume at the end of the considered time horizon is at least as large as that at the beginning, i.e.,

$$\sum_{e \in E_P \setminus E_A} V_e^1 \leq \sum_{e \in E_P \setminus E_A} V_e^T. \tag{5.36}$$

Notice that this is a linear inequality in pressure variables. Since transporting gas causes costs, the optimization process will tend to fulfill (5.36) at equality.

5.5 Objective Function

We conclude this chapter with the objective function of the gas network optimization model. It consists of two parts. On the one hand there is the fuel gas consumption and on the other hand there are the switching costs of the compressors in all time steps.
The total amount of fuel gas consumption of the TTO problem is given by the sum of the fuel gas of all compressors during the whole time horizon. So we receive

$$\sum_{t \in \mathbb{T}} \sum_{e \in E_C} f_e^t. \tag{5.37}$$

Further on, we consider in our model constant start-up and shut-down costs. With $C_{e,up}^t$ and $C_{e,down}^t$, $t \in \mathbb{T} \setminus \{1\}$, we denote the costs if compressor $e \in E_C$ is switched on or off in time step t. Hence, the switching costs in our model can be quoted by

$$\sum_{t \in \mathbb{T} \setminus \{1\}} \sum_{e \in E_C} C_{e,up}^t s_{e,up}^t + C_{e,down}^t s_{e,down}^t. \tag{5.38}$$

Addition of (5.37) and (5.38) yields our objective function.

Chapter 6

Handling of SOS Conditions

In this chapter we show how we incorporate the SOS conditions of the model in our branch-and-cut algorithm. We integrate them implicitly by means of our branching scheme without using additional (binary) variables. Remember that the SOS conditions in our model come from the approximation concepts for the nonlinear functions that arise from gas dynamics in pipes and fuel gas consumption of compressors. There are one-, two- and three-dimensional nonlinear functions in the problem of TTO, hence we must handle SOS conditions of Type 2, of Type 3 and of Type 4. At first we concentrate on the approximation grids for the nonlinear functions. In this context, we treat the approximation errors, and a good choice for the grid decomposition is presented. Then we introduce our branching strategies, where we also discuss the concept for the classical SOS Type 2 condition. The branching idea for SOS conditions of higher dimensions was developed within the scope of the stationary case of gas network optimization, see [Möl04, MMM06]. In the transient case, we improve this SOS branching by means of adequate preprocessing strategies. Finally, we describe a separation algorithm based on the idea of linking approximation grids, hence combining different SOS constraints. This separation algorithm results from polyhedral studies made in context with the stationary case of gas optimization, see [Möl04, MMM06]. We apply this separation idea to the transient case.

6.1 Linearizing Grids

In this section we concentrate on the grids that results from approximation of nonlinear functions. We regard approximation errors, infeasibility problems and present decomposition criteria.

At first we want to remind the general linearization concept for a nonlinear function (see Chapter 3). Initially, we triangulate the domain of the function. So we obtain a set of grid points and a set

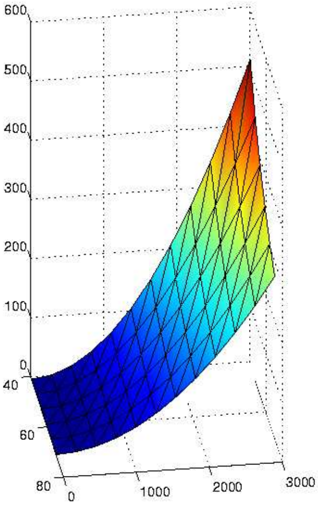

Figure 6.1: Triangulation of the $friction$-function

of simplices. Then we approximate the function linearly within each simplex by means of convex combination. The question is how to choose an adequate grid in order to receive a good approximation.

Remember that there are four kinds of nonlinear functions in our model. At first, the one-dimensional $conti$-function, see (5.21) , resulting from the continuity equation. Then, there are two functions of dimension two, the $friction$-function (5.25) and the $impact$-function (5.26), that come from the discretized momentum equation and differ just in a scaling factor from each other. Finally, the fuel gas consumption (4.11) of a compressor is defined by a three-dimensional function.

Note that if a linearization grid of a function is given, we can calculate the maximal absolute approximation error in each simplex, and we receive the overall maximal absolute error of the complete triangulation. But it does not seem reasonable to look just at the absolute error. In Figure 6.1 we see a picture of the $friction$-function which becomes steeper for large flow-values. Thus in the region of large flow-values the absolute error becomes bigger but the relative approximation error remains small because of the increasing function value. Moreover, the relative error is necessary if we want to compare the quality of different approximation grids. Therefore we consider the relative approximation error as accuracy criterion for the quality of our linearization grids. Note that in this context, we only regard the part of the function's domain that is relevant in practical applications. For example, for a pipe the theoretical flow bounds are given by 0 and 3000, but in practice flow values vary between 500 and 1500. Thus, if we calculate the maximal relative error of a pipe grid, we only consider simplices with flow values between 500 and 1500. Altogether we say that an approximation grid of a nonlinear function has *accuracy* ε, if the maximal relative error in the practical relevant part of the domain is at most ε.

Before we concretize the grid determination for the nonlinear functions in the problem of TTO we dwell on numerical difficulties that arose in our development phase. We look at the linearized momentum equation (5.31). As the common length of a pipe is about 100 km, the coefficients of the pressure variables in (5.31) are in the range of 10^4. All other coefficients in this equation equal one or are about one in case of the flow variables. This fact causes numerical difficulties, as a little change of a pressure variable has a big effect on the equation. Therefore, we introduce scaling factors $factfric$ and $factimpact$ for the $friction$- and $impact$-term in equation (5.31) yielding

$$\begin{gathered} 36^2\,10^3\,\frac{p_w^t - p_v^t}{L} + \frac{\rho_0}{A}(q_{e,w}^t - q_{e,w}^{t-1}) \\ + factfric \cdot friction(p_w^t, q_{e,w}^t) - factimpact \cdot impact(p_v^t, q_{e,v}^t) = 0, \end{gathered}$$

whereas we have to adapt our nonlinear functions (5.25) and (5.26) in an adequate manner, namely by dividing them by the corresponding factor. Note that we cannot scale the flow variables in the momentum equation as they also appear in other constraints. Testing our scaling method (also within the scope of the simulated annealing algorithm) we observe that we can overcome the calculation problems in the linearized momentum equation. It also turns out that the $friction$-term has more influence in this equation than the $impact$-term. Good choices for the scaling factors are $factfric = 5000$ and $factimpact = 5$. These values especially guarantee that the absolute approximation errors of the $impact$- and $friction$-term are in the same order of magnitude.

Moreover we have to handle feasibility problems. Because of the errors resulting from the discretizations of the partial differential equations and from the approximations of the nonlinearities, the mixed integer program that we consider is not feasible anymore. A first step to treat these infeasibility problems is to round the coefficients in all linearizing equations. Our tests show that it is not practicable to consider a big number of decimal places in the equations that give us the convex combinations of the approximated function value and of the individual components of the nonlinear function. Notice that in practice for pressure variables an absolute error of 0.1 bar is acceptable, for flow variables this error may even be worse. Assuming these pressure and flow accuracies, we round all coefficients of the linearizing equations down on two digits. Further on, we introduce slack variables. For each compressor we define slack variables for all linearizing equations (5.15) to (5.17) for each time step $t \in \mathbb{T}$. Accordingly, considering a certain pipe, we define slack variables for equations (5.23) and (5.24) of the linearized $conti$-function for each $t \in \mathbb{T}$. Furthermore, for $t \in \mathbb{T} \setminus \{1\}$, we add slacks to equations (5.28) to (5.30) of each $friction$-term and analogously for all $impact$-terms. Finally, we need slack variables in the linearized continuity and momentum equations. Note that these slack variables do not occur in the objective function. Instead, we introduce constant upper and lower bounds for them which depend on the accuracy acceptable in practice (as mentioned above). By this proceeding, the resulting mixed integer program becomes feasible, and our algorithm yields practical applicable solutions.

Having addressed the accuracy aspects for the linearizing grids, we continue with their concrete determination. At first we consider the one-dimensional $conti$-function. Its function values range from 40 to 75 for a standard pipe in our test networks. Figure 6.2 illustrates this function. As we can see, it is almost linear. So we approximate the $conti$-term by means of one line segment defined by the boundary points $(p^{min}, conti(p^{min}))$ and $(p^{max}, conti(p^{max}))$. Then we calculate

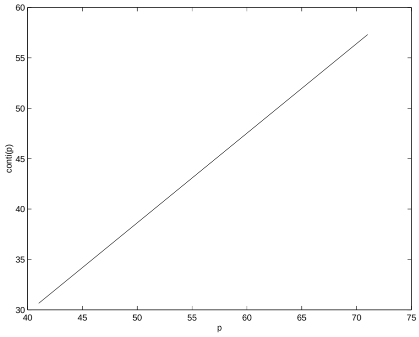

Figure 6.2: Illustration of the $conti$-function

approximation errors for this "trivial" grid. We receive 0.69 as maximal absolute error, and the maximal relative error is 0.012. This accuracy is sufficient for practical applications.

Thereafter, we regard the two-dimensional functions in our TTO model. As already mentioned, they just differ in a constant factor. Since such a factor does not affect the relative approximation error, we choose the $friction$-function as an example for this case. To obtain a two-dimensional grid, we us the mesh generator from the software package `KARDOS` [ELR02]. Remember that the domain of the $friction$-function is a rectangle. `KARDOS` comprises a function that determines a uniform triangulation of a rectangle as shown in Figure 6.3, if the user specifies the number of vertical and horizontal subdivisions. So we can test different partitions in pressure and flow direction whereas the approximation accuracy for the grid stays the same (see the computational results in Section 10.2). The function value of the friction term varies between about 0 and 500 (taking into account the scaling factor $factfric = 5000$) depending on the specific pipe data. Figure 6.1 shows a picture of the function. We see that for low flow values, the absolute approximation error is small but the relative error is very big because of small friction values. On the other side, considering big flow values, the absolute error becomes bigger whereas the relative error is small. In the region that is relevant for our practical application (flow of 500 to 1500) the absolute as well as the relative error are of adequate size.

Finally, we look at the three-dimensional nonlinear function in our model, the fuel gas consumption of a compressor. The software package `KARDOS` does not include a three-dimensional mesh generator. It just offers a refinement function for a given grid. Because of pressure and flow bounds, the fuel gas function is defined on a cube. In a first step, we subdivide this cube into 12 tetrahedra and use the refinement function of `KARDOS` to obtain a grid of higher accuracy. But this approach has the disadvantage that we receive a uniform triangulation that does not respect the characteristics of our function. Note that in the two-dimensional case we also consider uniform grids, but we can affect the form by choosing different subdivisions in pressure and flow direction. So, we developed another approach in cooperation with [Her06]. The idea is the following. At first, regarding the domain of the function - the cube - we see that we can omit half of it as we assume that the

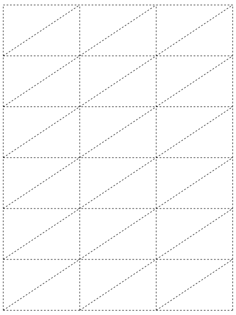

Figure 6.3: Triangulation of a rectangle with two vertical and five horizontal subdivisions

compressor increases the pressure. This means that the outgoing pressure of a compressor is at least as large as the pressure at the beginning of it. Thus, we have to triangulate half of the cube. We start with an arbitrary triangulation. Note that we choose this starting triangulation in such a way that its grid points are given by the valid vertices of the cube, i.e., vertices that fulfill our pressure condition. Now, we determine the tetrahedron with the maximal absolute approximation error. We take the point where this maximal error occurs and add it to our triangulation. Then we specify a new division into tetrahedra, respecting our extended set of grid points. Therefore we use the software package `polymake` [GJ99], which calculates the Delaunay triangulation of our grid points. This procedure is executed iteratively, i.e., determination of the tetrahedron with the maximal absolute error, adding the corresponding point to the set of grid points, and calculation of the Delaunay triangulation, until a specified accuracy is attained.

Notice that this accuracy is calculated based on the absolute error. Remember that we mentioned at the beginning of this section that we use the relative error as criterion for our approximation grids. But in this grid generation process we must consider the absolute error. Otherwise, our algorithm would tend to add points near the boundary where in- and outgoing pressure equalities hold, since there the function values are very low and hence the relative error is extensive. The resulting triangulation would be impractical for our purpose, since the pressure is always considerably increased by a compressor in applications, and therefore a fine grid near the boundary is useless. So we first determine a triangulation respecting the absolute error. Thereafter, we have to estimate the relative approximation error of the generated grid to evaluate its accuracy for our test runs.

In Figure 6.4, a triangulation for our three-dimensional function is shown that was generated by the strategie described above. Note that the colors in this picture indicate the fuel gas consumption of the compressor. Blue color stands for few fuel gas. Green or yellow regions denote mean consumption of the machine which are the relevant values in practice. Finally, orange and red shows very high costs. All these function values vary from zero (no pressure increase of the machine) to about 25.

This concludes our grid investigations. In our computational results, see Chapter 10, we consider

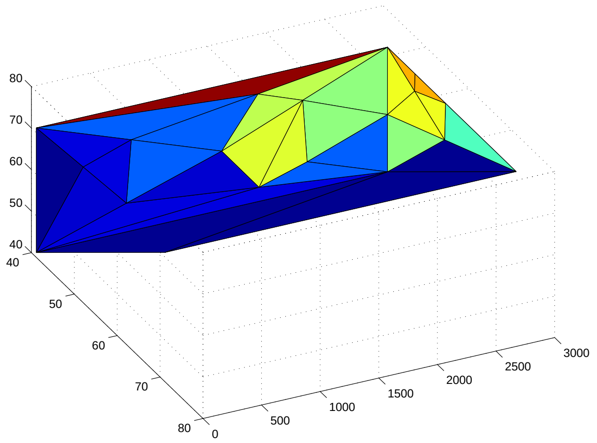

Figure 6.4: Triangulation for the function of fuel gas consumption

three accuracy levels, namely $\varepsilon = 0.15$, $\varepsilon = 0.1$ and $\varepsilon = 0.05$.

6.2 Branching for SOS Conditions

Now, we show how we guarantee the fulfillment of the SOS conditions in our algorithm. Remember that there are SOS conditions of Type 2, of Type 3 and of Type 4 in our gas network optimization model. We begin this section with a brief literature survey concerning SOS conditions. Then, we present the branching idea of the classical SOS Type 2 (briefly SOS 2) condition. Subsequently, we concentrate on the branching scheme for SOS in higher dimensions where we follow the description of [MMM06].

Branching strategies for the classical SOS 2 were developed in the seventies and can be found in [Bea79, BF76, BT70]. [Tom81, Möl04, MMM06] extend the SOS concept to higher dimensions. In [Tom81] nonlinearities are piece-wise linearly approximated using the standard simplicial subdivision of the hypercube. Further on, a branch-and-bound method is presented to handle the resulting extensions of SOS constraints. [Möl04, MMM06] adapt these method to the stationary case of gas network optimization. There, a general definition of SOS conditions in higher dimensions is given (see also Definition 3.2.1) and corresponding branching strategies are discussed. [dFJZZ05] present an SOS approach for discontinuous piece-wise linear functions. They consider optimization problems with discontinuous piece-wise linear objective function and prove the advantages of SOS over the binary model.

Let us begin with the classical branching idea for SOS 2. Remember that a set of consecutive variables $\{\lambda_1, \ldots, \lambda_n\}$ is SOS 2, if at most two variables can be nonzero and if so they must be adjacent. It is reasonable that in case of SOS it is more efficient to branch on sets of variables rather than on individual variables [BT70]. So, if there are two positive variables λ_j and λ_k with $|k-j| > 1$, we choose one of them and divide the set into two parts. If we take for example λ_k as cutting variable, we add the equation $\sum_{i=1}^{k} \lambda^i = 1$ to the first subproblem and $\sum_{i=k}^{n} \lambda^i = 1$ to the second one. Figure 6.5 illustrates this concept. In the literature cited above, several selection strategies for the cutting variable can be found.

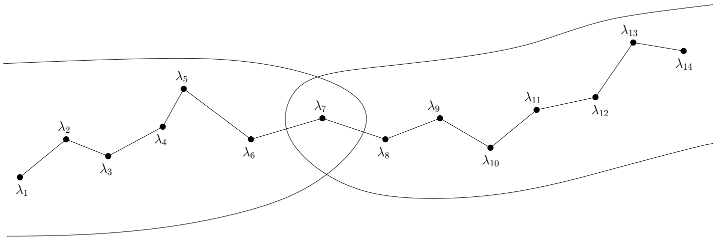

Figure 6.5: Branching on SOS 2

Now, we come to the branching concepts for SOS conditions in higher dimension. We illustrate the ideas by means of the two-dimensional approximation grid of the $friction$-function in the linearized momentum equation. As we will see they can be generalized to linearization grids of higher dimensional functions.

For pipe $e = vw \in E_P \setminus E_A$, the function $friction(p_w, q_{e,w})$ is defined on the rectangle $[p_w^{min}, p_w^{max}] \times [q_e^{min}, q_e^{max}]$. For a better understanding we neglect the time index t. We determine a uniform triangulation of this rectangle, i.e., we choose equidistant partitions $p_w^{min} = p_{w,1} \leq p_{w,2} \leq \ldots \leq p_{w,k} = p_w^{max}$ and $q_e^{min} = q_{e,1} \leq q_{e,2} \leq \ldots \leq q_{e,l} = q_e^{max}$ with $k, l \geq 2$. By construction we obtain $k \cdot l$ grid points $(p_{w,i}, q_{e,j})$, $i, = 1, \ldots, k$, $j = 1, \ldots, l$. For $s \in \{2, \ldots, k-1\}$ we divide them into two sets $L_s = \{(i,j) \in \{1, \ldots, k\} \times \{1, \ldots, l\} : p_{w,i} \leq p_{w,s}\}$ and $R_s = \{(i,j) \in \{1, \ldots, k\} \times \{1, \ldots, l\} : p_{w,i} \geq p_{w,s}\}$. We call the branching *vertical branching* (for s), if we add to the first subproblem the equation

$$\sum_{j \in L_s} \lambda^j = 1$$

and to the second subproblem the equation

$$\sum_{j \in R_s} \lambda^j = 1,$$

see Figure 6.6 for such an example.

Note that the branch-and-bound tree resulting from vertical branching has at most $k-1$ leaves (number of consecutive column pairs) and thus $2k-3$ nodes. Similarly, we branch in the second direction, i. e., $q_{e,w}$, which we call *horizontal branching*. Observe that if there are no further vertical

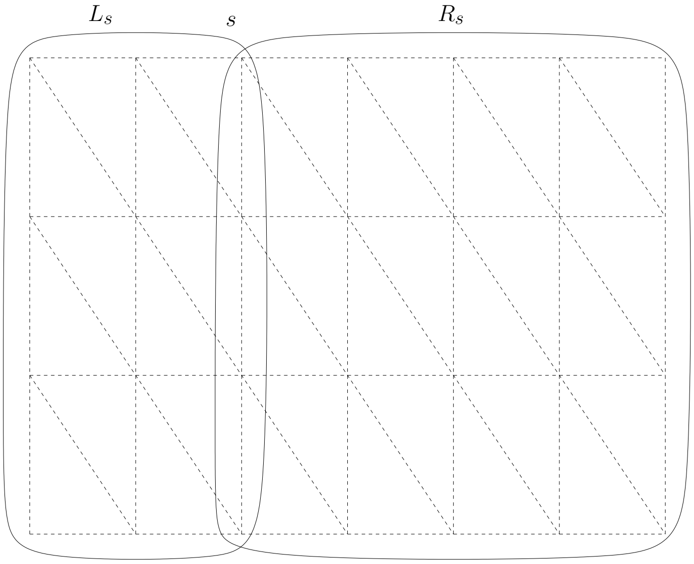

Figure 6.6: Illustration of vertical branching

or horizontal branchings the positive λ-variables are restricted to exactly one rectangle. Now we just have to branch on the two triangles constituting this rectangle.

In the general case of an n-dimensional grid, we have to branch in all n directions. Therefore, we generally call it *hyperplane branching* (in a certain direction). After hyperplane branching in all directions - if we suppose a uniform triangulation - the remaining positive λ-variables belong to the vertices of exactly one hypercube. Then we have to confine the positive λ-variables to a simplex of this hypercube. For example, consider a uniform grid of the three-dimensional fuel gas consumption function (our first grid approach). After branching in the three directions, the positive λ-variables are restricted to a cube which is divided into tetrahedra.

In principle the idea of vertical branching (and hyperplane branching in a certain direction) is that used for the classical SOS 2 branching. Let us depict the transfer to the general extension by means of the two-dimensional example above. We write λ_{ij} for the λ-variable associated with grid point $(i,j) \in \{1,\ldots,k\} \times \{1,\ldots,l\}$. Further we sum up the variables of column i and define $\hat{\lambda}_i = \sum_{j=1}^{l} \lambda_{ij}$. Standard branching on the (relaxed) one-dimensional SOS Type 2 constraint $\sum_{i=1}^{k} \hat{\lambda}_i = 1$ yields vertical branching.

The branching strategies presented till now just guarantee SOS conditions for uniform grids. But in case of the fuel gas function, we include nonuniform approximation grids in our model. So we return to hyperplane branching again. First look at the sets L_s and R_s. Instead of taking the subdivision points $p_{w,i}$ as ‘cut points’ for these sets, we could choose arbitrary values, for example a coarser step size. Using such a partition, we can handle compressor-grids as well as uniform

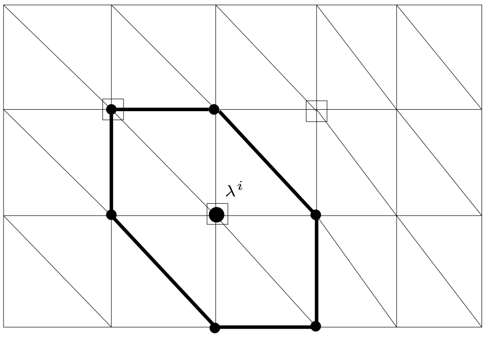

Figure 6.7: An example for variable branching: the bullet nodes indicate the neighbors of λ^i and the rectangles indicate fractional variables

pipe-grids or, for instance, pipe-grids that are no more uniform because of some refinements of triangles in a certain region.

Further on, in the case of nonuniform triangulations, the above statement that after hyperplane branching in all directions the positive λ-variables belong to the vertices of one hypercube, is no longer valid. Here the following algorithm assures that the positive λ-variables fulfill the general SOS condition. We use the notation of Chapter 3 and regard an arbitrary triangulation. Let Λ be a set of n-dimensional grid points and $Y = \{N^1, \ldots, N^d\}$ a set of simplices, where each of the simplices is represented by a subset $N^i \subset \Lambda$ of grid points. We say that a vector $\lambda \in \mathbb{R}^{|\Lambda|}$ satisfies the SOS Type k condition ($k = \max_i |N^i|$) with respect to (3.1) and the triangulation Y if there exists some index $r \in \{1, \ldots, d\}$ such that $\{i \in \Lambda : \lambda^i > 0\} \subseteq N^r$, i.e., the positive λ-variables belong to one simplex. Now let $\bar{\lambda}$ be an optimal LP-solution that does not satisfy this condition and $\bar{N}$ be a set of indices of the fractional variables, i. e., $0 < \bar{\lambda}^i < 1$ if and only if $i \in \bar{N}$. Let Γ^i denote the neighbors of grid point i in the triangulation Y, i.e., the set of indices of grid points that are adjacent to grid point i, see Figure 6.7. Consider the following algorithm.

Algorithm 3 Variable Branching

```
1: for $i \in \bar{N}$ do
2:     if $\Gamma^i \cup \{i\} \not\supseteq \bar{N}$ then
3:         goto 6
4:     end if
5: end for
6: Split the problem in the following way
7:         First subproblem: Add the condition $\sum_{j \in \Gamma^i \cup \{i\}} \lambda_j = 1$.
8:         Second subproblem: Add the condition $\lambda_i = 0$.
9: Stop.
```

Observe that - as the LP-solution $\bar{\lambda}$ does not satisfy the SOS condition - there exists some $i \in \bar{N}$

with $\Gamma^i \cup \{i\} \not\supseteq \bar{N}$ and the algorithm terminates with two branches in which $\bar{\lambda}$ is not feasible.
We call this kind of branching *variable branching*. An illustration of it can be found in Figure 6.7. Note that using this algorithm we can guarantee that the SOS condition is fulfilled. The number of branch-and-bound nodes generated by it depends on the simplicial subdivision of the triangulation. If we assume the neighborhood $|\Gamma^i|$ to be constant, which is the case in our applications, the algorithm yields $O(\Lambda)$ leaves (as in case of SOS 2).
In the branch-and-cut algorithm for the problem of TTO, we use hyperplane branching as long as there are possible candidates, since it is more efficient to branch on sets than on variables. Thereafter, we continue with variable branching to fulfill all SOS conditions.

6.3 Preprocessing for SOS Conditions

To accelerate our branch-and-cut algorithm, we extend the branching strategies with suitable preprocessing methods. We present two kinds of preprocessing processes that are based on similar ideas. The first strategie is in connection with hyperplane branching and the second one exploits flow bounds resulting from supplier and consumer behavior.

We begin our description with the preprocessing ideas concerning hyperplane branching. Observe that hyperplane branching (and SOS 2 branching) implicitly induces a cut for each of the two subproblems. Let us illustrate this idea by means of vertical branching for the two-dimensional grid of the $friction$-function, see above. Have a look at Figure 6.6. The subproblems resulting from vertical branching are defined by the sets L_s and R_s. Remember that these sets affect the pressure variable p_w^t at the end of the pipe for the considered time step as it is the first component of our $friction$-function. If we look at the left son of this branching step, hence the set L_s, the inequality $p_w^t \leq p_{w,s}$ is valid for the subproblem. Accordingly, $p_w^t \geq p_{w,s}$ holds for the right son. Obviously, these cuts are implicitly incorporated in the subproblems, but we received better computational results if we add them explicitly.

Notice that the designation of these cuts is a little bit more complicated, if we consider nonuniform grids or do not use grid points as ‘cut points’ for the determination of the sets L_s and R_s. Figure 6.8 shows such a situation. To obtain the upper bound of the pressure variable p_w^t in the left son, we have to determine the maximum pressure value of all grid points that lie in the set L_s, i.e., the maximal pressure that can be attained in this set. In Figure 6.8 the corresponding points are marked by black squares. If the pressure value, i.e., the first component of these points equals p_u, the inequality $p_w^t \leq p_u$ is valid for the left son and we add it to the branch. Analogously, we specify a valid cut for the right son. For this purpose, we calculate the minimum pressure value in the set R_s and receive a lower bound for p_w^t.

The cuts induced by hyperplane branching can be used for further preprocessing strategies. For an

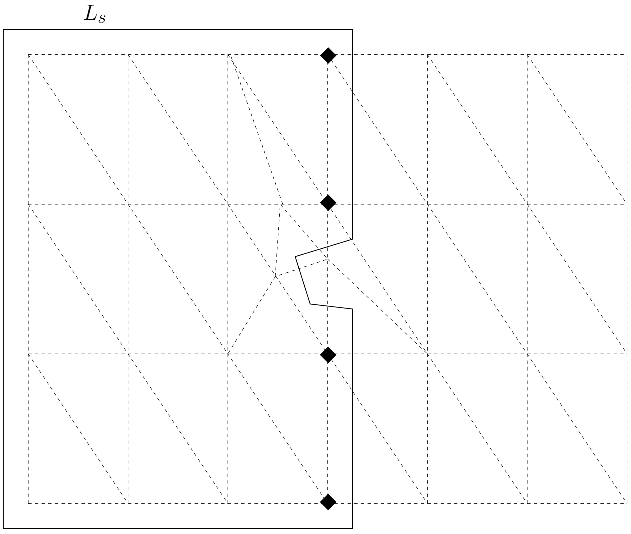

Figure 6.8: Illustration of cut induced by vertical branching for a nonuniform grid

explanation, we regard again the $friction$-function with vertical branching. There, we obtain cuts that concern the pressure variable p_w^t. As pressure variables are nodal variables, we can use the cut bounds to restrict the pressure values of other approximation grids and hence, to eliminate further linearizing variables in the subproblems. Let us choose another linearization grid that includes the pressure variable p_w^t (for example we can take the $impact$-function of an outgoing pipe of node w) and consider the left subproblem. An illustration of the idea can be found in Figure 6.9. The induced cut is sketched as straight line, thus the region of the rectangle to the right of it is infeasible. Therefore, we can eliminate the λ-variables belonging to marked grid points in the left branch by setting them to zero.

By means of this preprocessing method, we can branch on several grids in one iteration. There are several possibilities to combine approximation grids.

At first we consider the case that the hyperplane branching candidate affects the pressure variable of node v in time step t. Here, we consider the $conti$- and $friction$-function of each ingoing pipe of this node at time t (as they depend on pressure at the end of the pipe) and also the $fuel$-function of each ingoing compressor, and eliminate infeasible λ-variables. Further on, we consider the $impact$-function of each outgoing pipe and the $fuel$-function of each outgoing compressor of v and set the corresponding λ-variables to zero.

Even if flow variables belong to segments, as they indicate the gas flow in the segment or at the beginning/end of it, we can use the cuts induced by hyperplane branching on a flow variable to eliminate linearizing variables because of Kirchhoff's law. Remember that this law ensures flow balance in a node, i.e., the sum of ingoing gas flows equals the sum of outgoing gas flows. Thus, we can apply the induced cuts to other approximation grids, but just in case that there is exactly one outgoing and one ingoing segment. If there are several parallel segments, we do not know

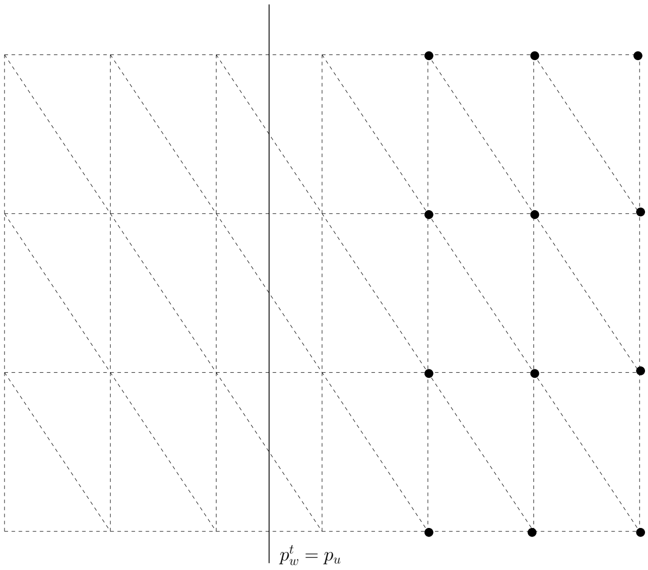

Figure 6.9: Elimination of λ-variables using cut induced by branching

how the gas flow distributes, and so we cannot use the flow bounds for other grids. Only if the considered segment is a compressor, we can accept a further parallel segment as each compressor has a bypass valve.

There are three possibilities that a flow variable is concerned by hyperplane branching, either by the first component of the $fuel$-function of a compressor or by the second component of the $friction$- or $impact$-function of a pipe. In case of compressor we only have to consider one flow variable, whereas for a pipe the flow differs at the beginning and at the end.

If hyperplane branching concerns the flow variable of a compressor $e = vw$, we look at both nodes v and w. If v has exactly one ingoing segment, a pipe, we can eliminate λ-variables of its $friction$-grid, but only if, in addition to its bypass valve, the compressor e has no further parallel segments. Alternatively, if v has just a compressor and its bypass valve as ingoing segments, the λ's of the $fuel$-grid are concerned. Analogously, we handle the end node w of the compressor. Here, we must take the $impact$-grid of a pipe, and additionally we have to take the fuel gas consumption f_e^t of the compressor into account.

Otherwise, if one of the flow variables of a pipe $e = vw$ is affected by branching on a pipe grid, we just have to consider one of the nodes. In case of branching on the $impact$-grid, we regard node v (as impact values depend on the flow at the beginning of the pipe) and proceed as described above, but only if pipe e has no parallel segment. Accordingly, if we branch on the $friction$-grid, end node w must be considered.

We conclude this section with the second preprocessing strategie that we developed for our approximation grids. As already mentioned, we exploit flow bounds of sources and sinks that are defined by delivery and consumer behavior. In general, such bounds are stronger than the technical ones given for pipes or compressors. Note that only pipe grids are concerned using this preprocessing methods as a compressor $e = vw$ can only be found in the inner of a network, hence its defining nodes are innodes $v, w \in V \setminus (S \cup U)$.

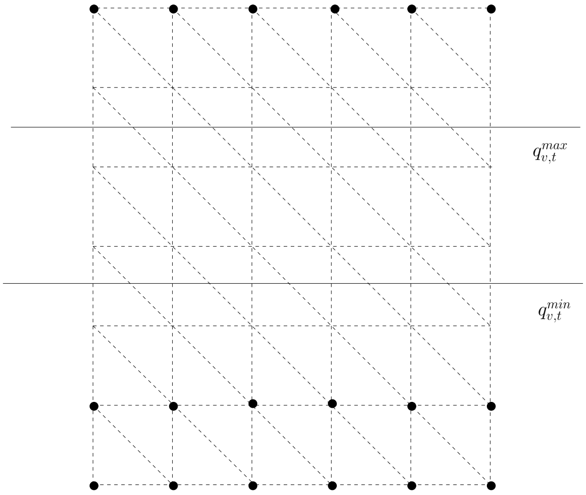

Figure 6.10: Preprocessing via supplier bounds of a source

At first, we consider a source $v \in S$ with its flow bounds $q_{v,t}^{min}$ and $q_{v,t}^{max}$. If source v has exactly one outgoing segment and this is a pipe, we can apply these bounds to determine the feasible region of its $impact$-grid in time step t. Have a look at Figure 6.10. The source bounds are indicated by straight lines. Note that these lines are horizontal as the flow variable defines the second component of the $impact$-function. The feasible region of this grid is given by the small stripe between the two lines. Therefore, all λ-variables corresponding to marked grid points must be zero.
In case of a sink $v \in U$, the procedure is similar. Again we take the flow bounds $q_{v,t}^{min}$ and $q_{v,t}^{max}$ for a certain time step. If v has a pipe as single ingoing segment and has no outgoing segments (note that it is rarely possible that a sink has also outgoing segments, see test network one in Section 10.1), we use the sink bounds for elimination of λ-variables of the $friction$-grid for time step t.

6.4 Separation for SOS Conditions

Finally, we present a separation algorithm for SOS conditions in connection with approximation of functions which combines different SOS conditions. To develop the separation strategie, substructures of the linearizing model are studied. The studies of such structures and the resulting algorithm were made within the scope of stationary gas network optimization. We just give a brief introduction of the idea where we refer to [MMM06]. A detailed description can be found in [Möl04]. We adapt this separation algorithm to the transient case of gas optimization.

We illustrate the idea by means of an example, whereby we neglect the time index t. This idea can be applied to the general case of linking arbitrary SOS conditions. At the end of this section we list the combination possibilities that we use in the transient model of gas network optimization. Let $e = uv, g = vw \in E_P \setminus E_A$ be two pipes in a row and consider the $friction$-function $friction(p_v, q_{e,v})$ of the first pipe and the $impact$-function $impact(p_v, q_{g,v})$ of the second one. As the $friction$-term depends on the outgoing pressure and the $impact$-term on the ingoing, the first input parameters of these functions are identical. Taking the modeling of the two functions by SOS constraints into account this identity is expressed in the polytope

$$P_\Delta = \Big\{ \binom{\lambda_1}{\lambda_2} \in \mathbb{R}^{|\Lambda_1|+|\Lambda_2|} \mid \begin{array}{rcl} \sum\limits_{j\in\Lambda_1} \lambda_1^j \qquad\qquad & = & 1 \\ \sum\limits_{j\in\Lambda_2} \lambda_2^j & = & 1 \\ \sum\limits_{j\in\Lambda_1} p_1^j \lambda_1^j - \sum\limits_{j\in\Lambda_2} p_2^j \lambda_2^j & = & 0 \\ \lambda_1^j, \qquad \lambda_2^j & \geq & 0 \\ \lambda_1,\ \lambda_2 \text{ satisfy SOS Type 3 for } Y_1 \text{ and } Y_2 & & \end{array} \Big\},$$

where for ease of exposition we choose the subindex 1 for the first pipe and the subindex 2 for the second one. Furthermore, the notation of Chapter 3 is used, where Λ_1 and Λ_2 stand for the sets of grid points and $Y_1 = \{N_1^1, \ldots, N_1^d\}$ and $Y_2 = \{N_2^1, \ldots, N_2^d\}$ describe the triangulation with its simplices. In the formulation of this polytope, we see the linking of the SOS Type 3 conditions via the pressure equality. Note that the subdivision points of the first and second pipe, p_1^j and p_2^j, may differ, as they appear in different approximation grids.

In [Möl04] the linear description of $\operatorname{conv}(P_\Delta)$ was computed on small instances using `PORTA` [CL00], whereas the polytope P_Δ is based on a nonlinear function of the stationary gas model. But as the structure of the polytopes in the stationary and transient case are similar and the magnitude of the numbers are equal (in both cases pressure values are considered), we can transfer the results to our problem. [Möl04] describes that already for up to 30 grid points and simplices not only the number of facet-defining inequalities, but also the range of the coefficients increase dramatically (up to five digits). Therefore, it seems very difficult to give a complete linear description of the polytope in the general case.

Considering the linking polytope P_Δ, the computations with `PORTA` also indicate that the number of vertices grows moderately, precisely speaking quadratically, for larger instances. Let us motivate this fact. Observe that each vertex of $\operatorname{conv}(P_\Delta)$ must also be extreme in some $P_{ij} = \operatorname{conv}(P_\Delta) \cap \{\binom{\lambda_1}{\lambda_2} \in \mathbb{R}^{|\Lambda_1|+|\Lambda_2|} \mid \operatorname{supp}(\lambda_1) \subseteq N_1^i, \operatorname{supp}(\lambda_2) \subseteq N_2^j\}$, where $N_1^i \in Y_1$ and $N_2^j \in Y_2$ and $\operatorname{supp}(\lambda) := \{i \mid \lambda_i \neq 0\}$ denotes the support of λ, as the SOS Type 3 conditions have to be fulfilled. Such a polyhedron P_{ij} may be described by $|N_1^i| + |N_2^j|$ variables, three equations and $|N_1^i| + |N_2^j|$ nonnegativity constraints. Remember that for our example all simplices are triangles (as we linearized the two-dimensional $friction$- and $impact$-function), hence $|N_1^i| = |N_2^j| = 3$ holds for all i, j. Because of the SOS constraints, each vertex of P_{ij} must contain at least one positive λ in the set N_1^i and one in the set N_2^j, respectively. Altogether, at most three variables can be positive. If we determine all possible combinations, we observe that P_{ij} has at most 9 vertices, implying that the number vertices of polytope P_Δ is bounded by $9\,|Y_1|\,|Y_2|$. This observation is generalized in the following theorem [MMM06].

Theorem 6.4.1 *Consider some system of linear equalities $Ax = b$ with $A \in \mathbb{R}^{m\times n}$. Let $I \subseteq \{1,\dots,m\}$, $|I| = d$, be a set of rows which consists of SOS Type k constraints with $Y_i = \{N_i^1,\dots,N_i^{k(i)}\}$. Assume that each variable x_i appears in exactly one SOS Type k constraint and let*

$$P = \operatorname{conv}\{x \in \mathbb{R}^n \mid \begin{array}{l} Ax = b \\ x \textit{ satisfies the set condition for each } i \in I\}. \end{array}$$

If $s = \operatorname{rank} A$ and $r = \max_{i,l} |N_i^l|$, then the number of vertices of P is bounded by

$$\gamma(s,r,d)\prod_{i\in I} k(i),$$

where

$$\gamma(s,r,d) = \sum_{i=0}^{\lfloor d-\frac{c}{r}\rfloor} (-1)^i \binom{d}{i}\binom{(d-i)r}{c}$$

with $c = \min\{s, \lfloor \frac{n}{2}\rfloor\}$.

On the one hand this theorem covers the combination of SOS conditions of different dimension (for example necessary for coupling of pipe and compressor). On the other hand an arbitrary graph structure can be handled such as trees and cycles that might show up in a gas network. To illustrate this idea we consider some connected substructure of a graph on W nodes and F edges. For each edge (corresponding to a pipe or a compressor) we have one SOS constraint. We suppose that we can couple the SOS constraints via the pressure variables as in the above example. This results in $|\delta(v)| - 1$ coupling constraints per node $v \in W$, where $\delta(v)$ denotes the degree of node v. Taking the flow conservation (see Kirchhoff's law (5.18)) into account we receive $|\delta(v)|$ constraints per each node. Thus, $Ax = b$ in Theorem 6.4.1 consists of $m = 3|F|$ constraints including $d = |F|$ SOS conditions.

The observation that the number of vertices of polyhedron P_Δ grows quadratically, motivates the following separation algorithm developed in [Möl04]. Let $v_1,\dots,v_k$ be the vertices for P_Δ. Let $\bar\lambda$ be an optimal LP-solution to be cut off. We look for a violated cut of the form $a^T\lambda \le \alpha$ by solving the linear program

$$\begin{aligned} z^* = \max \quad & a^T\bar\lambda - \alpha \\ s.t. \quad & a^T v_i \le \alpha \quad \text{for all } i = 1,\dots,k. \end{aligned} \tag{6.1}$$

Observe that the feasible set of (6.1) is a polyhedral cone and thus the optimal solution value z^* is either zero (in which case there is no violated inequality) or the linear program is unbounded. To deal with this situation we normalize the set of solutions in the spirit of [BCC93]. Computational tests showed that the best choice is to restrict the coefficients $a_i \in [-1, 1]$.
Notice that this separation algorithm is only useful, if the number of vertices is moderate, since we must solve an LP to obtain a potential cut. Our observations above just indicate theoretical bounds for the number of vertices. [Möl04] determines concrete values for the number of vertices of polytope P_Δ for different instances. These investigations show that it is not useful to couple more than two approximation grids, hence linking more than two SOS constraints.

In comparison with the stationary case, here are much more possibilities to link approximation grids as our problem contains four kinds of nonlinear functions (in the stationary case there are just two, the fuel gas consumption of a compressor and a function describing the pressure loss in a pipe). Especially, we can even combine two linearization grids of the same pipe, if we consider the $conti$- and the $friction$-function as they both depend on the outgoing pressure of the pipe. In the example given above, we coupled two grids via the pressure variable. We can also link SOS constraints using flow variables, but this case is a little bit more complicated as flow variables are not nodal, but they depend on the segments (see also the description of the preprocessing methods above). Thus, we just can couple via flow values, if the intersection node of both segments has exactly one outgoing and one ingoing segment (the only exception is a compressor with its bypass valve). In that case, we can combine the corresponding flow variables because of Kirchhoff's law.

Chapter 7

A Primal Heuristic: Simulated Annealing

In this chapter we present a primal heuristic based on the idea of simulated annealing for our gas network optimization problem. The aim of the heuristic is to yield a good feasible solution for our TTO model in adequate runnning time, and thus to obtain a good upper bound for our branch-and-cut algorithm. The described algorithm was developed within the scope of a diploma thesis, see [Mah05, MMMar]. In the first section we introduce the general idea of the algorithm and give a mathematical description of it. Thereafter we adapt the simulated annealing algorithm for the problem of TTO. We conclude this chapter with an overview of computational results that can be found in [Mah05].

7.1 The Simulated Annealing Algorithm

7.1.1 The Idea

The simulated annealing (SA) algorithm is a meta-heuristic (see Section 2.3). It was originally developed for solving large combinatorial optimization problems and uses local search. In contrast to general local search algorithms (Algorithm 1 in Section 2.3), it randomly accepts solutions with increasing objective function value. Thus, SA can overcome local minima and the dependence on the initial solution is marginal, but it stays flexible and robust as local search algorithms [AK89].
The original idea of simulated annealing for solving large combinatorial optimization problems was independently introduced by [KJV83] and [Čer85]. As implied by the name, the simulated annealing algorithm is based on the analogy between the physical process of annealing liquids to the thermal equilibrium (solid phase) and the problem of finding the solution of combinatorial optimization problems. In the following we describe the annealing process and show this analogy

(see also [LA87, AK89, Ree93, Mah05]).

At the beginning the solid material is heated up until it melts. Because of the high temperature of the substance the molecules move in a highly disordered way. Cooling down slowly reduces the (kinetic) energy of the molecules and they arrange uniformly in a lattice structure. After reaching a certain temperature each molecule finds its ideal position and the substance changes into the solid phase.

If the initial temperature is high enough and if the cooling down is sufficiently slow, the molecules are optimally ordered and the energy of the resulting system is minimal. If the lowering of the temperature is too fast, the solid material does not reach its ground state and it will have several defects. For obtaining the optimal state, the substance must reach the thermal equilibrium at each temperature $\mathcal{T}$, before the temperature is again reduced. This means that the system must achieve the state of minimal energy for each temperature $\mathcal{T}$. A mathematical description of the thermal equilibrium yields the *Boltzmann distribution* (with random variable X)

$$P_{\mathcal{T}}(X = i) = \frac{1}{Z(\mathcal{T})} \exp\left(\frac{-E_i}{k_B \mathcal{T}}\right),$$

where E_i is the energy, $\mathcal{T}$ the temperature, k_B the Boltzmann constant and $Z(\mathcal{T})$ is a normalization factor (so called partition function). This distribution specifies the probability of the system being in state i with corresponding energy E_i at a certain temperature $\mathcal{T}$. We see that the more the temperature decreases the more probable are states with lower energy.

The authors of [MRR$^+$53] developed an algorithm that simulates the evolution of a substance with interacting molecules to the thermal equilibrium at a fixed temperature $\mathcal{T}$. This algorithm is known as *Metropolis algorithm* and uses Monte Carlo methods, see also [LA87, AK89]. It generates a sequence of states of the substance, where a state is defined by the position of the molecules. Starting from a state i with corresponding energy E_i a new state j with energy E_j is generated using a little perturbation, e.g. a small displacement of a randomly chosen particle. If $E_j - E_i \leq 0$, i.e., a state with lower energy was found, we move to the new state j and continue the iteration process. If the energy difference is greater than zero, we accept state j with probability $\exp\left(\frac{E_i - E_j}{k_B \mathcal{T}}\right)$. Thereby we randomly choose a number $\theta \in (0, 1)$, and if $\theta < \exp\left(\frac{E_i - E_j}{k_B \mathcal{T}}\right)$, we change to the new state j, otherwise we retain i. This acceptance rule is called *Metropolis criterion*. After numerous iterations of this perturbation mechanism, the probability distribution of the system states tends to the Boltzmann distribution. Hence, using this criterion the substance can reach its thermal equilibrium at given temperature $\mathcal{T}$.

Knowing the physical background it is easy to see the analogy between statistical mechanics and optimization problems (see also [Ree93]). A state of the substance with its energy corresponds to a feasible solution and its objective function value. Perturbation of a state stands for the generation of a neighbor solution. The temperature can be interpreted as a control parameter for the heuristic. Finally, the solid phase of the substance reflects the approximated solution of the problem.

7.1.2 The Algorithm

The simulated annealing algorithm is widely discussed in the literature, see for instance [LA87, AK89, Ree93, OK96, NW88]. It can be seen as a sequence of consecutively executed Metropolis algorithms (see Algorithm 4, step 5 to 12). After each iteration of the Metropolis algorithm the control parameter $\mathcal{T}$ is reduced. Here we give a description of the algorithm using the notation of problem (2.3) [Mah05].

Algorithm 4 Simulated Annealing

```
 1: Initialize control parameter $\mathcal{T}$
 2: Find initial solution $S$
 3: Calculate $c(S)$
 4: repeat
 5:    repeat
 6:       Randomly generate neighbor $S'$ of $S$
 7:       Set $\Delta = c(S') - c(S)$
 8:       Choose $\theta \in (0, 1)$ randomly
 9:       if $\Delta < 0$ or $\theta < e^{-\Delta/\mathcal{T}}$ then
10:          Set $S = S'$
11:       end if
12:    until Equilibrium criterion is fulfilled
13:    Decrement control parameter $\mathcal{T}$
14: until Specified stop criterion is fulfilled
15: Return the best solution found
```

The key point here is that the algorithm also accepts worse solutions with a certain probability which diminishes with decreasing $\mathcal{T}$ (see Algorithm 4, step 8 and 9). This stochastic aspect should avoid termination in local minima. Therefore, this algorithm is also called *stochastic hill-climber* [MF00].

A mathematical model of the SA algorithm can be given by *Markov chains*. By means of the theory of Markov chains theoretical (asymptotic) convergence can be proven [LA87, AK89]. Thus, the algorithm can be viewed as a global optimization algorithm, if an infinite number of transitions is allowed. Since this conclusion is not useful for practical applications, a finite time approximation must be developed [LA87, AK89]. In the literature finite time bounds for SA can also be found. The best bound known at present can be found in [NS00]. But there the number of necessary steps exceeds the cardinality of the solution space, hence it would be more reasonable to determine the optimal solution by complete enumeration.

Before implementing SA for the solution of a special problem, a lot of decisions must be made which can be divided into generic and problem specific ones [LA87, Ree93]. Generic decisions comprise parameters of the annealing algorithm itself. These are initialization and decrement of the control parameter $\mathcal{T}$, specification of the number of steps until the equilibrium condition is fulfilled, and a stopping criterion. Together these parameters build the so called *cooling schedule*. The second problem specific class includes the characterization of feasible solutions, specification of the cost function, generation of an initial solution, and the definition of the neighborhood structure.

As already mentioned, the presented SA algorithm is a method for solving combinatorial optimization problems. In that case it is often intuitive to specify a neighborhood structure. Because of its general and flexible form SA can easily be adapted to other optimization problems, since no special properties as for example differentiability or convexity are imposed on the objective function or the constraints.

Due to successful implementations of the SA algorithm in the field of combinatorial optimization, its practicability for problems with continuous variables was investigated. Approaches for global optimization in case of an n-dimensional function defined on a bounded subset can be found for instance in [BJS86, DA91]. Solution methods based on SA are developed and yield good results. [DA91] also proves convergence of the algorithm in analogy to the classical SA for combinatorial optimization problems.

[WW99] describes a modified SA algorithm to optimize functions with continuous variables subject to equality and inequality constraints. All constraints are relaxed by means of Lagrange multipliers and the resulting problem is solved using so called constrained simulated annealing. Note that if the optimization problem has numerous constraints one obtains a lot of additional variables. In [CMMR87] and [WC00] further modifications of SA can be found to tackle (constrained) global optimization of functions with continuous variables.

All these approaches provide a basis for solving the TTO problem. Note that our problem comprises numerous integer as well as continuous variables. Therefore, we have to combine different ideas of the cited literature and develop a new SA algorithm.

7.2 The Simulated Annealing Algorithm for TTO

To adapt the idea of SA to the TTO problem we have to make several problem specific and generic decisions. As already mentioned, generic decisions correspond to the cooling schedule which is similar for each application of SA. Problem specific aspects for our gas network optimization are the characterization of a solution and constraint-handling respectively, i.e., we follow the approach of [WW99] and relax some, but not all, of the constraints. Accordingly, we have to define a cost function for the relaxed conditions. A further crucial point is the specification of a neighborhood structure including a step size selection for the continuous variables. Finally we have to generate an initial solution.

7.2.1 Constraint-Handling and Cost Function

Let us begin with the handling of constraints. Basically there are two alternative approaches dealing with it (see [MF00]).

The first one only accepts feasible solutions. Hence, no time is wasted investigating infeasible solutions. A disadvantage is that in each iteration step a feasible neighbor must be found which might be difficult for complex problems.

The second approach also allows a transition to infeasible solutions. Thus, it becomes easier to determine a neighbor solution. Obviously the main disadvantage is that the solution space increases and a further handling of infeasibilities is needed in order to guarantee a feasible solution at the end of the algorithm.

In case of TTO it is very complicated to find feasible solutions, even the determination of an initial solution poses a challenge. Therefore, we follow the latter approach.

A common treatment for infeasible solutions is to introduce penalty terms. In [MF00] two well known methods are described: static penalty costs on the one hand and dynamic ones on the other hand. For both methods we need to define violation of constraints whereby we refer to the general form of a nonlinear optimization problem (2.2). For $j \in \{1, \ldots, m\}$, $v_j(x)$ indicates the (absolute) violation of the jth constraint with

$$v_j(x) = \begin{cases} |h_j(x)| & \text{if } 1 \leq j \leq q \\ \max\{0, g_j(x)\} & \text{if } q+1 \leq j \leq m \end{cases},$$

where $g_j(x)$ are the inequalities and $h_j(x)$ the equations.

The idea of the static penalty method is the following. For each constraint j we define l levels of violation and corresponding penalty coefficients R_{ij}, $i = 1, \cdots, l$. Then the penalty term can be written as

$$\sum_{j=1}^{m} R_{ij} v_j^2(x), \tag{7.1}$$

where index j refers to constraints and i to the respective level of violation. Note that such penalty costs can be calculated simply and fast. The disadvantage of this method is that numerous parameters ($m(2l+1)$, for each constraint j we need to determine the number of levels, l parameters for the boundaries defining these levels and l penalty coefficients R_{ij}) must be specified which might be very difficult to ensure feasibility of the solution.

To overcome the problem of determining good parameters, penalty functions with dynamic aspects are applied. Here the iteration step n is integrated in the penalty term. Hence, costs for infeasibilities increase dynamically during the execution of the algorithm. For the nth iteration the penalty function is given by

$$\sum_{j=1}^{m} (C_j n)^{\alpha} v_j^{\beta}(x), \tag{7.2}$$

where C_j, α and β are positive constants. For this method less parameters have to be determined. It turns out that both kinds of cost functions fail for our TTO problem. While testing the SA algorithm applying these two methods to our problem, there arise difficulties in finding a feasible solution. Therefore, a combination of both methods was developed (see [Mah05]). Besides the minimization of fuel consumption the main goal of our SA algorithm is the feasibility of the resulting solution. In our context we consider a solution to be feasible, if the maximal absolute violation of the relaxed constraints is less than a given accuracy ε and the remaining constraints are satisfied. The basic idea of the new, combined method is additional dynamic penalty of violations greater than ε. The penalty function for TTO is defined as

$$Q(x) = \sum_{j=1}^{m} P_j v_j^2(x), \tag{7.3}$$

where

$$P_j = \begin{cases} R_j & \text{if } v_j(x) \leq \varepsilon \\ R_j + (C_j n)^{\alpha} & \text{if } v_j(x) > \varepsilon. \end{cases}$$

Having presented different penalty functions, we apply this approach to our problem. Nevertheless, since determination of good parameters might be difficult, we neglect only some constraints which pose most problems to us. Therefore, we relax the discretized continuity and momentum equation for each pipe of the gas network, as basically they connect single time steps as well as flow and pressure variables. The remaining constraints have to be fulfilled in each iteration.

For a better understanding, in the following we abbreviate a solution (p, q, f, N, s) consisting of pressure, flow, fuel gas consumption, power, and switching variables by x. Introducing penalty functions $Q_C(x)$ and $Q_M(x)$ of the form (7.3) for the continuity and momentum equations of the pipes, we receive a relaxed problem which can be written in the following abbreviated form:

$$\min \quad f(x) + Q_C(x) + Q_M(x)$$

s.t. x satisfies (7.4)

- lower and upper bounds for pressure in nodes
- lower and upper bounds for flow in segments
- flow balance in nodes (Kirchhoff's first law)
- supplier and consumer behavior for sources and sinks
- compressor constraints
- switching constraints for valves and compressors
- terminal condition for the last time step,

where $f(x)$ denotes the original objective function, see Section 5.5.

7.2.2 Neighborhood Structure

We continue with an important part of the algorithm, the specification of a suitable neighborhood structure which has to guarantee a powerful search. As there are continuous and integer variables in case of the TTO problem, the definition of an adequate neighborhood structure is not as intuitive as for combinatorial problems.

Due to the relaxed continuity and momentum equations the flow and pressure variables as well as the single time steps (except the first and last time step because of the terminal condition, see (5.36)) are decoupled. Thus, each time step can be treated separately, and flow and pressure variables can be determined independently for each time step. Hence, the key idea of the neighborhood generation is a small perturbation of flow or pressure variables. At each iteration we randomly choose a time step $t \in \mathbb{T} \setminus \{1\}$ and either a segment $e = vw \in E$ or a node $v \in V$. Then the flow variable q_e^t, $q_{e,v}^t$ or $q_{e,w}^t$ (depending if the chosen segment is a pipe) or the pressure variable p_v^t is altered, yielding a flow or pressure neighbor. Note that the change of such a variable often results in a violation of constraints in the same time step t. In order to generate a feasible neighbor for the relaxed problem (7.4), we use so called *repair procedures* in order to adapt the neighbor to the violated constraints or to return that the generated neighbor is infeasible.

So we define two kinds of neighborhoods for a given solution x, the flow $N_{flow}(x)$ and the pressure neighborhood $N_{pressure}(x)$ and denote x' to be a neighbor of x if $x' \in N(x) := N_{pressure}(x) \cup N_{flow}(x)$. In the following we describe in detail how to generate an element of $N_{flow}(x)$. As the generation idea is similar in case of a pressure neighbor, we only give a brief description of it.

For generating a flow neighbor, we randomly select a segment $e = vw \in E$ and a time step $t \in \mathbb{T} \setminus \{1\}$. In case of a pipe we additionally randomly choose between the beginning and end of it, since it has two flow variables. Then we modify the corresponding flow variable q_e^t, $q_{e,v}^t$ or $q_{e,w}^t$ by increasing or decreasing its value by Δq_e^t, where Δq_e^t denotes the step size (for the step size selection see below). In case of a flow capacity violation of the segment, the generation terminates and returns “infeasible”.

After changing the flow variable, just the flow balance equations (5.18) in the initial node v and the end node w of segment e at time t can be violated. Therefore, we need two repair procedures to handle these infeasibilities. The first procedure is called *adjustFlowBackward()* and recursively adapts the flow variables corresponding to v in reverse flow direction. Similarly the second one *adjustFlowForward()* iteratively adjusts the flow variables concerning node w in flow direction.

The idea of these procedures is based on the specific properties of each segment type. Basically the segments can be divided into two groups: the set of *free* segments and the set of *fixed* segments. A segment belongs to the first group, if a modification of the flow variable at the beginning of the segment does not effect the flow variable at the end of it and reversely. All other segments are called fixed. The assignment of the segments depends on the specific constraints. Compressors, valves, control valves and connections are fixed segments as they have one flow variable, whereas pipes are free segments since they have flow variables for beginning and end, and these variables are decoupled because of the relaxed continuity and momentum equations.

The *adjustFlowBackward()* procedure works as follows. First it checks whether v is a source. If this is the case, the procedure terminates. Otherwise it randomly chooses an ingoing segment $e' \in \delta^-(v)$ of v. If $e' = uv$ is a fixed segment, the corresponding flow variable is modified such that (5.18) is fulfilled in node v, and the procedure is recursively applied to node u to continue adaption in reverse flow direction. In case of a free segment, hence a pipe, the flow variable $q_{e',v}^t$

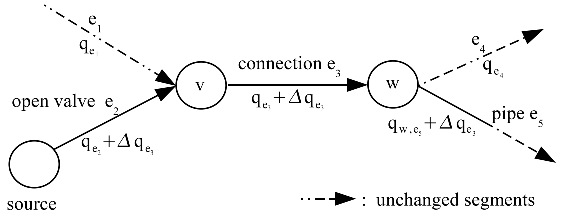

Figure 7.1: Generation of flow neighbor

is adjusted and the procedure terminates. Unless no capacity constraints such as flow boundaries or supplier conditions are violated, the procedure returns “feasible”. The *adjustFlowForward()* operates analogously in flow direction.

Let us illustrate the flow neighbor generation by an example (see Figure 7.1). Since we consider a fixed, randomly selected time step t, we omit this index in the following. We randomly choose segment $e_3 \in E$ which is a connection. We alter its flow variable q_{e_3} by Δq_{e_3} and assume that no flow bounds are violated. Now the *adjustFlowBackward()* procedure is called to guarantee flow balance in node v. This procedure randomly selects the open valve e_2 from the set of ingoing edges of v and changes the corresponding flow variable q_{e_2} by Δq_{e_3}. As a valve is a fixed segment, recursion begins. Next, the initial node of valve e_2 is considered. Since it is a source, the *adjustFlowBackward()* procedure terminates successfully, assuming that no supplier condition is violated. After this, the neighbor generation continues in flow direction with the *adjustFlowForward()* procedure for node w. The outgoing segment e_5 is selected. Since e_5 is a pipe, hence a free segment, the variable $q_{e_5,w}$ is adjusted and the procedure stops. Supposing that all capacity constraints are fulfilled, a feasible flow neighbor was generated.

Now we come to the pressure neighbors in $N_{pressure}(x)$. The main differences to the flow neighbor generation are that pressure variables are nodal variables and that pressure variations must be adapted independently of the flow direction. A node $v \in V$ and a time step $t \in \mathbb{T} \setminus \{1\}$ are randomly selected. After this, the pressure variable p_v^t is changed by Δp_v. The resulting constraint violations of the relaxed problem (7.4) are corrected by the *adjustPressure()* procedure. Note that we need just one repair procedure, since pressure changes direction independently. Again, we have to divide the set of segments into fixed and free elements, but this time with respect to pressure. Hence, open valves and connections are fixed segments as they require pressure equality at beginning and end. Whereas pipes, compressors, control valves, and closed valves are free segments, since they can handle pressure differences. Now, the *adjustPressure()* procedure is called recursively for every fixed segment incident with node v to guarantee feasibility of the generated neighbor.

Moreover, we have to pay attention to the controllable elements valves/control valves, and compressors. The algorithm provides the possibility of closing or opening these operable segments. If during the neighbor generation the fuel gas consumption of an operating compressor falls below a certain lower bound, the machine can be shut down with a certain probability and with respect to switching constraints. Accordingly, the same holds for an open valve, if the corresponding flow variable reaches a certain minimal value. On the other hand, a compressor or a valve can be opened, if its associated flow variable is increased during the neighborhood generation.

To conclude the neighbor structure, we have to dwell on step size selection, as the generation mechanism works on continuous variables. The choice of the step size has a significant effect on the accuracy of the solution. If the step size is too small, the algorithm needs a very long time to approximate the optimal solution. Otherwise, if the step size is chosen too large, it is difficult to obtain an exact solution. Thus, using a fixed step size Δx_i for each variable results in a compromise between solution accuracy and time. To overcome this problem adaptive step size selection was developed, see for example [MHF03]. In this approach a neighbor range R_i is specified for each variable x_i depending on the problem. Then the adaptive step size Δx_i is determined via the following equation

$$\Delta x_i = r \cdot R_i, \quad (-1 \leq r \leq 1), \tag{7.5}$$

where r denotes a random variable. In the case of TTO we use a uniform distribution for the random variable.
Basically we differentiate between the step sizes Δq and Δp for flow and pressure variables. To guarantee that the change of the cost function is of the same order of magnitude for the generation of a flow and a pressure neighbor, the pressure variation Δp must be less than Δq by a factor of 100. Then a neighbor of $N_{pressure}(x)$ is accepted with the same probability as a neighbor of $N_{flow}(x)$ (see [Mah05]).

7.2.3 Initial Solution

The last aspect of the specific decisions for the SA algorithm is the generation of an initial feasible solution for the relaxed problem (7.4) from which the search starts. For its generation we use the given initial state of the gas network, i.e., the values of all flow and pressure variables at the beginning of the planning horizon for $t = 1$. This state is multiplied to the other time steps by assigning the variable values to the corresponding variables at time steps $t \in \mathbb{T} \setminus \{1\}$. Thus, the resulting solution fulfills the stationary constraints (see Section 5.2) and only transient conditions are violated. Note that because of the relaxed continuity and momentum equations, violated transient constraints are in principle supplier and consumer behavior and fixations of controllable elements. By means of the repair procedures of the neighborhood generation such violations can be corrected.

7.2.4 Cooling Schedule

In this section we have a look at the generic decisions for the SA algorithm. As already mentioned, they build the cooling schedule and comprise the initial value and the decrement function of the control parameter $\mathcal{T}$, the specification of a finite number of iterations for each value of the control parameter, and a stop criterion. All these settings control the acceptance of worse solutions during the algorithm. A lot of literature can be found concerning the determination of an efficient cooling schedule, whereby we concentrate on [LA87, DA91]. In the following we present the cooling schedule developed for the problem of TTO.

First we concentrate on the *initial value* $\mathcal{T}_0$ of the control parameter $\mathcal{T}$. This value should be sufficiently large so that at the beginning almost all transitions are accepted, i.e., the ratio χ_0 between the number of accepted transitions and the number of generated ones is close to 1. We follow [DA91] who proposes the following empirical rule. Suppose a given number of m_0 trials is randomly generated under the assumption that all are accepted. Let m_1 and m_2 denote the number of trials with decreasing and increasing cost function ($m_1 + m_2 = m_0$). Further on, let $\overline{\Delta} f^+$ be the average increase of costs, hence the average change of the cost function for transitions that are considered in m_2. The initial value is calculated with the expression

$$\mathcal{T}_0 = \bar{\Delta} f^+ \left(ln \left(\frac{m_2}{m_2 \chi_0 + (1 - \chi_0) m_1} \right) \right)^{-1}, \tag{7.6}$$

where $\chi_0 \in (0, 1)$ is a given acceptance ratio.

Now we outline the *number* L *of iterations* for each value of the control parameter $\mathcal{T}$. This parameter specifies how carefully the search space is explored. It should be large enough to allow investigation of the neighborhood of a given solution in all directions. In [DA91], L is determined dependent on the problem dimension n via

$$L = L_0 \cdot n$$

with a constant L_0, the so called standard length. Note that this definition leads to a constant number for a given problem instance. The dimension of the TTO problem is in principle influenced by the size of the considered time horizon. Hence we choose L proportional to the number of time steps T with $L_0 = 500$.

We continue with the *decrement function* of the control parameter $\mathcal{T}$. We tested two alternative approaches for the problem of TTO, the geometric and the adaptive one. The geometric function is a commonly used decrement rule. It simply requires a constant $\alpha \in (0, 1)$ and the reduced control parameter is calculated with the following formula

$$\mathcal{T}_k = \alpha \mathcal{T}_{k-1}, \tag{7.7}$$

where $\mathcal{T}_k$ stands for the value of the control parameter in the k-th inner loop (see [LA87]).

The second approach is presented in [AdBHvL86]. This adaptive method uses informations gathered during execution of the algorithm in order to obtain an optimal cooling of the control parameter $\mathcal{T}$. Therefore, the reduction multiplicator is varied in each iteration step and depends on the standard deviation $\sigma(\mathcal{T}_{k-1})$ of the objective function in the previous inner loop. The decremented value $\mathcal{T}_k$ is given by

$$\mathcal{T}_k = \mathcal{T}_{k-1} \left(1 + \frac{ln(1+\delta)\,\mathcal{T}_{k-1}}{3\sigma(\mathcal{T}_{k-1})} \right)^{-1}, \tag{7.8}$$

where δ is a parameter which influences the velocity of reduction of the control parameter.
Finally we consider the *stop criterion* for the algorithm. One termination criterion is that there is no significant improvement of the objective function for a fixed number of iterations, another is that the algorithm terminates if the control parameter reaches a predefined value (see [LA87]). In case of the TTO problem, we combined these two criteria, where we use $\bar{\mathcal{T}} = 0.001$ and $n = 100000$. Additionally the algorithm stops if the maximum absolute violation of the relaxed constraints is less than a given accuracy value ε. Note that in the latter case we find a feasible solution, since we allow minor violations of the discretized continuity and momentum equations.
This completes the description of our SA algorithm for gas network optimization.

7.2.5 Solution for the Linearized Problem

The SA algorithm described above gives a solution for the mixed integer nonlinear formulation of the TTO problem. Hence, it works on the exact discretized continuity and momentum equations (4.5) and (4.6) for the pipes and the fuel consumption function (4.11) for the compressors, and does not consider any approximation model. If we want to integrate the SA heuristic in our branch-and-cut algorithm, we have to determine a solution for the linearized problem. This adaptation is straightforward as the nonlinear and the linear problem only differ slightly from each other. The main difference occures in those parts of the algorithm where the nonlinear components of the model appear. Let us regard for example the fuel consumption (4.11) of a compressor $e = vw \in E_c$. We know that it is a three-dimensional function $f(p_v, p_w, q_e)$ of pressure p_v at the beginning of the compressor, of pressure p_w generated by it and of flow q_e through it. We also remember that we have a three-dimensional grid and the function is linearized within each tetrahedron by means of the convex combination of the exact function values at the vertices of the tetrahedron (see Section 5.2.5). For determining the linearized function value in a point (p_v, p_w, q_e) which specifies the current state of the compressor, we have to find the tetrahedron in which this point lies. Then the point (p_v, p_w, q_e) can be written as the convex combination of the vertices of this tetrahedron by means of the λ-variables. The values of the λ-variables are given by the solution of a linear system of equations. Finally the linearized function value can be calculated with the help of these λ-values as the convex combination of the exact function values in the grid points.
Accordingly for a general grid of arbitrary dimension, we have to determine the simplex in which the point lies where the nonlinear function is to be approximated. The λ-values can be calculated analogously and we obtain the linearized function value by the corresponding convex combination. Note that besides the fuel gas consumption of the compressors, we have a nonlinear

one-dimensional term in the discretized continuity equation and two two-dimensional terms in the discretized momentum equation. Altogether we receive a solution for the linearized model of TTO and thus, an upper bound for our branch-and-cut algorithm.

7.3 Computational Results

In [Mah05] the SA algorithm is tested considering three gas networks (see networks in Section 10.1) over different time horizons. At first, alternative approaches are compared and diverse parameter settings are tried. This is done by means of the smallest test network to obtain a general calibration of the algorithm. After this, varying time horizons for all test networks and fixations of compressors are considered. In the following we summarize the computational results.

For the initial value $\mathcal{T}_0$ we use the formula given by [DA91], see (7.6). The number of iterations L for each value of the control parameter $\mathcal{T}$ is defined depending on the considered time horizon T, the dimension of the TTO problem. It turns out that $L = 500 \cdot (T - 1)$ provides a good value.

First of all, the fixed step size selection is compared with the adaptive one. Test runs show that the optimal value for the fixed step size must be determined experimentally for each tolerance limit ε. Otherwise it could happen that the algorithm fails and does not yield any feasible solution. The adaptive step size approach is more flexible for different accuracy values ε. Moreover, it always finds a feasible solution and the running time is better. Therefore we integrate the adaptive step size selection because of its flexibility and speed.

Thereafter the different penalty cost approaches for the relaxed constraints are compared, static costs (7.1), dynamic costs (7.2), and the combined method (7.3). For the dynamic penalties no parameter settings could be specified to find a feasible solution. Hence, this method does not seem to be applicable for the problem of TTO. For small planning horizons, e.g. $T = 3$, there is no difference between the static and the combined cost function. But with increasing complexity, the static method fails as it cannot determine any feasible solution. So we choose the combined penalty approach since it certainly gives a feasible solution. This alternative was especially developed for the TTO problem where $R_j = 10$ and $C_j = 0.0005$ for all j and $\alpha = 2$ are good parameter values.

Then the geometric and the adaptive decrement function (7.7) and (7.8), respectively, of the control parameter are compared. For small time horizons the geometric method is considerably faster with decreasing α than the adaptive one. But at the same time the probability decreases that it finds a feasible solution. In case of greater values of T the adaptive decrement is more reliable and faster. Thus, we take the adaptive decrement function where $\delta = 20$ is a good parameter choice.

For concluding the calibration of the parameters we again have a look at the step size selection. We have to specify step size ranges R_q and R_p for the adaptive method applicable for different accuracy parameters ε. As already mentioned, R_p must be less than R_q by the factor of 100. Hence, different values of ε are considered while varying the neighbor ranges R_q and R_p. In contrast to minor differences of the objective function value between several test runs, the solution time varies considerably. Again the flexibility of the adaptive step size method with respect to changing accuracies can be observed. For all considered values of ε, the best results are obtained using $R_q = 5$ and $R_p = 0.05$. Besides, we remark that there is a clear dependency between time and accuracy. With decreasing ε the running time increases moderately.

After determination of the parameter settings, the three networks are tested over different time horizons T with accuracy $\varepsilon = 0.1$. Note that for the third network we have to decrease the distance parameter δ of the adaptive decrement function to 5 in order to obtain a feasible solution. Beyond this, no parameter changes are necessary. The algorithm yields feasible solutions of the problem of TTO for all considered examples. Naturally a notable raise of running time can be observed with increasing T.

For the first test network, the smallest one, the running times range between 6 seconds for $T = 3$ and 5 minutes for 24 time steps. These results are very good.

For the second network the running times are considerably longer even though the complexity of these test instance increases only slightly. Here the times varies from 15 seconds for 3 time steps to about 16 minutes for a day. Amongst others this is due to good initial solutions generated for the SA algorithm in case of the first network. The initial solutions generated for the middle network do not give feasible settings of the controllable segments, thus at the beginning, the algorithm has to test different states of the compressors which increases the running time.

Surprisingly, we receive the best solution times for the biggest network, the third one. Again this can be attributed to good initial solutions for SA. Here the algorithm needs 4 seconds for 3 and 77 seconds for 24 time steps.

Altogether, the SA algorithm for TTO is able to find feasible solutions considering up to 24 time steps in very fast running times.

To summarize, we presented a SA algorithm tailored for TTO. Since this problem comprises integer as well as continuous variables, we follow the relaxation approach for some constraints. To decouple time steps as well as pressure and flow variables we relaxed the discretized continuity and momentum equations and integrated them by means of an especially developed penalty term in the objective function. For the local search we defined flow and pressure neighborhoods for the relaxed problem by the perturbation of variables. An adequate cooling schedule was found according to the literature.

The tuning of the methods and parameters was very extensive. But altogether we obtain a flexible algorithm which does not use special properties of the problem instance. Moreover, it is robust against modifications of the basic conditions. The resulting SA algorithm yields feasible solutions in very good running times where only marginal changes of parameters are necessary.

Chapter 8

Switching Polytopes

In this chapter we consider 0/1 polytopes - which we call *switching polytopes* - that are defined by modeling minimum runtime and downtime and of switching processes for a machine. Such conditions appear for compressors in our transient gas network optimization.
At first we state the mathematical formulation of these conditions. After this we give a survey on related literature. Finally we summarize the studies of these 0/1 polytopes that were developed within the scope of a diploma thesis and can be found in [Mar05]. For the computation of the linear description of the switching polytopes for small instances we used `PORTA` [CL00].

8.1 Mathematical Formulation

As already mentioned in Chapter 5, min-up and min-down conditions and switching processes (for consideration of constant start-up and shut-down costs) are modeled via

$$x^t - x^{t-1} \leq x^j \qquad \text{for} \quad 2 \leq t < j \leq \min\{t + L - 1, T\} \tag{8.1}$$

$$x^{t-1} - x^t \leq 1 - x^j \qquad \text{for} \quad 2 \leq t < j \leq \min\{t + l - 1, T\} \tag{8.2}$$

$$x^t - x^{t-1} - x^t_{up} + x^t_{down} = 0 \qquad \text{for} \quad t = 2, \dots, T \tag{8.3}$$

$$x^t_{up} + x^t_{down} \leq 1 \qquad \text{for} \quad t = 2, \dots, T. \tag{8.4}$$

Here $T \in \mathbb{N}$ denotes the number of time steps (in our case hours), into which the planning horizon is divided. $L \in \mathbb{N}$ is the min-up time for the machine, i.e., when the machine is switched on it must be on for at least L time steps. Similarly $l \in \mathbb{N}$ is the min-down time of the machine. The binary variable x^t indicates if the machine is operating ($= 1$) in time period t or not ($= 0$). Finally the binary variable x^t_{up} or x^t_{down} equals 1 if and only if the machine is switched on or off in time period t, respectively. The inequalities (8.1) and (8.2) assure the min-up and min-down times

of the machine. Whereas conditions (8.3) and (8.4) guarantee the correct settings of the up- and down-variables.
Note that in this chapter we use a minor modification for the denotations. In Chapter 5, we indicate by s^t_e, $s^t_{e,up}$ and $s^t_{e,down}$ the switching, start-up and shut-down variable of a compressor $e \in E_C$, respectively. Here, we choose the letter 'x' instead of 's' and neglect the compressor index e.

8.2 Literature Survey

Besides in gas network optimization, minimum runtime and downtime conditions for machines can be found in other applications as for example power production or regenerative energy (see [GNRS00, NNR$^+$00, AC00, TKW00, HNNS04]). In the following we give a detailed description of the article [LLM04] of Lee, Leung and Margot. In this paper, $P_T(L,l)$ is defined as the convex hull (in $\mathbb{R}^T$) of the 0/1 solutions of (8.1) and (8.2). The authors investigate these 0/1 polytopes - so called *min-up/min-down polytopes* - defined by the min-up and min-down conditions for a machine. There the following classes of inequalities are defined.

Definition 8.2.1 *For a nonnegative integer* k*, consider a nonempty set of* $2k+1$ *indices from the discrete interval* $[1,T]$ *with* $\phi(1) < \psi(1) < \phi(2) < \psi(2) < \ldots < \phi(k) < \psi(k) < \phi(k+1)$ *such that* $\phi(k+1) - \phi(1) \leq L$*. With these indices the* alternating up inequality

$$-\sum_{j=1}^{k+1} x^{\phi(j)} + \sum_{j=1}^{k} x^{\psi(j)} \leq 0 \tag{8.5}$$

is associated.
By requiring $\phi(k+1) - \phi(1) \leq l$*, the* alternating down inequality

$$\sum_{j=1}^{k+1} x^{\phi(j)} - \sum_{j=1}^{k} x^{\psi(j)} \leq 1 \tag{8.6}$$

is defined.

Note that inequalities (8.1) and (8.2) and the simple lower bound inequalities $-x^j \leq 0$ and upper bounds $x^j \leq 1$ are alternating up or down inequalities. Notice the symmetry in these inequalities by switching the roles of L and l and complementing the variables x^t (i.e., replacing x^t by $1 - x^t$). Validity of these inequalities for $P_T(L,l)$ is checked by considering all feasible 0/1 vectors that fulfill the min-up and min-down conditions and inserting these vectors in the corresponding inequality.
Moreover, the alternating up and down inequalities describe facets of $P_T(L,l)$. This is shown by

specifying T affinely independent points that are tight for the respective inequality. The main result of [LLM04] is that the alternating up and down inequalities provide a complete linear inequality description of $P_T(L, l)$. The idea of the corresponding proof is that any point, satisfying all of the alternating up and down inequalities, can be expressed as a convex combination of extreme points of $P_T(L, l)$. The authors also indicate a separation algorithm yielding (for a given $x \in \mathbb{R}^T$) a maximally violated inequality (if any) of the form (8.5) and (8.6) in running time $O(T)$.

Now we come to switching costs. In applications, if switching costs are taken into account, start-up costs for machines are always considered whereas shut-down costs are mostly neglected. In the literature two models of start-up costs are distinguished, constant and down-time dependent start-up costs.

First we look at the basic model of constant start-up costs, i.e., down-time independent, see for example [GNRS00, NNR$^+$00]. For time step t, these costs can be expressed by

$$C_{up}^t \max\{x^t - x^{t-1}, 0\},$$

where C_{up}^t are the positive and constant start-up costs for a machine at this time. Note that the maximum is one, if and only if $x^t = 1$ and $x^{t-1} = 0$. If we want to integrate this cost factor in a MIP, we have to linearize it. Therefore we need the additional variable x_{up}^t and the further constraints

$$x_{up}^t \geq x^t - x^{t-1} \tag{8.7}$$

$$x_{up}^t \leq x^t \tag{8.8}$$

$$x_{up}^t \leq 1 - x^{t-1} \tag{8.9}$$

which assure the right setting of the variables (see [PW06]). The variables x_{up}^t can only be equal to one, if the machine is operating in time step t (see inequality (8.8)) and not operating in time step $t - 1$ (see inequality (8.9)). If both states occur simultaneously, inequality (8.7) forces x_{up}^t to one. Notice that the inequalities above linearize the equation $x_{up}^t = x^t(1 - x^{t-1})$. In [Mar05] polyhedral studies of the polytope defined by (8.1), (8.2) and (8.7) to (8.9) can be found. Facet-defining inequalities and an appropriate polynomial separation algorithm are specified, which is tested by means of compressors in a gas network.

Now we come to an extended approach where the start-up costs depend on the preceding down time of the machine. The longer the time that the machine was shut down the higher are the start-up costs. If the down time of the machine exceeds a certain bound, the costs are assumed to be constant, since this necessitates a so called cold start of the machine. This results in piece-wise constant start-up costs.

For time step t they can be modeled by (see [GNRS00])

$$\max_{\tau=0,\ldots,\tau_c} \left\{ C_\tau (x^t - \sum_{k=1}^{\min\{\tau,t-1\}} x^{t-k}) \right\},$$

where C_τ are the constant costs if the compressor is turned on and has been switched off the τ preceding time steps. These cost coefficients are in ascending order with $C_0 = 0$ to ensure non-negative start-up costs for time step t. The compressor needs τ_c time steps to cool down, hence C_{τ_c} are the costs for a cold start. The term $x^t - \sum_{k=1}^{\min\{\tau,t-1\}} x^{t-k}$ is bounded from above by 1. For $\tau > 0$ it equals 1 if and only if the machine is operating at time t ($x^t = 1$) and has been off-line the $\min\{\tau, t-1\}$ preceding time steps ($x^{t-1} = \ldots = x^{\max\{1,t-\tau\}} = 0$).

In [AC00] a linearization of these time-dependent start-up costs can be found. This formulation needs additional binary and integer variables and constraints resulting in a complex model.

The results presented in this chapter were independently developed in [Mar05] and [RT05] where polyhedral studies concerning minimum up and down constraints and start-up and shut-down costs can be found. In the paper [RT05] of Rajan and Takriti the down-variables x^t_{down} are eliminated from the model via equations (8.3) and the resulting polytope is investigated. Taking into account this variation, the authors obtain the same set of inequalities as we give in Lemma 8.3.2. They prove validity of these inequalities and facet-defining property with the same arguments as we do (see [Mar05] and Section 8.3.1). The polyhedral studies differ in the proof of the main result, i.e., that the given constraints yield a linear inequality description of the polytope. Following the idea of [LLM04], the authors of [RT05] show integrality of the resulting polytope by proving inductively that any point in it can be written as a convex combination of integral elements. Whereas we demonstrate that the system of linear inequalities, including the down-variables, is totally dual integral. [RT05] tested the developed separation algorithm within the scope of a branch-and-cut algorithm for the unit commitment problem and show the benefit of it. It is clear that the inequalities resulting from runtime and switching conditions dominate the alternating up/down inequalities defining the min-up/min-down polytope. Moreover [RT05] point out that the alternating up/down inequalities describe a projection of the switching polytope if all up- and down-variables are eliminated.

8.3 Investigation of the Switching Polytopes

In this section we investigate the switching polytopes. As mentioned above these polytopes are connected with runtime conditions and constant switching costs of compressors in our transient optimization.

8.3.1 Polyhedral Studies

For short we define

$$(x, x_{up}, x_{down})^\top := ((x^1, \ldots, x^T), (x^2_{up}, \ldots, x^T_{up}), (x^2_{down}, \ldots, x^T_{down})^\top.$$

We consider the switching polytope $P^{switch}_{T(L,l)}$ which is defined as the convex hull of all feasible solutions of inequalities (8.1) to (8.4), that is

$$P^{switch}_{T(L,l)} := \text{conv}\{(x, x_{up}, x_{down})^\top \in \{0,1\}^{3T-2} \mid (x, x_{up}, x_{down})^\top \text{ satisfies (8.1) - (8.4)}\},$$

where $T \in \mathbb{N}$ is the number of time steps of the planning horizon and $L, l \in \mathbb{N}$ are the min-up and min-down times of the machine.

In the following we study this switching polytope. Detailed proofs of the statements can be found in [Mar05], whereas we just sketch the ideas.

At first we have a look at the vertices of the polytope. Obviously all vertices are 0/1-vectors. Since a vertex must fulfill the minimum runtime condition, a sequence of 1-entries in the first subvector x must have a length of at least L, except at the beginning and end of the horizon where shorter 1-sequences are allowed. Likewise the minimum downtime condition is assured for a vertex if a zero sequence (in the first subvector) has a length of at least l, except at the beginning and end of the horizon. The subvector x_{up} of a vertex can have a positive entry if and only if the corresponding state variable equals 1 and the preceding one is 0. Conversely, x^t_{down} is positive if and only if the corresponding state variable is 0 and the preceding one has value 1. Trivial vertices are

$$e := ((1,\dots,1),(0,\dots,0),(0,\dots,0))^\top \quad \text{and} \quad z := ((0,\dots,0),(0,\dots,0),(0,\dots,0))^\top,$$

where the machine is on resp. off during the whole time horizon. Further vertices with simple structure are given by solutions a_j where the compressor is operating at the beginning of the planning horizon and it is switched off in time step $j \in \{2,\dots,T\}$. Note that for a_j the subvector x_{up} has no positive entry and exactly one of the down variables - namely x^j_{down} - equals 1. Similarly, we have vertices b_j where the machine is initially off and it is turned on at time $j \in \{2,\dots,T\}$. Here $x_{down} = 0$ and just x^j_{up} of the up-variables is positive.

By means of these vertices, we can determine the dimension of $P^{switch}_{T(L,l)}$.

Lemma 8.3.1 *dim* $(P^{switch}_{T(L,l)}) = 2T - 1$.

Proof. Because of equations (8.3) $P^{switch}_{T(L,l)}$ has dimension at most $2T-1$. Now we choose the following $2T$ vertices

$$e, z \quad \text{and} \quad a_j, b_j \text{ with } j \in \{2,\dots,T\}.$$

It is easy to show that these points are affinely independent, which implies the statement. □

In the following lemma we present inequalities that are valid for $P^{switch}_{T(L,l)}$.

Lemma 8.3.2 *The following inequalities are valid for* $P^{switch}_{T(L,l)}$.

(i) The nonnegativity constraints

$$x^i_{up} \geq 0 \qquad \textit{for} \quad i = 2,\dots,T, \tag{8.10}$$

$$x^i_{down} \geq 0 \qquad \textit{for} \quad i = 2,\dots,T. \tag{8.11}$$

(ii) For $T - L + 1 \geq 2$

$$-x^T + \sum_{k=i}^{T} x_{up}^k - \sum_{k=i+L}^{T} x_{down}^k \leq 0 \qquad \textit{for} \quad i = 2, \dots, T - L + 1 \tag{8.12}$$

and if $T - L + 1 < 2$ *these inequalities reduces to*

$$-x^T + \sum_{k=2}^{T} x_{up}^k \leq 0.$$

(iii) For $T - l + 1 \geq 2$

$$x^T - \sum_{k=i+l}^{T} x_{up}^k + \sum_{k=i}^{T} x_{down}^k \leq 1 \qquad \textit{for} \quad i = 2, \dots, T - l + 1 \tag{8.13}$$

and if $T - l + 1 < 2$ *these inequalities reduces to*

$$x^T + \sum_{k=2}^{T} x_{down}^k \leq 1.$$

Proof. We just explain the validity of (8.12) in case $T - L + 1 \geq 2$. The idea can be adjusted for (8.13) and the proofs for all other inequalities are trivial.
Let $i \in \{2, \dots, T - L + 1\}$ and let $(x, x_{up}, x_{down})^\top$ be a vertex of $P_{T(L,l)}^{switch}$. Because of the minimum runtime L the value of the sum $\sum_{k=i}^{T} x_{up}^k$ exceeds the value of $\sum_{k=i+L}^{T} x_{down}^k$ by at most one. If $\sum_{k=i}^{T} x_{up}^k \leq \sum_{k=i+L}^{T} x_{down}^k$ the point $(x, x_{up}, x_{down})^\top$ fulfills inequality (8.12). In case of $\sum_{k=i}^{T} x_{up}^k > \sum_{k=i+L}^{T} x_{down}^k$ the machine must be on in the last time step T, hence the left side of the inequality equals zero. □

As in the case of the alternating up and down inequalities for the min-up/min-down polytope, we can detect a certain symmetry. Complementing the variable x^T, switching the roles of L and l and interchanging the variables x_{up}^k and x_{down}^k maps (8.13) to (8.12).
Having the validity of these inequalities, we can also show that these inequalities induce facets of the switching polytope.

Lemma 8.3.3 *The inequalities* (8.10), (8.11), (8.12) *and* (8.13) *describe facets of the polytope* $P_{T(L,l)}^{switch}$.

Proof. For each kind of inequality, we can specify $2T - 1$ affinely independent points that satisfy the corresponding inequality at equality.

For inequality (8.10) we choose a_j for $j \in \{2, \ldots, T\}$, b_j for $j \in \{2, \ldots, T\} \setminus \{i\}$, and the points e and z.
For inequality (8.11) we can take a_j for $j \in \{2, \ldots, T\} \setminus \{i\}$, b_j for $j \in \{2, \ldots, T\}$, e and z.
In case of inequality (8.12) the situation is a little bit more complicated. We have a_j for $j \in \{2, \ldots, i + L - 1\}$, b_j for $j \in \{i, \ldots, T\}$ and z. Beyond those we choose the following $T - L - 1$ points, where the machine is shut down at the beginning and at the end of the planning horizon and it is switched on at time $j \in \{2, \ldots T - L\}$ for exactly L time steps.
For inequality (8.13) we take a_j for $j \in \{i, \ldots, T\}$, b_j for $j \in \{2, \ldots, i + l - 1\}$ and e from the simple vertices. Furthermore we have $T - l - 1$ points, where the compressor is on at beginning and end of the horizon and it is switched off at time $j \in \{2, \ldots T - l\}$ for exactly l time steps.
The cases $T - L + 1 < 2$ and $T - l + 1 < 2$ are easy to handle and are neglected here. □

In the next step we prove that these inequalities together with equations (8.3) yield a complete description of $P^{switch}_{T(L,l)}$.

Theorem 8.3.1 *Equations* (8.3) *and inequalities* (8.10), (8.11), (8.12) *and* (8.13) *give a complete linear description of* $P^{switch}_{T(L,l)}$.

The crucial point of the proof is that the system of linear inequalities (8.3), (8.10), (8.11), (8.12) and (8.13) is totally dual integral (TDI). Interesting to note is that this linear inequality description of the switching polytope is already TDI without adding further constraints.

Proof. We restrict ourselves to the case $T - L + 1 \geq 2$ and $T - l + 1 \geq 2$. The other cases can be handled analogously.
Let $\tilde{P}^{switch}_{T(L,l)}$ be the polytope defined by (8.3), (8.10), (8.11), (8.12) and (8.13). We show $P^{switch}_{T(L,l)} = \tilde{P}^{switch}_{T(L,l)}$ to prove the statement. Since the above inequalities are valid for $P^{switch}_{T(L,l)}$, we have $P^{switch}_{T(L,l)} \subseteq \tilde{P}^{switch}_{T(L,l)}$.
Let $LP^{switch}_{T(L,l)}$ denote the set of solution of the linear programming relaxation of $P^{switch}_{T(L,l)}$.
First, we verify the inclusion $\tilde{P}^{switch}_{T(L,l)} \subseteq LP^{switch}_{T(L,l)}$ by showing that a point that does not lie in $LP^{switch}_{T(L,l)}$ cannot lie in $\tilde{P}^{switch}_{T(L,l)}$. Thereafter we just have to show that the polytope $\tilde{P}^{switch}_{T(L,l)}$ is integral, which implies the missing direction.
As mentioned before one can show that the above system of inequalities, which describes the polytope $\tilde{P}^{switch}_{T(L,l)}$, is TDI, hence the integrality of $\tilde{P}^{switch}_{T(L,l)}$ follows. In the following we sketch the ideas of this proof, details can be found in [Mar05].
Let

$$c := ((c_1, \ldots, c_T), (c^{up}_2, \ldots, c^{up}_T), (c^{up}_2, \ldots, c^{up}_T))^\top \in \mathbb{Z}^{3T-2}$$

be an integral vector. We consider the dual of the inequality system (combined with objective function $c^\top(x, x_{up}, x_{down})^\top$) given by

$$
\begin{aligned}
d_{LP} = \min \sum_{i=2}^{T-l+1} y_i & & & \\
-u_i + \sum_{k=2}^{\min\{i,T-L+1\}} w_k - \sum_{k=2}^{i-l} y_k - z_i &= c_i^{up} & \text{for } & i = 2, \dots, T & (8.14)\\
-v_i - \sum_{k=2}^{i-L} w_k + \sum_{k=2}^{\min\{i,T-l+1\}} y_k + z_i &= c_i^{down} & \text{for } & i = 2, \dots, T & (8.15)\\
-z_2 &= c_1 & & & (8.16)\\
z_i - z_{i+1} &= c_i & \text{for } & i = 2, \dots, T-1 & (8.17)\\
-\sum_{k=2}^{T-L+1} w_k + \sum_{k=2}^{T-l+1} y_k + z_T &= c_T & & & (8.18)\\
u_i, v_i &\geq 0 & \text{for } & i = 2, \dots, T &\\
w_i &\geq 0 & \text{for } & i = 2, \dots, T-L+1 &\\
y_i &\geq 0 & \text{for } & i = 2, \dots, T-l+1 &\\
z_i &\in \mathbb{R} & \text{for } & i = 2, \dots, T. &
\end{aligned}
$$

For proving the TDI property we must find an integral optimal solution if d_{LP} is finite. It is possible to construct such an integral optimal solution for an arbitrary integral c.
By means of constraints (8.16) and (8.17) the values of the z-variables can recursively be calculated. Since c is integral, they are also integral.
Now we come to the more complicated determination of the y- and w-variables. Note that we initially neglect the remaining variables u and v. Let us first sketch the idea. As the y-variables constitute the objective function, we have to minimize their values. In the following we determine positive settings of the y-variables that are forced by (8.15). Thereafter, we iteratively calculate eventually positive values of the w-variables enforced by (8.14). Note that if in this iterative process we receive a positive w-variable, we have to increase the value of exactly one y-variable. Moreover, we see a dependence of the w-variables on the y-variables and vice versa. Hence the increment of this one y-variable might influence other w-variables. But as we will see, this increment only concerns w-variables that are considered later in the iterative determination process. After the setting of all y- and w-variables the value for the u- and v-variables can be calculated.
Let us begin. At first we assign positive values to the y-variables which are forced by (8.15) since

$$
y_i = c_i^{down} - z_i - \sum_{k=2}^{i-1} y_k + v_i + \sum_{k=2}^{i-L} w_k \geq c_i^{down} - z_i - \sum_{k=2}^{i-1} y_k
$$

and calculate

$$y_i = \max\{c_i^{down} - z_i - \sum_{k=2}^{i-1} y_k, 0\}$$

for $i = 2, \dots, T-l$. Variable y_{T-l+1} is considered in a moment, since it may appear in several equations. In the following we must increase a variable y_i if one of the variables w_k for $k = 2, \dots, i-L$ gets positive.

Furthermore we iteratively determine eventually positive values of the w-variables if demanded by equations (8.14). Because of

$$w_i = c_i^{up} + z_i - \sum_{k=2}^{i-1} w_k + \sum_{k=2}^{i-l} y_k + u_i \geq c_i^{up} + z_i - \sum_{k=2}^{i-1} w_k + \sum_{k=2}^{i-l} y_k$$

we have

$$w_i = \max\{c_i^{up} + z_i - \sum_{k=2}^{i-1} w_k + \sum_{k=2}^{i-l} y_k, 0\}$$

for $i = 2, \dots, T-L$. Again the last variable w_{T-L+1} is considered below. Here we recognize that the w-variables depend on the y-variables and vice versa. But this does not put a problem as we will see in the following.

If $w_i > 0$ during the recursive calculation we have to increase variable y_s with $s = \min\{L+i, T-l\}$ if and only if $\min\{L+i, T-l\} = L+i$. Note that because of equations (8.15) it suffices to change exactly one y-variable namely the first one that is affected by w_i. Moreover we have to take into account that an increase of y_s only influences variables w_j with $j \geq s+l = L+i+l$, i.e., variables not yet considered in the iterative process.

We are able to calculate nonnegative integral values of the variables w_i, $i = 2, \dots, T-L$ and y_i, $i = 2, \dots, T-l$ that are required by (8.15) and (8.14).

We set

$$y_{T-l+1} = \max\left\{ \max_{T \geq i \geq T-l+1} \{c_i^{down} - z_i - \sum_{k=2}^{T-l} y_k + \sum_{k=2}^{i-L} w_k\}, 0 \right\}$$

and

$$w_{T-L+1} = \max\left\{ \max_{T \geq i \geq T-L+1} \{c_i^{up} + z_i - \sum_{k=2}^{T-L} w_k + \sum_{k=2}^{i-l} y_k\}, 0 \right\}.$$

To guarantee constraint (8.18) we have to change w_{T-L+1} or y_{T-l+1} depending on the sign of the violation of the equation. If $c_T - z_T > -\sum_{k=2}^{T-L+1} w_k + \sum_{k=2}^{T-l+1} y_k$, we have to increase y_{T-l+1} accordingly, since this has no influence on the values of the w-variables (and hence on other y-variables). Otherwise we increase w_{T-L+1} which has no consequence for values of y-variables.

Finally we determine the u- and v-variables such that equations (8.15) and (8.14) are fulfilled. By construction of the other variables this results in nonnegative variables. For example a variable v_i, for $i = 2, \dots, T-l$ may only be positive if y_i equals zero.

Altogether we have determined integral values of the variables and all constraints are fulfilled. By construction this solution is even optimal, since the decrease of any y-variable would result in a

violation of condition (8.14), (8.15) or (8.18).
□

Therewith the studies of the switching polytope are complete. In the next subsection we have a look on separation in this context.

8.3.2 Separation

For typical practical applications, i.e., $T - L + 1 \geq 2$ and $T - l + 1 \geq 2$ holds, the switching polytope can be described by $T - 1$ equations, $2T - 2$ nonnegativity constraints and $2T - L - l$ additional inequalities with coefficients ± 1 and right-hand side 0 or 1. Hence separation of these conditions is an easy task.
In [Mar05] a corresponding separation algorithm with running time $O(T)$ can be found, which is tested considering three gas networks (see networks in Section 10.1) over different time horizons. The results of the separation algorithm are compared with the model where conditions (8.1) to (8.4) (the formulation found in the literature) are explicitly integrated.

At first switching costs of the compressors are set to zero (hence the coefficients $C^t_{c,up}$ and $C^t_{c,down}$ equals zero in the objective function, see Section 5.5). For the two smaller gas networks the separation algorithm could improve the lower bound by factor 2 and 3. Moreover the number of nodes in the branch-and-bound tree is reduced by factor around 10 for some planning horizons in case of the middle test instance. For the biggest network less improvement of the lower bound could be observed, only a factor around 1.1 was noticeable.
In case of no switching costs, [Mar05] also tested the model where switching processes and minimum runtime and downtime for compressors are completely omitted. It turns out that the neglection of these conditions yields the worst computational results. For the middle test instance the results are comparable with the explicit integration of (8.1) to (8.4), but for the smallest gas network the number of nodes in the tree increases dramatically by factor 2 up to 10, and for the biggest one the lower bound is worse. Thus, the additional conditions for the states of the compressors help to improve the solution algorithm.
In a second step, the two solution methods (separation algorithm and formulation (8.1) to (8.4)) are compared considering several positive values of switching costs and fixations of compressors, i.e., the state variable of a machine is forced to be zero or one in a certain time step. In practice fixations of machines are interesting for example because of maintenance work. In that case the improvement of the lower bound by the separation algorithm is less (factor 1.1 to 1.6) but the number of nodes in the branch-and-bound tree could be reduced significantly by a factor 3 up to 11.
Altogether, the comparisons show the benefit of the separation algorithm, since the lower bound and the size of the tree could be improved. So we integrate it in our solution process for the TTO model.

Chapter 9

Linearization of Functions: SOS Polytopes

In this chapter we concentrate on theoretical studies in connection with the linearization of functions. We focus on an approach using additional binary variables which comes from our first gas network optimization model. In practice we follow the SOS approach without binary variables, see Chapter 6. Nevertheless, we received some interesting polyhedral results that we want to present in this chapter. We used the software package `PORTA` [CL00] to calculate the linear description for small polytopes concerning approximation models.

At first, we introduce a well-known linearization model including binary variables for one-dimensional functions (see also Chapter 3). After this, we concentrate on the paper [LW01] of Lee and Wilson as our polyhedral studies are related to their work. Finally, we completely characterize the SOS 2 and the SOS 3 polytope arising from the linearization of one- and two-dimensional functions.

9.1 Binary Linearization Model

Let $f\colon [a,b] \to \mathbb{R}$ be a nonlinear function with $a, b \in \mathbb{R}$. By subdividing the interval $[a,b]$ into $n-1$ parts we receive $n-1$ segments and n grid points $a = x_1 < \cdots < x_n = b$, see Figure 9.1. Let Λ define the set of indices of these grid points and Y the set of indices of the corresponding segments.

We present the binary model based on which we made our polyhedral studies. This method is known as *convex combination* or *lambda method* and can be found for example in [NW88] (see also Chapter 3). We have the modeling variables λ^i for $i \in \Lambda$ and y^j for $j \in Y$. Further on, we define for each binary variable y^j, $j \in Y$, as neighborhood $N(y^j) = \{j, j+1\}$ the indices of the adjacent λ-variables, and for each λ^i, $i \in \Lambda$, the neighborhood $N(\lambda^i) = \{i-1, i\}$ with the indices of the adjoining y-variables. Now we can formulate an MIP which yields for each $x \in [a,b]$ an

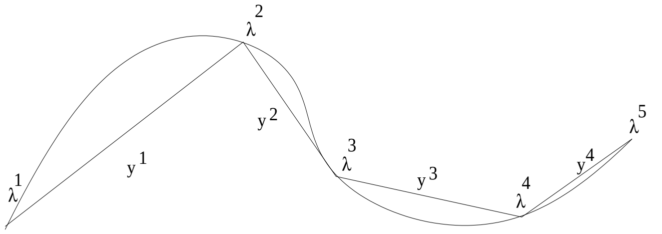

Figure 9.1: Approximation of a function

approximated function value.

$$\sum_{j \in Y} y^j = 1 \tag{9.1}$$

$$\sum_{i \in \Lambda} \lambda^i = 1 \tag{9.2}$$

$$\lambda^i \leq \sum_{k \in N(\lambda^i)} y^k \qquad \text{for all} \quad i \in \Lambda \tag{9.3}$$

$$y^j \in \{0, 1\} \qquad \text{for all} \quad j \in Y \tag{9.4}$$

$$\lambda^i \geq 0 \qquad \text{for all} \quad i \in \Lambda \tag{9.5}$$

$$x = \sum_{i \in \Lambda} x_i \, \lambda^i \tag{9.6}$$

$$f = \sum_{i \in \Lambda} f(x_i) \, \lambda^i \tag{9.7}$$

Because of the first constraint exactly one segment of the decomposition is selected. The second constraint shows the convex combination of the λ-variables. Constraint (9.3) ensures that only those λ-variables can be positive which are adjacent to the chosen segment, that is f cannot be approximated by function values belonging to grid points of different segments. The approximated function value $f(x)$ results from the last equality.
Instead of inequalities (9.3) we could also integrate

$$y^j \leq \sum_{k \in N(y^j)} \lambda^k \quad \text{for all} \quad j \in Y.$$

These formulations are equivalent due to equations (9.1) and (9.2).
Remark that at most two of the λ^i's are positive and if λ^k and λ^l are positive, then $l = k - 1$ or $k + 1$. This kind of sets like Λ are called Special Ordered Sets of Type 2, briefly *SOS Type 2*, see also Section 3.1. They were introduced by [BT70].

In the following we concentrate on the paper [LW01] of Lee and Wilson. The authors determine a related extension of the SOS definition developed in [MMM06] (see also Section 3.2) to approximate functions of more than one variable. There the domain of the function is triangulated by a finite set of simplices, maybe having different dimensions, and then the function is linearly interpolated within each simplex. Therefore, the lambda method is extended to the higher dimensional case. The term ‘adjacency condition’ is introduced corresponding to definition of set condition (see Definition 3.2.1). The authors study the resulting polytope, give facet-defining conditions, and provide a complete linear description.

Remember that in this chapter we consider the SOS 2 and the SOS 3 polytope which arise from the linearization of one- and two-dimensional functions, respectively, over uniform grids. Hence, we just consider two special cases which are also tackled by Lee and Wilson. The studies presented in this chapter were obtained independently. Obviously, we obtain the same results concerning dimension and vertices of the polytopes as well as the characterization of facet-defining trivial inequalities and of valid inequalities. However, in case of the facet-defining conditions for nontrivial inequalities and the complete linear description of the polytope the results differ. We found a mistake in the facet proof of [LW01]. There, only one condition is needed to guarantee the facet property. We analyzed that this condition suffices in case of a one-dimensional triangulation, i.e., the SOS 2 polytope. But already for a two-dimensional triangulation, i.e., the SOS 3 polytope, further conditions are necessary. In Theorem 9.3.1 we show that in this case three conditions are needed. Furthermore, we suppose that in higher dimensions at least as much conditions must be fulfilled as in the two-dimensional case.

As a consequence of this mistake, the proof for the complete linear description of the polytope in [LW01] is not correct. The authors show that every nontrivial facet of the polytope is described by an inequality fulfilling their facet condition. But since the facet proof is incorrect some of these inequalities are redundant. Thus, we can say that to the best of our knowledge we are the first who found a complete irredundant linear description of the SOS 3 polytope.

In the following, we present a counter-example for the facet criterion of [LW01]. At first we give an introduction of the notation in [LW01]. A function $f : D \to \mathbb{R}$ is to be approximated, where the domain $D \subseteq \mathbb{R}^d$ is the union of a finite number of polytopes and no component of D is a single point. D is triangulated with a finite set of simplices $\mathscr{T}$ having the vertex set $\mathscr{V}(\mathscr{T})$, where the intersection of any two simplices is a proper face of each. For a simplex $T \in \mathscr{T}$ let $V(T)$ be the set of its vertices. For each vertex $v \in \mathscr{V}(\mathscr{T})$, a variable λ_v and, for each simplex $T \in \mathscr{T}$, a variable y_T is introduced. Then the function $f(x)$ is approximated by $\sum_{v\in\mathscr{V}(\mathscr{T})} \lambda_v f(v)$ via the following constraints (note that this formulation corresponds to (9.1) till (9.5))

$$\lambda_v \geq 0 \qquad \text{for all} \quad v \in \mathscr{V}(\mathscr{T}) \tag{9.8}$$

$$\sum_{v\in\mathscr{V}(\mathscr{T})} \lambda_v = 1 \tag{9.9}$$

$$\sum_{T\in\mathscr{T}} y_T = 1 \tag{9.10}$$

$$\lambda_v - \sum_{T:v\in V(T)} y_T \leq 0 \qquad \text{for all} \quad v \in \mathscr{V}(\mathscr{T}) \tag{9.11}$$

$$y_T \in \{0,1\} \quad \text{for all} \quad T \in \mathscr{T}. \tag{9.12}$$

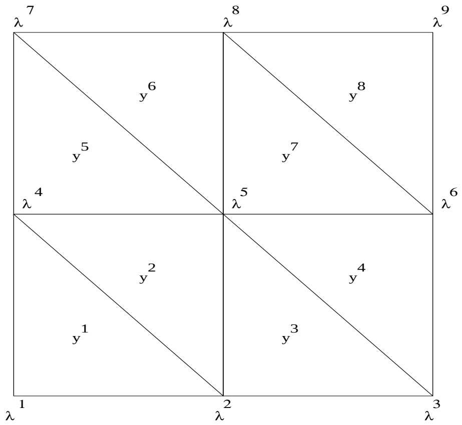

Figure 9.2: Triangulation for counter-example

Let $P(\mathscr{T})$ be the polytope defined by these constraints, i.e., the convex hull of the solutions of (9.8) to (9.12).
For $\mathscr{B} \subset \mathscr{T}$, Lee and Wilson consider the inequality

$$\sum_{T \in \mathscr{B}} y_T - \sum_{v \in \mathscr{V}(\mathscr{B})} \lambda_v \leq 0 \tag{9.13}$$

which says that if a simplex in $\mathscr{B}$ is selected, then the vertices used to interpolate the function must all belong to simplices from $\mathscr{B}$. Note that this type of inequality corresponds to valid inequalities given in Lemma 9.2.3 and 9.3.3 for the SOS 2 and SOS 3 polytope. Furthermore, let

$$D_{\mathscr{B}} := \{x \in D \mid x \in T, \text{ for some } T \in \mathscr{B}\}$$

for $\mathscr{B} \subset \mathscr{T}$. They define that the set $\mathscr{B} \subset \mathscr{T}$ is a *k-breaking set* of simplices if

$$\kappa(D_{\mathscr{B}}) + \kappa(D \setminus D_{\mathscr{B}}) = \kappa(D) + k,$$

where $\kappa(S)$ is the number of components of some set $S \subseteq \mathbb{R}^d$. Now we can specify the facet-defining condition of Lee and Wilson.

Proposition 9.1.1 *(See Proposition 9 in [LW01])*
An inequality of the form (9.13) *describes a facet of $P(\mathscr{T})$ if and only if $\mathscr{B} \subset \mathscr{T}$ is a 1-breaking set.*

One direction of this proposition is correct: If an inequality of the form (9.13) describes a facet, then $\mathscr{B}$ is a 1-breaking set. For the other direction '$\Leftarrow$' we found a counter-example. In the following we use the canonical form of inequalities given in [LW01].

For a counter-example regard the triangulation $\mathscr{T}_E$ shown in Figure 9.2. It consists of eight triangles and nine vertices. We choose the set $\mathscr{B} = \{1, \ldots, 6\}$ which is a 1-breaking set with the corresponding inequality

$$y^1 + \ldots + y^6 - \lambda^1 - \ldots - \lambda^8 \leq 0. \tag{9.14}$$

But this inequality does not yield a facet of $P(\mathscr{T}_E)$. To see this, take the sets $\mathscr{B}_1 = \{1, \ldots, 7\}$ and $\mathscr{B}_2 = \{1, \ldots, 6, 8\}$. Note that these sets are also 1-breaking sets with inequalities

$$\begin{gathered} y^1 + \ldots + y^7 - \lambda^1 - \ldots - \lambda^8 \leq 0, \\ y^1 + \ldots + y^6 + y^8 - \lambda^1 - \ldots - \lambda^9 \leq 0. \end{gathered}$$

Adding these inequalities gives

$$y^1 + \ldots + y^8 + y^1 + \ldots + y^6 - \lambda^1 - \ldots \lambda^9 - \lambda^1 - \ldots - \lambda^8 \leq 0.$$

Taking equations (9.9) and (9.10) into account this results in (9.14), hence, it cannot be a facet of $P(\mathscr{T}_E)$.

9.2 The SOS 2 Polytope

In this section we analyze the polytope arising from the linearization of a one-dimensional function and the SOS Type 2 condition.

We use the notation of the introduction, i.e., let $\Lambda = \{1, ..., n\}$ be the SOS Type 2 of λ-variables and let $Y = \{1, ..., n-1\}$ be the set of indices of binary variables. Further let $N(\lambda^i) = \{i-1, i\}$ and $N(y^j) = \{j, j+1\}$ be the neighborhoods of a λ-variable and a y-variable indicating the indices of the adjacent binary and λ-variables, respectively. Remark that all neighborhoods have cardinality two except that of the first and last λ-variable which have cardinality one.

As already shown, the Special Ordered Set of Type 2 problem, briefly SOS 2 problem, can be formulated as a mixed integer program as follows

$$\sum_{j \in Y} y^j = 1 \tag{9.15}$$

$$\sum_{i \in \Lambda} \lambda^i = 1 \tag{9.16}$$

$$\lambda^i \leq \sum_{k \in N(\lambda^i)} y^k \quad \text{for all} \quad i \in \Lambda \tag{9.17}$$

$$y^j \in \{0, 1\} \quad \text{for all} \quad j \in Y \tag{9.18}$$

$$\lambda^i \geq 0 \quad \text{for all} \quad i \in \Lambda. \tag{9.19}$$

As abbreviation, we define

$$\lambda := (\lambda^1, ..., \lambda^n)^\top \quad \text{and} \quad y := (y^1, ..., y^{n-1})^\top.$$

We are interested in the SOS 2 polytope $P^n_{SOS\ 2}$ which is defined as the convex hull of all feasible solutions of the SOS 2 problem, that is

$$P^n_{SOS\ 2} := \mathrm{conv}\{(\lambda, y) \in \mathbb{R}^{2n-1} \mid (\lambda, y) \text{ satisfies (9.15) - (9.19)}\}.$$

At first, we have a look at the dimension of $P^n_{SOS\ 2}$.
Before, let us denote by

$$e^k_l \in \mathbb{R}^{2n-1} \quad \text{for } k \in \Lambda \text{ and } l \in Y$$

the point with all components equal to zero except the two components belonging to λ^k and y^l which have value one (these are the kth and the $(n + l)$th component of the vector). As we will see later, the vertices of the SOS 2 polytope are of that form.

Lemma 9.2.1 *dim* $(P^n_{SOS\ 2}) = 2n - 3$.

Proof. Because of the equations (9.15) and (9.16) it is obvious that dim $(P^n_{SOS\ 2}) \leq 2n - 3$.
Now we choose the following $2n - 2$ points

$$e^1_1, e^n_{n-1} \quad \text{and} \quad e^k_{k-1}, e^k_k \quad \text{for } k \in \Lambda \setminus \{1, n\}$$

which lie in $P^n_{SOS\ 2}$. Showing that they are linearly independent implies the statement.
Hence consider

$$\mu^1_1 e^1_1 + \mu^n_{n-1} e^n_{n-1} + \sum_{k\in\Lambda\setminus\{1,n\}} \sum_{l\in N(\lambda^k)} \mu^k_l e^k_l = 0$$

and show that all μ-variables are equal to zero.
Positive entries of the components belonging to λ^1 and λ^n can just be found in the first two terms, so their coefficients μ^1_1 and μ^n_{n-1} are equal to zero.
Further one can conclude inductively that for each neighborhood $N(\lambda^k)$, $k = 2, ..., n - 1$, the μ-variables vanish. $\mu^k_{k-1} = 0$ holds since y^{k-1} is the y-variable with the lowest index (still in the equation) and it appears just at this place. From this it follows that $\mu^k_k = 0$ because here is the last positive λ^k-component. □

Having determined the dimension of the polytope we consider its vertices. We already used them in the preceding proof.

Lemma 9.2.2 *The vertices of* $P^n_{SOS\ 2}$ *are given by the points* e^k_j *with* $j \in Y$ *and* $k \in N(y^j)$. *Altogether there exist* $2n-2$ *vertices.*

Proof. The feasible solutions of the problem have the following form. Exactly one of the y-variables is equal to one the remaining are zero. If $y^j = 1$ only those λ-variables adjacent to the jth segment can be positive, moreover they are convex combined. This proves the claim. □

In the next step we introduce a new class of valid inequalities. They generalize the constraint (9.17) of the SOS 2 formulation.

Lemma 9.2.3 *Let $\emptyset \neq J \subseteq Y$. The inequality*

$$\sum_{k\in\Lambda:\ N(\lambda^k)\subseteq J} \lambda^k \leq \sum_{l\in J} y^l \tag{9.20}$$

is valid for $P^n_{SOS\ 2}$.

Proof. Let $\emptyset \neq J \subseteq Y$ and show that the inequality is valid for all vertices of $P^n_{SOS\ 2}$. Let e^k_l be a vertex with $l \in Y$ and $k \in N(y^l)$.
If $l \in J$, the point e^k_l fulfills the inequality because the right-hand side is equal to one and the sum of the left-hand side is at most one.
If $l \notin J$, it fulfills the inequality because in that case both sides are equal to zero since $l \in N(\lambda^k) \not\subseteq J$. □

Later we will give conditions for this kind of inequality to be facet-defining, but we previously examine which of the nonnegativity constraints represent facets.

Lemma 9.2.4

- (i) *The nonnegativity constraints $\lambda^1 \geq 0$ and $\lambda^n \geq 0$ define facets of $P^n_{SOS\ 2}$.*
- (ii) *The nonnegativity constraints $\lambda^k \geq 0$ for $k \in \Lambda \setminus \{1, n\}$ and $y^l \geq 0$ for $l \in Y$ are redundant in the description of $P^n_{SOS\ 2}$.*

Proof.

- (i) The $2n - 3$ points e^k_{k-1}, e^k_k for $k \in \Lambda \setminus \{1, n\}$ and e^n_{n-1} (e^1_1) satisfy $\lambda^1 \geq 0$ ($\lambda^n \geq 0$) at equality and are linearly independent (see proof of Lemma 9.2.1).
- (ii) Let $k \in \Lambda \setminus \{1, n\}$. We choose the sets $J_1 = \{1, ..., k-1\}$ and $J_2 = \{k, ..., n-1\}$ and take the corresponding inequalities given by Lemma 9.2.3. Adding these two valid inequalities yields $\lambda^k \geq 0$.
  Let $l \in Y$. We choose the sets $J_1 = \{1, ..., l\}$ and $J_2 = \{l, ..., n-1\}$ and take the corresponding inequalities given by Lemma 9.2.3. Adding these two valid inequalities yields the redundance of $y^l \geq 0$.

□

After describing the trivial facets we return to the more interesting class of inequalities of Lemma 9.2.3 and give facet-defining conditions.

Lemma 9.2.5 *Let* $\emptyset \neq J \subseteq Y$, $J \neq Y$. *The inequality* (9.20) *defines a facet of* $P^n_{SOS\ 2}$ *if and only if* $J = \{1, ..., j\}$ *or* $J = \{j, ..., n-1\}$ *for a* $j \in Y$.

It is easy to illustrate the condition for the inequality to be irredundant. The indices in J as well as $Y \setminus J$ must be consecutive.
Before we prove this lemma, we consider as example the case $n = 5$ as depicted in Figure 9.1. In this case we have nine variables $\lambda_1, ..., \lambda_5$ and $y_1, ..., y_4$ and the corresponding formulation (9.15) to (9.19). The point $\left(\frac{1}{2}, \frac{1}{2}, 0, 0, 0, \frac{1}{2}, 0, \frac{1}{2}, 0\right)$ is a vertex of the polytope $LP^n_{SOS\ 2}$ defined by the linear programming relaxation, where the integrality condition of the y-variables is neglected. This point is cut off by the inequality

$$\lambda^1 + \lambda^2 \leq y^1 + y^2$$

which defines a facet of $P^5_{SOS\ 2}$ according to Lemma 9.2.5.

Proof. We first show that the inequality is facet-defining if the set J has one of the above mentioned form. W.l.o.g. let $J = \{1, ..., j\}$ with $1 \leq j < n-1$ and get the inequality

$$\sum_{k=1}^{j} \lambda^k \leq \sum_{l=1}^{j} y^l.$$

The $2n-3$ points e^1_1, e^n_{n-1}, e^{j+1}_{j+1} and e^k_{k-1}, e^k_k for $k \in \{2, ..., j\} \cup \{j+2, ..., n-1\}$ are linearly independent (see proof of Lemma 9.2.1) and are tight for it.
It remains to prove the converse direction. Here we make a case differentiation.
At first we assume that $J = \{m_b, ..., m_e\}$ with $1 < m_b \leq m_e < n-1$ that is only the indices of the set $Y \setminus J$ are not consecutive. For J we get the inequality

$$\sum_{k=m_b+1}^{m_e} \lambda^k \leq \sum_{l=m_b}^{m_e} y^l.$$

We choose the sets $J_1 = \{1, ..., m_e\}$ and $J_2 = \{m_b, ..., n-1\}$ and take the corresponding inequalities. Adding them yields

$$\sum_{k=1}^{n} \lambda^k + \sum_{k=m_b+1}^{m_e} \lambda^k \leq \sum_{l=1}^{n-1} y^l + \sum_{l=m_b}^{m_e} y^l$$

which is equivalent to the above inequality.
After that we consider the case that either J is not consecutive or both sets J and $Y \setminus J$ are not consecutive. Therefore let J be of the following form

$$J = \bigcup_{p=1}^{q} J^p$$

whereby $q \geq 2$ and $\emptyset \neq J^p = \{m_b^p, ..., m_e^p\} \subseteq Y$ for $p = 1, ..., q$ with $m_e^p < m_b^{p+1}$ for $p = 1, ..., q-1$. So J is the disjoint union of q sets consisting of consecutive elements of Y. Notice that this representation is unique since it is required that the sets J^p have maximal possible cardinality (see condition $m_e^p < m_b^{p+1}$).
It is easy to see that we receive the inequality belonging to J by adding the q inequalities defined by the sets J^p because of the maximal and disjunctive property. This proves the statement. □

Now we show that we have already found all inequalities describing $P^n_{SOS\ 2}$.

Theorem 9.2.1 *The facet-defining inequalities (i) of Lemma 9.2.4 and the ones of Lemma 9.2.5 together with the equalities* (9.15) *and* (9.16) *provide a complete linear description of* $P^n_{SOS\ 2}$.

Remark 9.2.2 *The SOS 2 polytope* $P^n_{SOS\ 2}$ *is described by two equality constraints and altogether* $2n - 2$ *inequalities, i.e., two nonnegativity constraints and* $2n - 4$ *constraints given by Lemma 9.2.5.*

Proof. Let $\tilde{P}^n_{SOS\ 2}$ be the polytope defined by (9.15), (9.16), $\lambda^1 \geq 0$, $\lambda^n \geq 0$ and the inequalities

$$\sum_{k=1}^{j} \lambda^k \leq \sum_{l=1}^{j} y^l \text{ for } j = 1, ..., n-2 \quad \text{and} \quad \sum_{k=j+1}^{n} \lambda^k \leq \sum_{l=j}^{n-1} y^l \text{ for } j = 2, ..., n-1 \qquad (9.21)$$

from Lemma 9.2.5.
To prove the statement we will show that $\tilde{P}^n_{SOS\ 2} = P^n_{SOS\ 2}$. First note that $P^n_{SOS\ 2} \subseteq \tilde{P}^n_{SOS\ 2}$, since the above inequalities are valid for $P^n_{SOS\ 2}$. Furthermore, it is clear that $\tilde{P}^n_{SOS\ 2} \subseteq LP^n_{SOS\ 2}$ holds, where $LP^n_{SOS\ 2}$ is the solution set of the linear programming relaxation of $P^n_{SOS\ 2}$. Therefore, whenever $(\lambda, y) \in \tilde{P}^n_{SOS\ 2}$ and y is integer, then $(\lambda, y) \in P^n_{SOS\ 2}$. We will conclude $\tilde{P}^n_{SOS\ 2} \subseteq P^n_{SOS\ 2}$ by proving that a vertex of $\tilde{P}^n_{SOS\ 2}$ cannot have fractional y-components.

Let $(\bar{\lambda}, \bar{y})$ be a vertex of $\tilde{P}^n_{SOS\ 2}$ and suppose that the point $(\bar{\lambda}, \bar{y})$ has more than one positive y-component. We will show that $(\bar{\lambda}, \bar{y})$ is a nontrivial convex combination of other points in $\tilde{P}^n_{SOS\ 2}$, which is a contradiction, since in that case $(\bar{\lambda}, \bar{y})$ cannot be a vertex of $\tilde{P}^n_{SOS\ 2}$. Thus, for every vertex of $\tilde{P}^n_{SOS\ 2}$ the y-components must be integer.

Notice that $n \geq 3$ holds, since at least two y-variables are positive, and thus $n-1 \geq 2$ holds. Since $(\bar{\lambda}, \bar{y})$ is a vertex of $\tilde{P}^n_{SOS\ 2}$, it has to satisfy at least $2n-1$ constraints at equality. As mentioned above, there are $2n-2$ inequalities and two equalities defining the polytope $\tilde{P}^n_{SOS\ 2}$, thus, at most one inequality cannot be satisfied at equality by $(\bar{\lambda}, \bar{y})$. Here we consider two cases. Either $(\bar{\lambda}, \bar{y})$ is not tight for one of the nonnegativity constraints or it is not tight for one of the inequalities (9.21).

First, we assume $\bar{\lambda}^1 > 0$. The proof of the case $\bar{\lambda}^n > 0$ works analogously. In the former case all inequalities (9.21) must be fulfilled at equality by $(\bar{\lambda}, \bar{y})$. Inequality $\lambda^1 \leq y^1$ yields $\bar{\lambda}^1 = \bar{y}^1$. For $j = 2, ..., n-2$, $(\bar{\lambda}, \bar{y})$ is tight for

$$\sum_{k=1}^{j-1} \lambda^k \leq \sum_{l=1}^{j-1} y^l \quad \text{and} \quad \sum_{k=1}^{j} \lambda^k \leq \sum_{l=1}^{j} y^l$$

yielding $\bar{\lambda}^j = \bar{y}^j$. Furthermore, $\bar{\lambda}^n = 0$ holds due to the second nonnegativity constraint. And finally, we have

$$\bar{\lambda}^{n-1} \overset{(9.16)}{=} 1 - \sum_{k=1}^{n-2} \bar{\lambda}^k - \bar{\lambda}^n = 1 - \sum_{k=1}^{n-2} \bar{y}^k \overset{(9.15)}{=} \bar{y}^{n-1}.$$

Thus $(\bar{\lambda}, \bar{y})$ can be written as $(\bar{\lambda}, \bar{y}) = \sum_{j=1}^{n-1} \bar{y}^j \ e^j_j$. This is a convex combination of at least two points in $\tilde{P}^n_{SOS\ 2}$, as $e^j_j \in P^n_{SOS\ 2} \subseteq \tilde{P}^n_{SOS\ 2}$ holds, and we supposed more than one y-component to be positive. Hence, we obtain a contradiction to $(\bar{\lambda}, \bar{y})$ being a vertex of $\tilde{P}^n_{SOS\ 2}$.

Second, we consider w.l.o.g. $\sum_{k=1}^{j^*} \bar{\lambda}^k < \sum_{l=1}^{j^*} \bar{y}^l$ for a $j^* \in \{1, ..., n-2\}$. The case $\sum_{k=j^*+1}^{n} \bar{\lambda}^k < \sum_{l=j^*}^{n-1} \bar{y}^l$ for some $j^* \in \{2, ..., n-1\}$ can be treated analogously. In the former case we obtain $\bar{\lambda}^n = \bar{y}^{n-1}$ from the inequality $\lambda^n \leq y^{n-1}$. For $j = 3, ..., n-1$, $(\bar{\lambda}, \bar{y})$ is tight for

$$\sum_{k=j}^{n} \lambda^k \leq \sum_{l=j-1}^{n-1} y^l \quad \text{and} \quad \sum_{k=j+1}^{n} \lambda^k \leq \sum_{l=j}^{n-1} y^l$$

yielding $\bar{\lambda}^j = \bar{y}^{j-1}$ for $j = 3, ..., n$. Now we have to consider the cases $j^* \neq 1$ and $j^* = 1$.

If $j^* \neq 1$ holds, the inequality $\lambda^1 \leq y^1$ gives $\bar{\lambda}^1 = \bar{y}^1$. Finally, we can calculate

$$\bar{\lambda}^2 \overset{(9.16)}{=} 1 - \sum_{k=3}^{n} \bar{\lambda}^k - \bar{\lambda}^1 = 1 - \sum_{k=3}^{n} \bar{y}^{k-1} - y^1 \overset{(9.15)}{=} 0.$$

Thus $(\bar{\lambda}, \bar{y})$ can be written as $(\bar{\lambda}, \bar{y}) = \sum_{j=3}^{n} \bar{y}^{j-1} \ e^j_{j-1} + \bar{y}^1 \ e^1_1$ which is a convex combination of at least two points in $\tilde{P}^n_{SOS\ 2}$, a contradiction to $(\bar{\lambda}, \bar{y})$ being a vertex.

For $j^* = 1$, we have $\bar{\lambda}^1 < \bar{y}^1$. Beyond, $\bar{\lambda}^1 = 0$ holds because of the nonnegativity constraint. Therefore, inequality $\lambda^1 + \lambda^2 \leq y^1 + y^2$ gives $\bar{\lambda}^2 = \bar{y}^1 + \bar{y}^2$. Further on, we can calculate

$$1 \overset{(9.16)}{=} \sum_{k=1}^{n} \bar{\lambda}^k \overset{\bar{\lambda}^1=0}{=} \bar{y}^1 + \bar{y}^2 + \sum_{k=3}^{n} \bar{y}^{k-1} \overset{(9.15)}{=} \bar{y}^2 + 1.$$

Hence, $\bar{y}^2 = 0$ holds yielding $\bar{\lambda}^2 = \bar{y}^1$. Note that in the case $n = 3$ we have already finished the proof, since we get a contradiction. There are only two y-variables and we assumed that at least two are positive. Thus, $(\bar{\lambda}, \bar{y})$ can be written as $(\bar{\lambda}, \bar{y}) = \sum_{j=2}^{n} \bar{y}^{j-1}\, e^{j}_{j-1}$ which is a convex combination of at least two points in $\tilde{P}^n_{SOS\ 2}$, again a contradiction to $(\bar{\lambda}, \bar{y})$ being a vertex.
This completes the proof. □

To complete the studies of the polytope $P^n_{SOS\ 2}$, we show how to separate the inequalities (9.21). For this purpose we consider the inequalities

$$\sum_{k=1}^{j} \lambda^k \leq \sum_{l=1}^{j} y^l \quad \text{for } j = 1, ..., n-2 \tag{9.22}$$

and indicate a separation algorithm. This algorithm can be adjusted to the other kind of inequalities (9.21).
Let (λ^*, y^*) be an optimal solution for the linear relaxation. If y^* is not integer, that is, (λ^*, y^*) is not feasible for the SOS 2 problem, we want to find an index j^* of the maximally violated inequality. It is easy to see that j^* is given by

$$j^* = \arg \max_{j \in \{1,...,n-2\}} \sum_{k=1}^{j} (\lambda^{*k} - y^{*k}),$$

if the maximum is positive. Hence, we receive the following algorithm.

Algorithm 5 Separation of inequalities (9.22)

```
Define Sum := 0, MaxSum := 0 and $j^*$ := 0.
for $j = 1, ..., n-2$ do
    Let Sum := Sum + $\lambda^{*j}$ - $y^{*j}$.
    if Sum > MaxSum then
        Update MaxSum := Sum.
        Set index $j^*$ := $j$.
    end if
end for
Return index $j^*$.
```

Either the algorithm returns the index j^* of a maximally violated inequality or it returns $j^* = 0$ signifying that there is no violated inequality among the inequalities (9.22).

9.3 The SOS 3 Polytope

As extension of the SOS 2 problem we investigate the polytope defined by the SOS Type 3 condition, see also Section 3.2. This condition can be used to linearize two-dimensional functions.

We start by concretizing the term SOS Type 3, briefly SOS 3. Given a rectangle by the intervals $[a, b] \times [c, d]$ with $a, b, c, d \in \mathbb{R}$, we determine an (equidistant) decomposition. We subdivide the first interval $[a, b]$ into $n - 1$ parts and the second one into $m - 1$ parts with $n, m \geq 2$. So we obtain nm grid points and $(n-1)(m-1)$ small rectangles. These small rectangles are then halved in triangles. We identify a nonnegative λ-variable with each grid point and a y-variable with each triangle and number them serially, see Figure 9.3 for an illustration where $n = 6$ and $m = 4$.

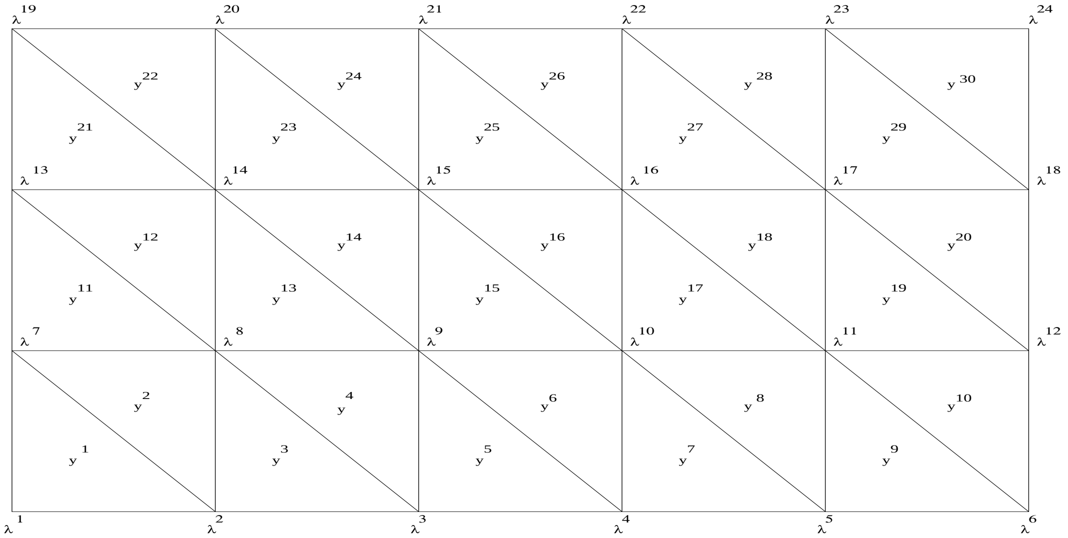

Figure 9.3: Decomposition of a rectangle

For the set of all λ-variables the SOS 3 condition states that the set of λ-variables which are strictly positive must belong to grid points of the same triangle.

Now we formulate a mixed integer program modeling the SOS 3 condition. Hereby, we extend the concept of the SOS 2 problem.

Once again let $\Lambda = \{1, ..., nm\}$ be the set of indices of λ-variables and $Y = \{1, ..., 2(n-1)(m-1)\}$ the set of indices of y-variables. The λ-variables are bounded by zero and one and the y-variables are binary. Further on, for each binary variable y^j, $j \in Y$, we define as neighborhood $N(y^j)$ the indices of the λ-variables belonging to the vertices of the jth triangle, and for each λ^i, $i \in \Lambda$, the neighborhood $N(\lambda^i)$ is given by the set of numbers of the adjacent triangles. Remark that the neighborhoods of the y-variables all have cardinality three whereas the cardinality of the λ-neighborhoods varies depending on the position of its grid point. If λ^i corresponds to a vertex or lies on an edge of the rectangle, then $|N(\lambda^i)| \in \{1, 2\}$ or $|N(\lambda^i)| = 3$, respectively. If it lies inside

the rectangle, $|N(\lambda^i)| = 6$ holds.
With our numbering (see Figure 9.3) we receive

$$\begin{aligned} &|N(\lambda^1)| = |N(\lambda^{nm})| = 1, \quad |N(\lambda^n)| = |N(\lambda^{n(m-1)+1})| = 2, \\ &|N(\lambda^i)| = 3 \quad \text{for } i \in \{2, ..., n-1\} \cup \{n(m-1)+2, ..., nm-1\} \cup \\ &\qquad\qquad \{n+1, 2n, 2n+1, 3n, ..., n(m-2)+1, n(m-1)\}, \end{aligned}$$

and cardinality six for all other λ.
The SOS 3 condition can be formulated as the following mixed integer program

$$\sum_{j \in Y} y^j = 1 \tag{9.23}$$

$$\sum_{i \in \Lambda} \lambda^i = 1 \tag{9.24}$$

$$\lambda^i \leq \sum_{k \in N(\lambda^i)} y^k \quad \text{for all} \quad i \in \Lambda \tag{9.25}$$

$$y^j \in \{0, 1\} \qquad \text{for all} \quad j \in Y \tag{9.26}$$

$$\lambda^i \geq 0 \qquad \text{for all} \quad i \in \Lambda. \tag{9.27}$$

Equation (9.23) ensures that exactly one triangle is chosen and constraint (9.25) guarantees that only λ-variables belonging to the vertices of this triangle can be positive. These constraints extend the SOS 2 condition, where the corresponding constraints (9.15) and (9.17) guarantee that exactly one segment is choosen and the positive λ-variables must be adjacent to this segment.

If a nonlinear function $f : [a, b] \times [c, d] \to \mathbb{R}$ shall be approximated, we have to calculate the exact function value $f(x_i^1, x_i^2)$, $i \in \Lambda$, for each grid point of the decomposition (where $x_i^1 \in [a, b]$ and $x_i^2 \in [c, d]$). Then we obtain additional equalities

$$x^1 = \sum_{i \in \Lambda} x_i^1 \, \lambda^i, \quad x^2 = \sum_{i \in \Lambda} x_i^2 \, \lambda^i, \quad f(x^1, x^2) \approx \sum_{i \in \Lambda} f(x_i^1, x_i^2) \, \lambda^i.$$

We again abbreviate

$$\lambda := (\lambda^1, ..., \lambda^{nm})^\top \quad \text{and} \quad y := (y^1, ..., y^{2(n-1)(m-1)})^\top.$$

In the following we study the SOS 3 polytope $P_{SOS\ 3}^{nm}$ which is defined as the convex hull of the incidence vectors of sets fulfilling the SOS 3 condition, i.e.,

$$P_{SOS\ 3}^{nm} = \text{conv}\{(\lambda, y) \in \mathbb{R}^{nm+2(n-1)(m-1)} \mid (\lambda, y) \text{ satisfies (9.23) - (9.27)}\}.$$

Like in the previous section, we denote by

$$e_l^k \in \mathbb{R}^{nm+2(n-1)(m-1)} \quad \text{for } k \in \Lambda \text{ and } l \in Y$$

the point with all components equal to zero except the two components belonging to λ^k and y^l which have value one (these are the kth and the $(nm + l)$th component of the vector).
We begin with the vertices, since they have the same structure as the ones of the SOS 2 polytope.

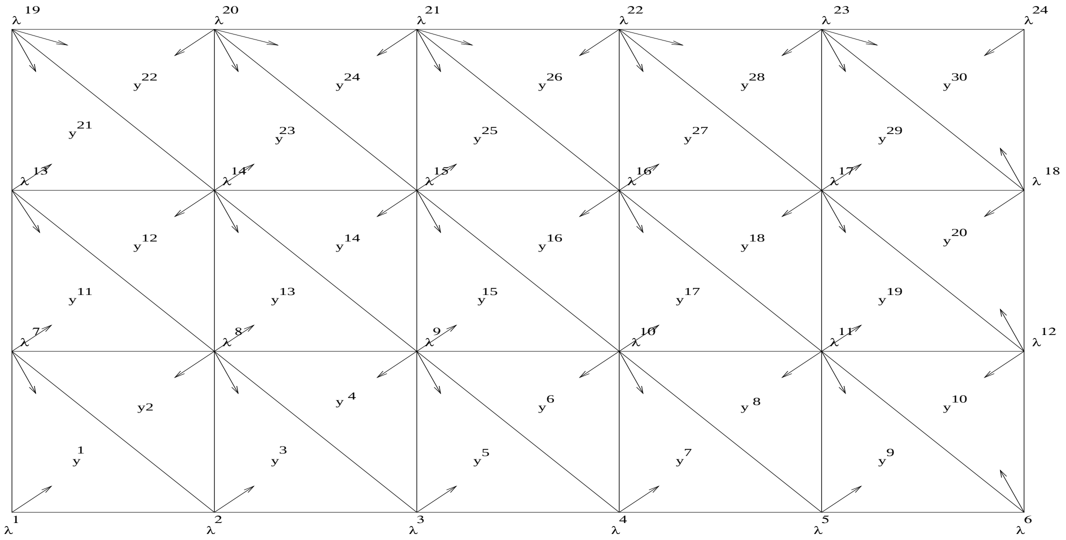

Figure 9.4: Linear independence

Lemma 9.3.1 *The vertices of* $P^{nm}_{SOS\ 3}$ *are given by the points* e^k_j *with* $j \in Y$ *and* $k \in N(y^j)$. *Altogether there exist* $6(n-1)(m-1)$ *vertices.*

The argumentation of Lemma 9.2.2 can be aplied to prove this lemma.
Likewise, we adapt the dimension to the SOS 3 polytope.

Lemma 9.3.2 *dim* $(P^{nm}_{SOS\ 3}) = nm + 2(n-1)(m-1) - 2.$

Proof. Because of equations (9.23) and (9.24) the dimension of $P^{nm}_{SOS\ 3}$ is at most $nm + 2(n-1)(m-1) - 2$.
Now we choose the following $nm + 2(n-1)(m-1) - 1$ vertices of $P^{nm}_{SOS\ 3}$

$$\begin{aligned} \text{for } i \in \Lambda: \qquad & e^i_{j^*} \quad \text{with} \quad j^* = \max_{j \in N(\lambda^i)} j \\ \text{for } j \in Y \setminus \{2(n-1)(m-1)\}: \qquad & e^{i^*}_j \quad \text{with} \quad i^* = \max_{i \in N(y^j)} i \end{aligned}$$

and point out that they are linearly independent which proves the lemma. In Figure 9.4 this selection is illustrated whereby each little arrow stands for a vertex. Head and tail of an arrow specify the positive y- resp. λ-component of the vector. Let us consider

$$\sum_{i \in \Lambda} \mu^i_{j^*} e^i_{j^*} + \sum_{j \in Y \setminus \{2(n-1)(m-1)\}} \mu^{i^*}_j e^{i^*}_j = 0$$

and show that the μ-variables must be zero. Instead of juggling with indices we argue by means of sketch 9.4. We proceed iteratively by the rows of (small) rectangles from the bottom up.

We begin with the lowest row. If we look at the λ-variables at the lower edge of this row we remark that they all have just one arrow thus the corresponding μ-factors ($\mu^i_{j^*}$) must vanish. Deleting these arrows there is exactly one arrow left in each triangle implying that the μ-variables of the associated y-variables must be zero ($\mu^{i^*}_j$). So we can neglect the arrows of the first row.
The remaining rows can be treated in an analogous manner except the last (upper) one. Here the arrows at the lower edge can be deleted in the same way but then some triangles are left including two arrows. However we handle the rectangles of this upper row iteratively from left to right. In each rectangle there are three arrows. The first one (left) can be omitted since it is the only one in the lower triangle. As a consequence the middle arrow that remains for the left upper λ-variable can be neglected. Finally the upper triangle owns just one arrow. $\square$

Furthermore, we obtain the same class of valid inequalities as for the SOS 2 problem extending the neighborhood constraints (9.25).

Lemma 9.3.3 *Let $\emptyset \neq J \subseteq Y$. The inequality*

$$\sum_{k \in \Lambda:\ N(\lambda^k) \subseteq J} \lambda^k \leq \sum_{l \in J} y^l$$

is valid for $P^{nm}_{SOS\ 3}$.

The proof of this lemma is similar to the corresponding proof for the SOS 2 polytope, see Lemma 9.2.3.
Before we give facet-defining conditions for this kind of inequalities in case of the SOS 3 polytope we look at trivial facets resulting from nonnegativity constraints.

Lemma 9.3.4

- *(i) For $i \in \Lambda$ the nonnegativity constraint $\lambda^i \geq 0$ defines a facet of $P^{nm}_{SOS\ 3}$.*
- *(ii) Let $j \in Y$ with $|N(\lambda^k)| > 1$ for all $k \in N(y^j)$. Then the nonnegativity constraint $y^j \geq 0$ defines a facet of $P^{nm}_{SOS\ 3}$.*
- *(iii) Let $j \in Y$ such that there exists a $k \in N(y^j)$ with $|N(\lambda^k)| = 1$. Then the nonnegativity constraint $y^j \geq 0$ is redundant in the description of $P^{nm}_{SOS\ 3}$.*

For example for the SOS 3 problem given by Figure 9.3 all nonnegativity constraints except the two ones for y^1 and y^{30} represent facets of $P^{6,4}_{SOS\ 3}$ (as $|N(\lambda^1)| = |N(\lambda^{24})| = 1$). The redundance is obvious, since the inequalities $\lambda^1 \leq y^1$ and $\lambda^{24} \leq y^{30}$ are valid for $P^{6,4}_{SOS\ 3}$.
In general, for a grid of nm points, just the inequalities $y^1 \geq 0$ and $y^{2(n-1)(m-1)} \geq 0$ do not induce trivial facets, since $y^1 \geq \lambda^1$ and $y^{2(n-1)(m-1)} \geq \lambda^{nm}$ are valid for $P^{nm}_{SOS\ 3}$.

Proof.

(i) Let $\tilde{i} \in \Lambda$. We take (assuming that $\lambda^{\tilde{i}}$ does not lie at the upper edge) the $nm + 2(n-1)(m-1) - 2$ points

$$\begin{array}{rll} \text{for } i \in \Lambda \setminus \{\tilde{i}\}: & e^i_{j^*} \quad \text{with} & j^* = \max\limits_{j \in N(\lambda^i)} j \\ \text{for } j \in Y \setminus \{2(n-1)(m-1)\}: & e^{i^*}_j \quad \text{with} & i^* = \max\limits_{i \in N(y^j) \setminus \{\tilde{i}\}} i. \end{array}$$

satisfying $\lambda^{\tilde{i}} \geq 0$ at equality. Linear independence of them can be shown in a similar way as in the proof of Lemma 9.3.2, therefore we omit the details. Because of the linear independence $\lambda^{\tilde{i}} \geq 0$ defines a facet. (If $\lambda^{\tilde{i}}$ lies at the upper edge we must change $\max$ to $\min$ in the determination of i^* for one $j \in Y \setminus \{2(n-1)(m-1)\}$.)

(ii) As in the proof of Lemma 9.3.2 and as in the case (i), $nm + 2(n-1)(m-1) - 2$ linearly independent vertices can be determined satisfying the corresponding nonnegativity constraint at equality.

(iii) Let $j \in Y$ and $k \in N(y^j)$ with $|N(\lambda^k)| = 1$, i.e., $N(\lambda^k) = \{j\}$. Because of Lemma 9.3.3 the inequality $\lambda^k \leq y^j$ is valid for $P^{nm}_{SOS\ 3}$. So $y^j \geq 0$ must be redundant.

$\square$

Having characterized trivial facets, the subsequent theorem states conditions for the inequality of Lemma 9.3.3 in order to be facet-defining.

Theorem 9.3.1 *Let $\emptyset \neq J \subseteq Y$, $J \neq Y$. Define $I := \{k \in \Lambda \mid N(\lambda^k) \subseteq J\}$. The inequality*

$$\sum_{k \in I} \lambda^k \leq \sum_{l \in J} y^l \tag{9.28}$$

defines a facet of $P^{nm}_{SOS\ 3}$ if and only if the following conditions are fulfilled:

(1) *Each triangle of the set J has a corresponding λ-variable in the set I, i.e.,*
$\forall p \in J: \ N(y^p) \cap I \neq \emptyset$.

(2) *The λ-variables appearing in the inequality are connected, i.e.,*
$\forall k, l \in I \ \exists i_1, ..., i_r \in I$ *with* $i_1 = k$, $i_r = l$ *, and* $N(\lambda^{i_s}) \cap N(\lambda^{i_{s+1}}) \neq \emptyset$ *for* $s = 1, ..., r-1$.

(3) *The triangles given by the set $Y \setminus J$ touch, i.e.,*
$\forall p, q \in Y \setminus J \ \exists j_1, ..., j_r \in Y \setminus J$ *with* $j_1 = p$, $j_r = q$, *and* $N(y^{j_s}) \cap N(y^{j_{s+1}}) \neq \emptyset$ *for* $s = 1, ..., r-1$.

Remark that in our case the terms ‘connect’ and ‘touch’ specify two different kinds of connection, by edges on the one hand and by vertices on the other hand. To illustrate the connection of λ-variables in condition (2), we consider Figures 9.7 and 9.6. The λ-variables marked by black points in Figure 9.7 are connected. The marked λ-variables in Figure 9.6 are not connected, since λ^{18} is isolated. Finally, we describe the term ‘touch’ for triangles in condition (3). The white triangles in Figure 9.7 do not fulfill this condition, whereas the white triangles in Figure 9.5 touch (but they are not connected).

We see that the facet-defining conditions are a bit more complicated than the ones in the case of $P^n_{SOS\ 2}$. Furthermore, it is easy to prove that the triangles given by the set J must be connected, i.e., the following condition is fulfilled

(4) The triangles given by the set J are connected, i.e., $\forall p, q \in J \ \ \exists j_1, ..., j_r \in J$ with $j_1 = p$, $j_r = q$, and $|N(y^{j_s}) \cap N(y^{j_{s+1}})| = 2$ for $s = 1, ..., r-1$.

Note that if the set I fulfills condition (2) this implies condition (4) for set J.

In the following we first show that the individual conditions are necessary and quote illustrative examples.

Let us begin with condition (1).

Lemma 9.3.5 *If the set J of Theorem 9.3.1 does not fulfill condition (1), then inequality* (9.28) *is redundant to the description of $P^{nm}_{SOS\ 3}$.*

Before proving this lemma we present an example, see Figure 9.5.

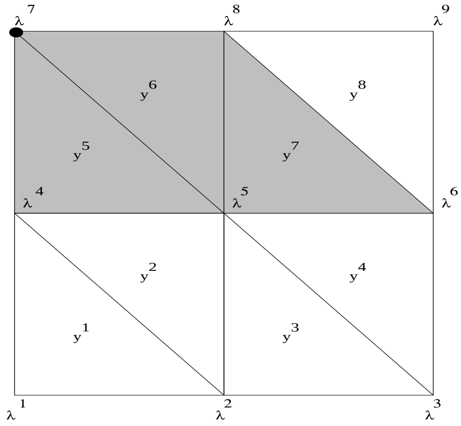

Figure 9.5: Necessity of condition (1)

Example 9.3.2 *Consider the SOS 3 polytope* $P^{3,3}_{SOS\ 3}$ *and take the set* $J = \{5, 6, 7\}$ *(grey triangles). Then we obtain the set* $I = \{7\}$ *of* λ*-variables (black point). Note that* I *and* $Y \setminus J$ *fulfill condition (2) and (3), respectively. The inequality given by* J *is redundant for* $P^{3,3}_{SOS\ 3}$*, since it is the sum of the valid inequalities*

$$0 \le y^7 \quad \textit{and} \quad \lambda^7 \le y^5 + y^6.$$

We extend this special proof to the general case.

Proof. Let $\emptyset \neq J \subseteq Y$, $J \neq Y$ be a set not fulfilling condition (1) and $I = \{k \in \Lambda \mid N(\lambda^k) \subseteq J\}$. Since (1) is not fulfilled we have

$$\exists p \in J : N(y^p) \cap I = \emptyset,$$

i. e. for all $k \in N(y^p)$ holds $N(\lambda^k) \not\subseteq J$.
Define $\tilde{J} := J \setminus \{p\}$, then $\tilde{I} = \{k \in \Lambda \mid N(\lambda^k) \subseteq \tilde{J}\} = I$ because of the assumption. Addition of the valid inequalities (see Lemma 9.3.3) $\sum_{k\in\tilde{I}} \lambda^k \le \sum_{l\in\tilde{J}} y^l$ and $0 \le y^p$ yields

$$\sum_{k\in I} \lambda^k = \sum_{k\in\tilde{I}} \lambda^k \le \sum_{l\in\tilde{J}} y^l + y^p = \sum_{l\in J} y^l.$$

□

Now we discuss the second condition.

Lemma 9.3.6 *If the set* I *of Theorem 9.3.1 does not fulfill condition (2), then inequality* (9.28) *is redundant to the description of* $P^{nm}_{SOS\ 3}$.

This time we choose $P^{5,6}_{SOS\ 3}$ for an illustrative example, see Figure 9.6.

Example 9.3.3 *The sets* $J = \{1, 2, 9, ..., 12, 17, ..., 22, 27, 28, 29\}$ *and* $I = \{1, 6, 11, 12, 18\}$ *fulfill conditions (1) and (3). The corresponding inequality results from the sum of the valid inequalities belonging to the sets* $J_1 = \{1, 2, 9, ..., 12, 17, 18, 19\}$ *and* $J_2 = \{20, 21, 22, 27, 28, 29\}$*, and thus, is redundant.*

Proof. Let $\emptyset \neq J \subseteq Y$, $J \neq Y$ be a set not fulfilling condition (2) and $I = \{k \in \Lambda \mid N(\lambda^k) \subseteq J\}$. Since (2) is not fulfilled we have

$$\begin{gathered}\exists k, l \in I\ \forall i_1, ..., i_r \in I \text{ with } i_1 = k, i_r = l \text{ there exists} \\ s \in \{1, ..., r-1\} \text{ with } N(\lambda^{i_s}) \cap N(\lambda^{i_{s+1}}) = \emptyset.\end{gathered}$$

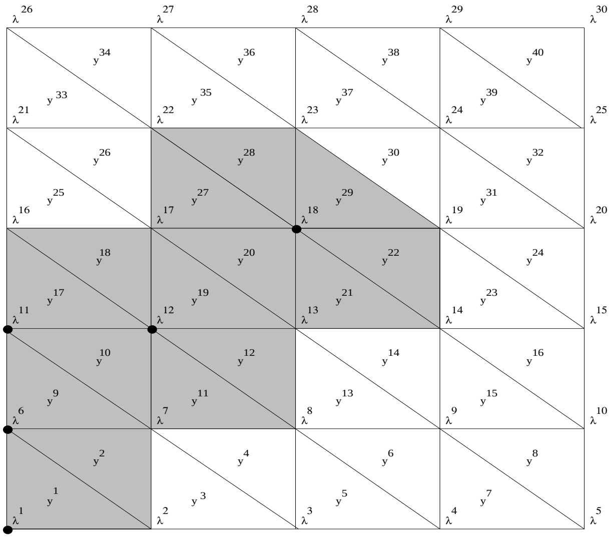

Figure 9.6: Necessity of condition (2)

We define

$$\begin{aligned} C_k &:= \{n \in I \mid \exists i_1, ..., i_r \in I \text{ with } i_1 = k, i_r = n \text{ and} \\ &\qquad N(\lambda^{i_s}) \cap N(\lambda^{i_{s+1}}) \neq \emptyset \text{ for } s = 1, ..., r-1\} \\ C_l &:= \{n \in I \mid \exists i_1, ..., i_r \in I \text{ with } i_1 = l, i_r = n \text{ and} \\ &\qquad N(\lambda^{i_s}) \cap N(\lambda^{i_{s+1}}) \neq \emptyset \text{ for } s = 1, ..., r-1\} \end{aligned}$$

which are the (maximal) sets of connected λ-variables containing the kth resp. lth λ-variable (in the above example λ^1 and λ^{18} are not connected, so we can choose $k = 1$ and $l = 18$ and get $C_1 = \{1, 6, 11, 12\}$ and $C_{18} = \{18\}$). $C_r := I \setminus (C_k \cup C_l)$ is the set of the remaining indices in I. Because of the assumption $C_k \cap C_l = \emptyset$ holds, and so $I = C_k \dot\cup C_l \dot\cup C_r$.
Further on we define

$$J_k := \bigcup_{n \in C_k} N(\lambda^n), \quad J_l := \bigcup_{n \in C_l} N(\lambda^n) \quad \text{and} \quad J_r := \bigcup_{n \in C_r} N(\lambda^n)$$

and get the corresponding sets of λ-indices

$$\begin{aligned} &I_k = \{n \in \Lambda \mid N(\lambda^n) \subseteq J_k\},\ I_l = \{n \in \Lambda \mid N(\lambda^n) \subseteq J_l\} \quad \text{and} \\ &I_r = \{n \in \Lambda \mid N(\lambda^n) \subseteq J_r\}. \end{aligned}$$

Obviously the following inclusions

$$C_k \subseteq I_k, \quad C_l \subseteq I_l \quad \text{and} \quad C_r \subseteq I_r$$

are fulfilled (One can even show that equality holds).
Addition of the valid inequalities given by the sets J_k, J_l and J_r from Lemma 9.3.3 yields

$$\sum_{n\in I_k}\lambda^n+\sum_{n\in I_l}\lambda^n+\sum_{n\in I_r}\lambda^n\leq\sum_{n\in J_k}y^n+\sum_{n\in J_l}y^n+\sum_{n\in J_r}y^n.$$

Because of the above relations the left-hand side of this inequality is greater or equal than $\sum_{n\in I}\lambda^n$.
Now we show that the following claims hold

$$\text{(a) } J_k\cup J_l\cup J_r\subseteq J\quad\text{(b) } J_k\cap J_l=\emptyset\quad\text{(c) } J_k\cap J_r=\emptyset\quad\text{(d) } J_l\cap J_r=\emptyset$$

which proves the lemma since (9.28) must be redundant.
ad (a): Let $m\in J_k\cup J_l\cup J_r$. W.l.o.g. we take $m\in J_k$. Because of definition we have

$$m\in J_k=\bigcup_{n\in C_k}N(\lambda^n)\ \Rightarrow\ \exists n\in C_k\text{ with } m\in N(\lambda^n)$$

and as $n\in C_k\subseteq I$, inclusion $N(\lambda^n)\subseteq J$ holds, hence $m\in J$.
ad (b): Suppose $J_k\cap J_l\neq\emptyset$. Let $m\in J_k\cap J_l$. Because of the definition of the sets J_k and J_l there exists $n_1\in C_k$ and $n_2\in C_l$ with $m\in N(\lambda^{n_1})$ resp. $m\in N(\lambda^{n_2})$, thus $m\in N(\lambda^{n_1})\cap N(\lambda^{n_2})\neq\emptyset$. Furthermore we find connected λ-variables from k to n_1 and from l to n_2:

$$\begin{aligned}n_1\in C_k\Rightarrow{}&\exists i_1,...,i_r\in I\text{ with } i_1=k, i_r=n_1\text{ and}\\&N(\lambda^{i_s})\cap N(\lambda^{i_{s+1}})\neq\emptyset\ \text{ for } s=1,...,r-1\\n_2\in C_l\Rightarrow{}&\exists\tilde{i}_1,...,\tilde{i}_{\tilde{r}}\in I\text{ with }\tilde{i}_1=l,\tilde{i}_{\tilde{r}}=n_2\text{ and}\\&N(\lambda^{\tilde{i}_s})\cap N(\lambda^{\tilde{i}_{s+1}})\neq\emptyset\ \text{ for } s=1,...,\tilde{r}-1.\end{aligned}$$

Putting them together we get a sequence of connected λ-variables $i_1,...,i_r,\tilde{i}_{\tilde{r}},...,\tilde{i}_1\in I$ with $i_1=k$, $\tilde{i}_1=l$ and $N(\lambda^{i_r})\cap N(\lambda^{\tilde{i}_{\tilde{r}}})\neq\emptyset$ since $i_r=n_1$ and $\tilde{i}_{\tilde{r}}=n_2$, a contradiction to our assumption.
Claim (c) and (d) can be handled similarly, there we get a contradiction to the definition of C_k or C_l. □

Finally we discuss condition (3).

Lemma 9.3.7 *If the set $Y\setminus J$ of Theorem 9.3.1 does not fulfill condition (3), then inequality* (9.28) *is redundant to the description of $P^{nm}_{SOS\ 3}$.*

Let us illustrate this condition by means of $P^{5,5}_{SOS\ 3}$, see Figure 9.7.

Example 9.3.4 *Here we see that the sets $J=\{3,\ldots,6,11,\ldots,16,18,\ldots,30\}$ and $I=\{3,8,13,14,15,17,18,21,22,23\}$ fulfill conditions (1) and (2). Adding the valid inequalities*

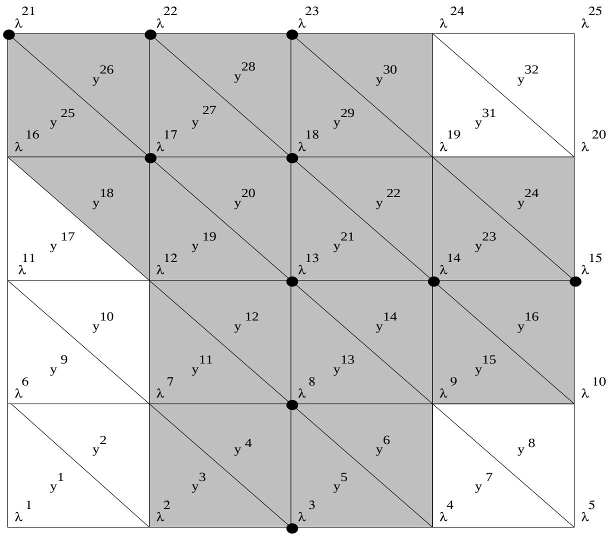

Figure 9.7: Necessity of condition (3)

resulting from the sets $J_1 = \{3, \dots, 6, 11, \dots, 16, 18, \dots, 32\}$, $J_2 = \{1, \dots, 6, 9, \dots, 30\}$ *and* $J_3 = \{3, \dots, 8, 11, \dots, 16, 18, \dots, 30\}$ *gives*

$$1 + 2\sum_{k\in I} \lambda^k \leq 1 + 2\sum_{l\in J} y^l$$

which shows the redundance of the corresponding inequality.

Proof. Let $\emptyset \neq J \subseteq Y$, $J \neq Y$ be a set not fulfilling condition (3) and $I = \{k \in \Lambda \mid N(\lambda^k) \subseteq J\}$. Since (3) is not fulfilled we have

$$\exists p, q \in Y \setminus J \;\; \forall j_1, ..., j_r \in Y \setminus J \;\text{ with } j_1 = p, j_r = q \;\text{ there exists}$$
$$s \in \{1, ..., r-1\} \text{ with } \; N(y^{j_s}) \cap N(y^{j_{s+1}}) = \emptyset.$$

We define

$$C_p := \{l \in Y \setminus J \mid \exists j_1, ..., j_r \in Y \setminus J \text{ with } j_1 = p, j_r = l \text{ and}$$
$$N(y^{j_s}) \cap N(y^{j_{s+1}}) \neq \emptyset \;\text{ for } s = 1, ..., r-1\}$$
$$C_q := \{l \in Y \setminus J \mid \exists j_1, ..., j_r \in Y \setminus J \text{ with } j_1 = q, j_r = l \text{ and}$$
$$N(y^{j_s}) \cap N(y^{j_{s+1}}) \neq \emptyset \;\text{ for } s = 1, ..., r-1\}$$

which are the (maximal) sets of touching triangles in $Y \setminus J$ containing the pth resp. qth triangle (in the above example the first and the last triangle do not touch, so we can choose $p = 1$ and $q = 32$ and get $C_1 = \{1, 2, 9, 10, 17\}$ and $C_{32} = \{31, 32\}$). $C_r := Y \setminus (J \cup C_p \cup C_q)$ is the

set of the remaining indices in $Y \setminus J$. Because of the assumption $C_p \cap C_q = \emptyset$ holds, and so $Y \setminus J = C_p \,\dot{\cup}\, C_q \,\dot{\cup}\, C_r$.
Further on we define

$$J_p := C_p \cup J, \quad J_q := C_q \cup J \quad \text{and} \quad J_r := C_r \cup J$$

and get the corresponding sets of λ-indices

$$\begin{aligned} &I_p := \{l \in \Lambda \mid N(\lambda^l) \subseteq J_p\},\ I_q := \{l \in \Lambda \mid N(\lambda^l) \subseteq J_q\} \quad \text{and} \\ &I_r := \{l \in \Lambda \mid N(\lambda^l) \subseteq J_r\}. \end{aligned}$$

Remark that $J_p \cap J_q = J_p \cap J_r = J_q \cap J_r = J$.
Addition of the valid inequalities given by the sets J_p, J_q and J_r from Lemma 9.3.3 yields

$$\sum_{l \in I_p} \lambda^l + \sum_{l \in I_q} \lambda^l + \sum_{l \in I_r} \lambda^l \leq \sum_{l \in J_p} y^l + \sum_{l \in J_q} y^l + \sum_{l \in J_r} y^l$$

As $Y \setminus J = C_p \,\dot{\cup}\, C_q \,\dot{\cup}\, C_r$ the right-hand side of this inequality equals

$$\begin{aligned} &\sum_{l \in C_p} y^l + \sum_{l \in J} y^l + \sum_{l \in C_q} y^l + \sum_{l \in J} y^l + \sum_{l \in C_r} y^l + \sum_{l \in J} y^l = 3\sum_{l \in J} y^l + \sum_{l \in Y \setminus J} y^l \\ &\quad = 2\sum_{l \in J} y^l + \sum_{l \in Y} y^l = 2\sum_{l \in J} y^l + 1. \end{aligned}$$

Now we show that the following claims hold

$$\text{(a) } I_p \cap I_q = I \quad \text{(b) } I_p \cap I_r = I \quad \text{(c) } I_q \cap I_r = I \quad \text{(d) } I_p \cup I_q \cup I_r = \Lambda.$$

Using these claims we can simplify the left-hand side of the inequality to

$$\sum_{l \in \Lambda} \lambda^l + 2\sum_{l \in I} \lambda^l = 2\sum_{l \in I} \lambda^l + 1$$

which proves the lemma since (9.28) must be redundant.
ad (a): Direction "$\supseteq$" is clear since $I \subseteq I_p, I_q$ (remember that $J \subseteq J_p, J_q$).
So let $l \in I_p \cap I_q$. Then $N(\lambda^l) \subseteq J_p \cap J_q = J$ and so $l \in I$.
Claim (b) and (c) can be handled similarly.
ad (d): Direction "$\subseteq$" is clear. So we just have to show that the union of I_p, I_q and I_r gives Λ.
Let $l \in \Lambda$. If $l \in I$ the claim follows since $I_p \cap I_q \cap I_r = I$. So let $l \in \Lambda \setminus I$. For proving that $l \in I_p \cup I_q \cup I_r$ we must show that $N(\lambda^l)$ lies entirely in one of the sets J_p, J_q or J_r implying that l must lie in the corresponding I-set.
First suppose that there exist $k_1, k_2 \in N(\lambda^l)$ with $k_1 \in J_p \setminus J = C_p$ and $k_2 \in J_q \setminus J = C_q$. Because of the defintion of the sets C_p and C_q we find touching triangles in $Y \setminus J$ from p to k_1 and from q

to k_2:

$$\begin{aligned} k_1 \in C_p \Rightarrow \exists j_1, ..., j_r \in Y \setminus J \text{ with } j_1 = p, j_r = k_1 \text{ and}\\ N(y^{j_s}) \cap N(y^{j_{s+1}}) \neq \emptyset \;\text{ for } s = 1, ..., r-1\\ k_2 \in C_q \Rightarrow \exists \tilde{j}_1, ..., \tilde{j}_{\tilde{r}} \in Y \setminus J \text{ with } \tilde{j}_1 = q, \tilde{j}_{\tilde{r}} = k_2 \text{ and}\\ N(y^{\tilde{j}_s}) \cap N(y^{\tilde{j}_{s+1}}) \neq \emptyset \;\text{ for } s = 1, ..., \tilde{r}-1. \end{aligned}$$

Putting them together at λ^l we get a sequence of touching triangles $j_1, ..., j_r, \tilde{j}_{\tilde{r}}, ..., \tilde{j}_1 \in Y \setminus J$ with $j_1 = p$, $\tilde{j}_1 = q$ and $N(y^{j_r}) \cap N(y^{\tilde{j}_{\tilde{r}}}) \neq \emptyset$ since $j_r = k_1, \tilde{j}_{\tilde{r}} = k_2 \in N(\lambda^l)$, a contradiction to our assumption.
The other cases (existence of $k_1, k_2 \in N(\lambda^l)$ with $k_1 \in C_p$ and $k_2 \in C_r$ or $k_1 \in C_q$ and $k_2 \in C_r$) can be handled similarly, there we get a contradiction to the definition of C_p or C_q. $\square$

Before we come to the proof of Theorem 9.3.1, we compare the conditions for the set J to define nontrivial facets in case of the SOS 2 polytope $P^n_{SOS\ 2}$ and the SOS 3 polytope $P^{nm}_{SOS\ 3}$. Remember that for $P^n_{SOS\ 2}$ the indices in J as well as in $Y \setminus J$ must be consecutive. This coincides with conditions (3) and (4) for $P^{nm}_{SOS\ 3}$. It is easy to see that for the SOS 2 problem where we have segments instead of triangles these conditions imply (1) and (2), the additional conditions for the SOS 3 case. If $J = \{1, ..., j\}$ or $J = \{j, ..., n-1\}$ for a $j \in Y$, then $I = \{1, ..., j\}$ or $I = \{j+1, ..., n\}$, respectively. Because of $N(y^k) = \{k, k+1\}$ the condition (1) is fulfilled. Since the indices in I are consecutive, the λ-variables are connected.

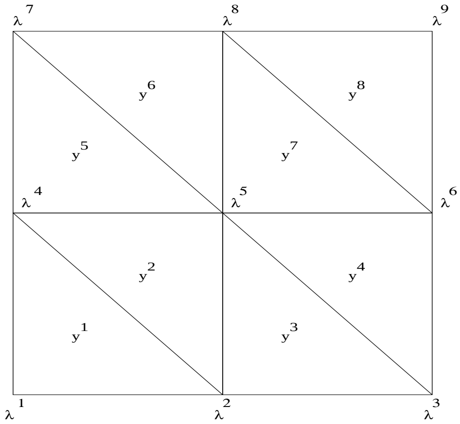

Figure 9.8: Example with $n = m = 3$

Furthermore, let us give an example. We consider the case $n = m = 3$ as depicted in Figure 9.8. Here we have 17 variables, nine λ-variables and eight y-variables, together with the corresponding formulation (9.23) - (9.27). The point $\left(0, 0, 0, 0, 0, 0, 0, \frac{1}{2}, \frac{1}{2}, 0, 0, 0, 0, \frac{1}{2}, 0, 0, \frac{1}{2}\right)$ is a vertex of the polytope $LP^{nm}_{SOS\ 3}$ defined by the linear programming relaxation, where the integrality condition of the y-variables is neglected.

It is cut off by the facet-defining inequality

$$\lambda^8 + \lambda^9 \le y^6 + y^7 + y^8$$

given by Theorem 9.3.1.
Now we give the proof of this theorem.

Proof. Let $\emptyset \neq J \subseteq Y$, $J \neq Y$ be a set fulfilling conditions (1) to (3), and let $I = \{k \in \Lambda \mid N(\lambda^k) \subseteq J\}$. Because of Lemma 9.3.3 we know that inequality (9.28) is valid for $P^{nm}_{SOS\ 3}$.
Define $F := \{(\lambda, y) \in P^{nm}_{SOS\ 3} \mid \sum_{k\in I} \lambda^k - \sum_{l\in J} y^l = 0\}$. Let $b^\top(\lambda, y) \le \beta$ be a valid inequality for $P^{nm}_{SOS\ 3}$ with $F \subseteq F_b := \{(\lambda, y) \in P^{nm}_{SOS\ 3} \mid b^\top(\lambda, y) = \beta\}$. By means of the vertices of $P^{nm}_{SOS\ 3}$ we show that b is a multiple of the coefficients of (9.28).
At first we specify for each element in J and $Y \setminus J$, respectively, vertices lying in F and thus in F_b. In the following we denote by b_{λ^k} and b_{y^l} the components of the b-vector that correspond to λ^k or y^l, respectively.
(i): $l \in Y \setminus J$
As $k \notin I$ for $k \in N(y^l)$, $e^k_l \in F$ holds. Hence,

$$b_{\lambda^k} + b_{y^l} = \beta \quad \text{for} \quad k \in N(y^l),$$

i.e., for each triangle l in $Y \setminus J$ all coefficients of λ-variables belonging to its vertices are equal to $\beta - b_{y^l}$.
(ii): $l \in J$
Because of condition (1) we have $N(y^l) \cap I \neq \emptyset$. Thus, $e^k_l \in F$ holds for $k \in N(y^l) \cap I$ and

$$b_{\lambda^k} + b_{y^l} = \beta \quad \text{for} \quad k \in N(y^l) \cap I,$$

i.e., for each triangle l in J the coefficients of λ-variables belonging to its vertices and lying in the set I are equal to $\beta - b_{y^l}$.

In the next step we use condition (2) to prove that the λ-variables in I and the y-variables in J each have the same coefficient.
Let $k, \tilde{k} \in I$. Because of condition (2) there exist $i_1, ..., i_r \in I$ with $i_1 = k$, $i_r = \tilde{k}$, and $N(\lambda^{i_s}) \cap N(\lambda^{i_{s+1}}) \neq \emptyset$ for $s = 1, ..., r-1$. Since λ^{i_s} and $\lambda^{i_{s+1}}$ have at least one common triangle $l_s \in N(\lambda^{i_s}) \cap N(\lambda^{i_{s+1}}) \subseteq J$, for which the coefficient is uniquely determined, we have by (ii) $b_{\lambda^{i_s}} = \beta - b_{y^{l_s}} = b_{\lambda^{i_{s+1}}}$ for $s = 1, ..., r-1$. Particularly, $b_{\lambda^k} = b_{\lambda^{\tilde{k}}}$ follows.
Thus, all λ-variables of the set I have the same coefficient, i.e., there exists an $a \in \mathbb{R}$ with

$$b_{\lambda^k} = a \quad \text{for} \quad k \in I,$$

and as a result of using (ii)

$$b_{y^l} = \beta - a \quad \text{for} \quad l \in J.$$

By means of condition (3) we show that the variables in $\Lambda \setminus I$ and in $Y \setminus J$ each have the same coefficient.

Let $p, q \in Y \setminus J$. Because of condition (3) there exist $j_1, ..., j_r \in Y \setminus J$ with $j_1 = p$, $j_r = q$, and $N(y^{j_s}) \cap N(y^{j_{s+1}}) \neq \emptyset$ for $s = 1, ..., r-1$. Since y^{j_s} and $y^{j_{s+1}}$ have at least one common vertex $k_s \in N(y^{j_s}) \cap N(y^{j_{s+1}}) \subseteq \Lambda \setminus I$, for which the coefficient is uniquely determined, we have by (i) $b_{y^{j_s}} = \beta - b_{\lambda^{k_s}} = b_{y^{j_{s+1}}}$ for $s = 1, ..., r-1$. In particular, $b_{y^p} = b_{y^q}$ follows.
Hence, all y-variables of the set $Y \setminus J$ have the same coefficient, i.e., there exists a $c \in \mathbb{R}$ with

$$b_{y^l} = c \quad \text{for} \quad l \in Y \setminus J,$$

and thus, by using (i)

$$b_{\lambda^k} = \beta - c \quad \text{for} \quad k \in \Lambda \setminus I.$$

Summarizing these results yields for the inequality defining F_b

$$\begin{aligned} \beta \geq b^\top(\lambda, y) &= \sum_{k \in I} b_{\lambda^k} \lambda^k + \sum_{k \in \Lambda \setminus I} b_{\lambda^k} \lambda^k + \sum_{l \in J} b_{y^l} y^l + \sum_{l \in Y \setminus J} b_{y^l} y^l \\ &= a \sum_{k \in I} \lambda^k + (\beta - c) \sum_{k \in \Lambda \setminus I} \lambda^k + (\beta - a) \sum_{l \in J} y^l + c \sum_{l \in Y \setminus J} y^l \\ &= (a - \beta + c) \sum_{k \in I} \lambda^k - (a - \beta + c) \sum_{l \in J} y^l + (\beta - c) \sum_{k \in \Lambda} \lambda^k + c \sum_{l \in Y} y^l. \end{aligned}$$

Since the equality set of $P^{nm}_{SOS\ 3}$ is given by $\sum_{k \in \Lambda} \lambda^k = \sum_{l \in Y} y^l = 1$, $b^\top(\lambda, y) = \beta$ is a multiple of $\sum_{k \in I} \lambda^k - \sum_{l \in J} y^l = 0$, and the statement that F is a facet follows. □

In the subsequent theorem we show that the facets stated so far already yield a complete linear description of $P^{nm}_{SOS\ 3}$.

Theorem 9.3.5 *The facet-defining inequalities given by Lemma 9.3.4 (i) and (ii) and by Theorem 9.3.1 together with the equalities* (9.23) *and* (9.24) *provide a complete linear description of* $P^{nm}_{SOS\ 3}$.

Proof. Let $\tilde{P}^{nm}_{SOS\ 3}$ be the polytope defined by (9.23), (9.24) and the facet-defining inequalities given by Lemma 9.3.4 (i) and (ii) and by Theorem 9.3.1.
To prove the statement we will show that $\tilde{P}^{nm}_{SOS\ 3} = P^{nm}_{SOS\ 3}$. First note that $P^{nm}_{SOS\ 3} \subseteq \tilde{P}^{nm}_{SOS\ 3}$, since the above inequalities are valid for $P^{nm}_{SOS\ 3}$. Furthermore, it is clear that $\tilde{P}^{nm}_{SOS\ 3} \subseteq LP^{nm}_{SOS\ 3}$ holds, where $LP^{nm}_{SOS\ 3}$ is the linear programming relaxation of $P^{nm}_{SOS\ 3}$. Therefore, whenever $(\lambda, y) \in \tilde{P}^{nm}_{SOS\ 3}$ and y is integer, then $(\lambda, y) \in P^{nm}_{SOS\ 3}$. We will conclude $\tilde{P}^{nm}_{SOS\ 3} \subseteq P^{nm}_{SOS\ 3}$ by proving that a vertex of $\tilde{P}^{nm}_{SOS\ 3}$ cannot have fractional y-components.

Let $(\bar{\lambda}, \bar{y})$ be a vertex of $\tilde{P}^{nm}_{SOS\ 3}$ and suppose that the point $(\bar{\lambda}, \bar{y})$ has more than one positive y-component. We will show that $(\bar{\lambda}, \bar{y})$ is a nontrivial convex combination of other points in $\tilde{P}^{nm}_{SOS\ 3}$. Hence, we obtain a contradiction, since $(\bar{\lambda}, \bar{y})$ cannot be a vertex of $\tilde{P}^{nm}_{SOS\ 3}$. Thus, for every vertex of $\tilde{P}^{nm}_{SOS\ 3}$ the y-components must be integer.

Define $S := \{k \in Y \mid \bar{y}^k > 0\}$. Because of the assumption $|S| \geq 2$ holds.
As $(\bar{\lambda}, \bar{y})$ is a vertex of $\tilde{P}^{nm}_{SOS\ 3}$, it has to satisfy at least $nm+2(n-1)(m-1)$ constraints at equality. Besides the facets given by Theorem 9.3.1, the polytope $\tilde{P}^{nm}_{SOS\ 3}$ is just defined by the equations (9.23) and (9.24) and the nonnegativity constraints (i) and (ii) of Lemma 9.3.4. Therefore, the point $(\bar{\lambda}, \bar{y})$ must fulfill at least $|S| - 1$ facets of the form (9.28) at equality and all except at most one of the y-variables in the set S can be found in these facets.
In the following we iteratively choose one of these facet-defining inequalities (9.28) defined by sets J and I and fulfilled at equality by $(\bar{\lambda}, \bar{y})$ in order to eliminate at least one positive y-variable of the vector $(\bar{\lambda}, \bar{y})$, i.e., one variable of the set S. This means that we show that the sub-vector $(\bar{\lambda}_I, \bar{y}_J)$ can be written as the conical combination

$$\begin{pmatrix} \bar{\lambda}_I \\ \bar{y}_J \end{pmatrix} = \sum_{l \in J} \sum_{k \in N(y^l)} x^k_l e^k_l$$

of points e^k_l in $\tilde{P}^{nm}_{SOS\ 3}$ with $x^k_l \geq 0$ and $\sum_{l \in J} \sum_{k \in N(y^l)} x^k_l = \sum_{k \in I} \bar{\lambda}^k = \sum_{l \in J} \bar{y}^l$. Thus, we can neglect these components of $(\bar{\lambda}, \bar{y})$ and subtract them, i.e., $(\bar{\lambda}, \bar{y}) - (\bar{\lambda}_I, \bar{y}_J)$, and continue the procedure with the remaining vector.
In each step we reduce the number of positive y-variables by at least one and after handling all possible facets there is at most one positive y-variable left. We will describe below how such an iteration works in detail, and especially, how to determine a conical combination for $(\bar{\lambda}_I, \bar{y}_J)$.

First, we consider the easier case that only one positive y-variable y^{l^*}, $l^* \in S$, is left.
The inequality

$$y^{l^*} \leq \sum_{k \in N(y^{l^*})} \lambda^k$$

is valid for $\tilde{P}^{nm}_{SOS\ 3}$. Hence, only the λ-variables belonging to the neighborhood of y^{l^*} can be positive. Since in each iteration we subtract $\sum_{k \in I} \bar{\lambda}^k = \sum_{l \in J} \bar{y}^l$, we obtain

$$\bar{y}^{l^*} = \sum_{k \in N(y^{l^*})} \bar{\lambda}^k,$$

and we can write

$$\begin{pmatrix} \bar{\lambda}_{N(y^{l^*})} \\ \bar{y}^{l^*} \end{pmatrix} = \sum_{k \in N(y^{l^*})} \bar{\lambda}^k e^k_{l^*}$$

with $e^k_{l^*} \in \tilde{P}^{nm}_{SOS\ 3}$ and $\bar{\lambda}^k \geq 0$.

Altogether we can show that $(\bar{\lambda}, \bar{y})$ is a nontrivial (since $|S| \geq 2$) convex (since $\sum_{j \in Y} y^j = \sum_{i \in \Lambda} \lambda^i = 1$) combination of points in $\tilde{P}^{nm}_{SOS\ 3}$. Hence, $(\bar{\lambda}, \bar{y})$ cannot be a vertex.

Now let us consider a facet-defining inequality (9.28) defined by the sets $J \subseteq Y$ and $I \subseteq \Lambda$ and fulfilled at equality by $(\bar{\lambda}, \bar{y})$. Before we give a detailed description of finding a conical combination of feasible points in $\tilde{P}^{nm}_{SOS\ 3}$ for $(\bar{\lambda}_I, \bar{y}_J)$, as mentioned above, we briefly map out our strategy.

At first, we show that we can assume w.l.o.g. that the set of positive $\bar{\lambda}$- and $\bar{y}$- variables in I and J, respectively, is connected. Thus, a connected graph given by these positive variables can be considered. Then we explain how to divide this connected graph into paths and non-separable subgraphs. We proceed in this is way, since we successively determine conical combinations for the variables appearing in such a path or non-separable subgraph. Given a path P of this decomposition and its associated sets P^{λ} and P^{y} of λ- and y-indices, we point out how to find a combination for $(\bar{\lambda}_{P^{\lambda}}, \bar{y}_{P^{y}})$. Additionally, we mention that all paths can be treated successively. At last, we look at the non-separable subgraphs one after the other. We show how to find a conical combination for the corresponding variables in that case. For this purpose we decompose a non-separable subgraph in a cycle with ears.

First step

We subdivide I and J in order to receive disjoint maximal connected sets of positive $\bar{\lambda}$- and $\bar{y}$-variables. So we have

$$\{i \in I \mid \bar{\lambda}^{i} > 0\} = I_1 \cup ... \cup I_p \quad \text{and} \quad J \cap S = J_1 \cup ... \cup J_p,$$

where $I_k \cap I_l = J_k \cap J_l = \emptyset$ for $k, l \in \{1, ..., p\}$ and the sets I_k and J_k, $k \in \{1, ..., p\}$, belong together, i.e., the λ-variables of I_k belong to vertices of the triangles given by J_k and vice versa. In this context ’connected’ stands for connected by positive $\bar{\lambda}$’s and $\bar{y}$’s. This means that we obtain pairs of sets I_k and J_k with

$\forall p, q \in I_k \; \exists \, i_1, ..., i_r \in I_k$ with $i_1 = p, i_r = q$ and $N(\lambda^{i_s}) \cap N(\lambda^{i_{s+1}}) \cap J_k \neq \emptyset$ for $s = 1, ..., r-1$

and

$\forall p, q \in J_k \; \exists \, j_1, ..., j_r \in J_k$ with $j_1 = p, j_r = q$ and $N(y^{j_s}) \cap N(y^{j_{s+1}}) \cap I_k \neq \emptyset$ for $s = 1, ..., r-1$.

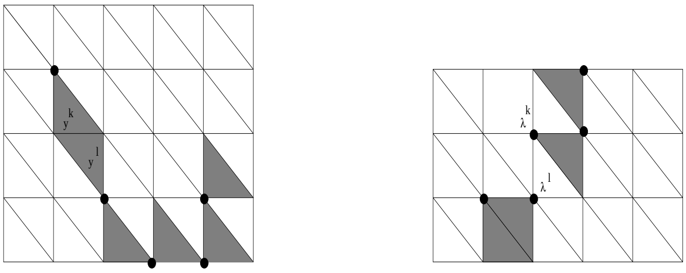

Figure 9.9: Example for non-connected sets

In Figure 9.9 we depict two examples to illustrate the term ’connected’. The positive $\bar{\lambda}$- and $\bar{y}$-variables are marked by black points and grey color, respectively. In the left picture y^k and y^l are not connected in our sense, since $\bar{\lambda}_a = 0$ holds for $a \in N(y^k) \cap N(y^l)$. Therefore, they must belong to different sets J_k and J_l. In the right picture λ^k and λ^l are not connected according to our definition, since $N(\lambda^k) \cap N(\lambda^l) \cap S = \emptyset$.

Now we prove the following claim

$$\forall k \in \{1, ..., p\} \qquad \sum_{i \in I_k} \bar{\lambda}^i = \sum_{j \in J_k} \bar{y}^j. \tag{9.29}$$

Let $k \in \{1, ..., p\}$. To prove (9.29) we first show that $\sum_{i \in I_k} \bar{\lambda}^i \leq \sum_{j \in J_k} \bar{y}^j$. This is done by deriving a valid inequality from the sets I_k and J_k.
We expand the set J_k such that the λ-variables of the set I_k are covered

$$J_k^e := \bigcup_{i \in I_k} N(\lambda^i) \supseteq J_k.$$

Because of the maximality of the set J_k, $\bar{y}^a = 0$ holds for all $a \in J_k^e \setminus J_k$.
Then we expand the set I_k in order to receive a valid inequality for $\tilde{P}^{nm}_{SOS\ 3}$

$$I_k^e := \{l \in \Lambda \mid N(\lambda^l) \subseteq J_k^e\} \supseteq I_k.$$

Here $\bar{\lambda}^a = 0$ for $a \in I_k^e \setminus I_k$ is fulfilled because of the maximality of the set I_k.
Altogether we get a valid inequality $\sum_{i \in I_k^e} \lambda^i \leq \sum_{j \in J_k^e} y^j$, where just the $\bar{\lambda}$- and $\bar{y}$-variables of the sets I_k and J_k are positive, thus $\sum_{i \in I_k} \bar{\lambda}^i \leq \sum_{j \in J_k} \bar{y}^j$.
Suppose now that $\sum_{i \in I_{k^*}} \bar{\lambda}^i < \sum_{j \in J_{k^*}} \bar{y}^j$ holds for some $k^* \in \{1, ..., p\}$. Then we obtain

$$\sum_{i \in I} \bar{\lambda}^i = \sum_{k=1}^{p} \sum_{i \in I_k} \bar{\lambda}^i < \sum_{k=1}^{p} \sum_{j \in J_k} \bar{y}^j = \sum_{j \in J} \bar{y}^j$$

which is a contradiction, since by our assumption the facet given by I and J is fulfilled at equality. Hence, equality holds for all the connected pairs of sets. Thus, it is sufficient to consider the case of finding a conical combination for positive $\bar{\lambda}$- and $\bar{y}$-variables that are connected. We can assume w.l.o.g. that the sets I and J already fulfill this condition.

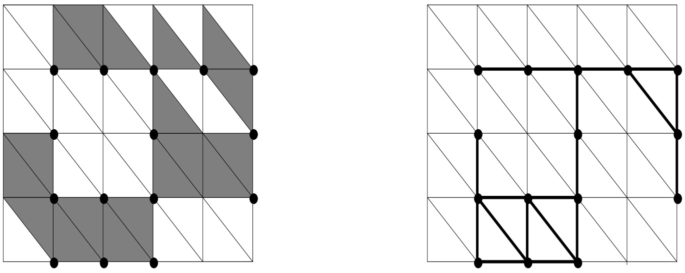

Figure 9.10: Connected graph

Second step

These positive $\bar{\lambda}$- and $\bar{y}$-variables define a connected graph G in the whole grid in the following way. All positive $\bar{\lambda}$-variables define the nodes of the graph. An edge is part of graph G, if its incident nodes belong to G and if at least one of the $\bar{y}$-variables of the adjacent triangles is positive. Figure 9.10 illustrates a connected graph G.

As mentioned in Section 2.2, we can decompose the connected graph G into non-separable subgraphs and maximal paths. Before we describe the conical combination for variables in these parts of the graph, we note two simplifications.

At first, consider a positive $\bar{y}^k$ having but one positive variable $\bar{\lambda}^l$ in its neighborhood. Here we choose $\bar{y}^k e_k^l$ and subtract it from $(\bar{\lambda}_I, \bar{y}_J)$. In this way we can eliminate all positive $\bar{y}$-variables with only one associated positive $\bar{\lambda}$-variable.

Second, for each edge having two triangles k and l with $\bar{y}^k > 0$ and $\bar{y}^l > 0$, see Figure 9.11, we can combine these two variables. We remove $\bar{y}^l$ from the set J by adding its value to $\bar{y}^k$, since concerning the conical combination the values of $\bar{\lambda}^a$ and $\bar{\lambda}^b$ can be arbitrarily distributed to $\bar{y}^k$ and $\bar{y}^l$, respectively.

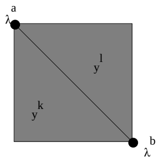

Figure 9.11: Eliminating a $\bar{y}$-variable

Third step

Now we consider the paths of the decomposition of the connected graph G and show how to find conical combinations for the variables belonging to them.

First, we examine a path having just one ‘point of contact’ with G, see Figure 9.12. Due to the above assumptions, each edge of the path has exactly one positive adjacent $\bar{y}$-variable and each triangle defining the path has exactly two positive $\bar{\lambda}$-variables. Let the indices of the path be given by

$$P^\lambda = \{i_0, i_1, ..., i_r\} \quad \text{and} \quad P^y = \{j_1, ..., j_r\},$$

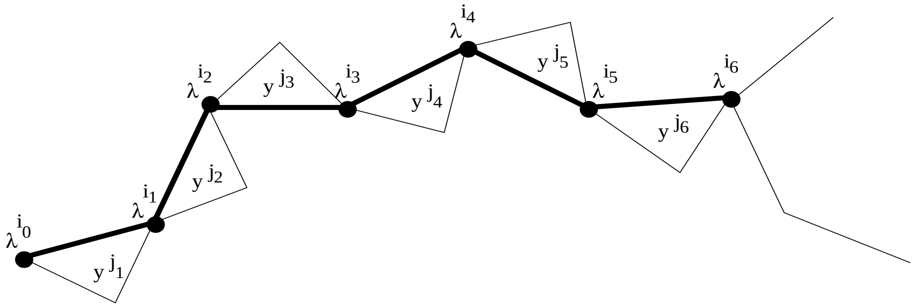

Figure 9.12: Path in decomposition

where r is the length of the path and $i_{l-1}i_l$ for $l \in \{1, ..., r\}$ are the edges of the path which are at the same time edges of the triangles y^{j_l}. Since $\bar{\lambda}^{i_0}$ has no positive neighbor except $\bar{y}^{j_1}$, we can iteratively determine a conical combination for the path variables with

$$\left(\sum_{l=0}^{s-1} \bar{\lambda}^{i_l} - \sum_{l=1}^{s-1} \bar{y}^{j_l}\right) e_{j_s}^{i_{s-1}} \quad \text{and} \quad \left(\sum_{l=1}^{s} \bar{y}^{j_l} - \sum_{l=0}^{s-1} \bar{\lambda}^{i_l}\right) e_{j_s}^{i_s}$$

for $s = 1, ..., r$. By means of this choice we can eliminate all variables of the path except the contact point variable $\bar{\lambda}^{i_r}$.

If there is a path that touches G in both endpoints λ^{i_0} and λ^{i_r}, the situation becomes a little bit more complicated, since we do not know which fraction of the variables $\bar{\lambda}^{i_0}$ and $\bar{\lambda}^{i_r}$ must be taken for the conical combination of the path variables. By deleting the path from the graph, G breaks into two components because of the assumption above that the connected graph is divided into maximal paths and non-separable subgraphs. Summing up the $\bar{y}$-variables for each component we can calculate the fraction of $\bar{\lambda}^{i_0}$ resp. $\bar{\lambda}^{i_r}$ needed by the respective component. Therefore, we can again specify a conical combination for the path variables and eliminate all except the end point variables $\bar{\lambda}^{i_0}$ and $\bar{\lambda}^{i_r}$.

We iteratively deal with all paths of the decomposition.

Fourth step

It remains to consider the non-separable subgraphs of G. As in the case of the paths, we can treat one non-separable set after another independent of the number of intersections with the remaining graph. Thus, let us outline the strategy of finding the conical combination for the variables in one non-separable subgraph G_S. By Theorem 2.2.5 of Section 2.2, we obtain an ear decomposition of G_S with

$$G_S = C \cup P_1 \cup ... \cup P_q,$$

where C is a cycle and P_{i+1} is an edge or a suspended chain having but its endnodes in common with $C \cup P_1 \cup ... \cup P_i$, see Figure 9.13. At any stage $C \cup P_1 \cup ... \cup P_i$ defines a non-separable graph.

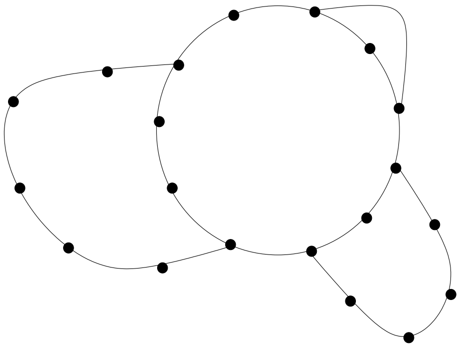

Figure 9.13: Cycle with ears

Using this presentation of G_S, we inductively construct the conical combination, where we begin with the last ear P_q. During the construction, the number of intersections of P_i with C or other ears gives us the degrees of freedom to 'tap' other ears or the cycle. Here we always tap that ear being the next in the list under consideration $P_{i-1}, ..., P_1, C$. Hence, if we make a wrong decision in the combination of λ- and y-variables for the conical combination, we notice this mistake as early as possible. Note that there are 'little' ears defined by one triangle with all adjacent $\bar{\lambda}$-variables positive. These exceptions can be treated in the same way as the following ears.

We begin with the last ear P_q which has only two intersection points with the graph at its endnodes. It is similar to a path with two contact points apart from the fact that we cannot calculate the fraction needed of the $\bar{\lambda}$-endpoint-variables, if we delete P_q from the graph G_S we still have a non-separable subgraph. Because of our assumptions each edge of the ear has exactly one positive adjacent $\bar{y}$-variable and each triangle defining the ear has exactly two positive $\bar{\lambda}$-variables. Let the indices of the ear be given by

$$P_q^{\lambda} = \{i_0, i_1, ..., i_r\} \quad \text{and} \quad P_q^{y} = \{j_1, ..., j_r\},$$

where $i_{l-1}i_l$ for $l = 1, ..., r$ are the edges of the ear, which are at the same time edges of the triangles y^{j_l}. Further, let $\bar{\lambda}^{i_0}$ belong to the intersection point with the next ear or the cycle in the list $P_{q-1}, ..., P_1, C$, that is the 'tap point'.

Now we come to the determination of the conical combination where we have to eliminate the variables $\bar{\lambda}^{i_l}$ for $l = 1, ..., r-1$ and $\bar{y}^{j_l}$ for $l = 1, ..., r$. The strategy is as follows. For the first variable $\bar{y}^{j_1}$ we take as much as we can of $\bar{\lambda}^{i_1}$ and, if it is not enough, we tap the rest of $\bar{\lambda}^{i_0}$. For an inner variable $\bar{y}^{j_k}$, $k = 2, ..., r-1$, we take the rest of $\bar{\lambda}^{i_{k-1}}$, as much as we need or can from $\bar{\lambda}^{i_k}$ and, at last, we tap $\bar{\lambda}^{i_0}$ if necessary. For the last variable $\bar{y}^{j_r}$ we completely use $\bar{\lambda}^{i_{r-1}}$, then we tap $\bar{\lambda}^{i_0}$ as much as we can, and the rest - if any - is taken from $\bar{\lambda}^{i_r}$.

Now we describe the details. We denote the fraction of $\bar{\lambda}^{i_{l-1}}$ and $\bar{\lambda}^{i_l}$ needed for $\bar{y}^{j_l}$ by $x_{j_l}^{i_{l-1}}$ and $x_{j_l}^{i_l}$, respectively. These fractions are determined iteratively so that

$$x_{j_l}^{i_{l-1}} + x_{j_l}^{i_l} = \bar{y}^{j_l} \text{ for } l = 1, ..., r, \qquad x_{j_l}^{i_l} + x_{j_{l+1}}^{i_l} = \bar{\lambda}^{i_l} \text{ for } l = 1, ..., r-1,$$
$$x_{j_1}^{i_0} \leq \bar{\lambda}^{i_0} \quad \text{and} \quad x_{j_r}^{i_r} \leq \bar{\lambda}^{i_r}$$

hold. Especially they must be 'shifted', if $\bar{\lambda}^{i_0}$ is tapped.

First we choose $x_{j_1}^{i_0}\, e_{j_1}^{i_0}$ and $x_{j_1}^{i_1}\, e_{j_1}^{i_1}$ with

$$x_{j_1}^{i_1} = \min\{\bar{y}^{j_1}, \bar{\lambda}^{i_1}\} \quad \text{and} \quad x_{j_1}^{i_0} = \bar{y}^{j_1} - x_{j_1}^{i_1}$$

for $\bar{y}^{j_1}$.

For $k \in \{2, ..., r-1\}$ we continue with $x_{j_k}^{i_{k-1}}\, e_{j_k}^{i_{k-1}}$ and $x_{j_k}^{i_k}\, e_{j_k}^{i_k}$ where

$$x_{j_k}^{i_{k-1}} = \bar{\lambda}^{i_{k-1}} - x_{j_{k-1}}^{i_{k-1}} \quad \text{and} \quad x_{j_k}^{i_k} = \min\{\bar{y}^{j_k} - x_{j_k}^{i_{k-1}}, \bar{\lambda}^{i_k}\}.$$

If $\min\{\bar{y}^{j_k} - x_{j_k}^{i_{k-1}}, \bar{\lambda}^{i_k}\} = \bar{y}^{j_k} - x_{j_k}^{i_{k-1}}$, we are done, otherwise $x_{j_k}^{i_{k-1}} + x_{j_k}^{i_k} < \bar{y}^{j_k}$ holds and we have to tap $\bar{\lambda}^{i_0}$. In that case we consider

$$M_k := \min\{\bar{\lambda}^{i_0} - x_{j_1}^{i_0}, \bar{y}^{j_k} - x_{j_k}^{i_{k-1}} - x_{j_k}^{i_k}, x_{j_1}^{i_1}, x_{j_2}^{i_2}, ..., x_{j_{k-1}}^{i_{k-1}}\},$$

where the first entry in the minimum quotes the remaining capacity of $\bar{\lambda}^{i_0}$, the second value indicates the rest needed by $\bar{y}^{j_k}$ and the other entries assure nonnegativity of the coefficients for the y-variables already handled, since we want a conical combination, and shift the coefficients set till now in the following way

$$\begin{aligned} x_{j_l}^{i_{l-1}} &= x_{j_l}^{i_{l-1}} + M_k \quad \text{and} \quad x_{j_l}^{i_l} = x_{j_l}^{i_l} - M_k \quad \text{for} \quad l = 1, ..., k-1 \\ x_{j_k}^{i_{k-1}} &= x_{j_k}^{i_{k-1}} + M_k. \end{aligned}$$

Suppose that $x_{j_k}^{i_{k-1}} + x_{j_k}^{i_k} < \bar{y}^{j_k}$ still holds after this shifting. Since we shifted the coefficients as much as possible, we obtain

$$\min\{\bar{\lambda}^{i_0} - x_{j_1}^{i_0}, \bar{y}^{j_k} - x_{j_k}^{i_{k-1}} - x_{j_k}^{i_k}, x_{j_1}^{i_1}, x_{j_2}^{i_2}, ..., x_{j_{k-1}}^{i_{k-1}}\} = 0$$

and consider two different cases.

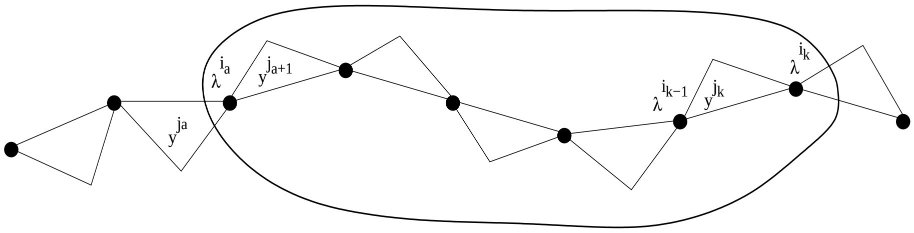

Figure 9.14: Violated inequality

First case: $\min\{x_{j_1}^{i_1}, x_{j_2}^{i_2}, ..., x_{j_{k-1}}^{i_{k-1}}\} = 0$
Let $a := \max\{l \in \{1, ..., k-1\} \mid x_{j_l}^{i_l} = 0\}$. So the combination of the variable $\bar{y}^{j_a}$ does not need a fraction of $\bar{\lambda}^{i_a}$, i.e., $\bar{\lambda}^{i_a}$ is entirely taken by $\bar{y}^{j_{a+1}}$. Then the valid inequality

$$\bar{\lambda}^{i_a} + ... + \bar{\lambda}^{i_k} \geq \bar{y}^{j_{a+1}} + ... + \bar{y}^{j_k}$$

is violated, pictured in Figure 9.14, since all $\bar{\lambda}$-variables of the left side are used up, but $\bar{y}^{j_k}$ has some rest capacity that must be covered.

Second case: $\bar{\lambda}^{i_0} = x_{j_1}^{i_0}$
Here $\bar{\lambda}^{i_0}$ is already entirely taken by $\bar{y}^{j_1}$, and therefore, cannot be tapped anymore. In that case the valid inequality

$$\bar{\lambda}^{i_0} + ... + \bar{\lambda}^{i_k} \geq \bar{y}^{j_1} + ... + \bar{y}^{j_k}$$

is violated, since all $\bar{\lambda}$-variables considered so far are used up, but $\bar{y}^{j_k}$ on the right hand side still has a fraction that must be covered.
Hence, we get a contradiction in both cases. Thus, $x_{j_k}^{i_{k-1}} + x_{j_k}^{i_k} = \bar{y}^{j_k}$ holds after shifting and we have finished.

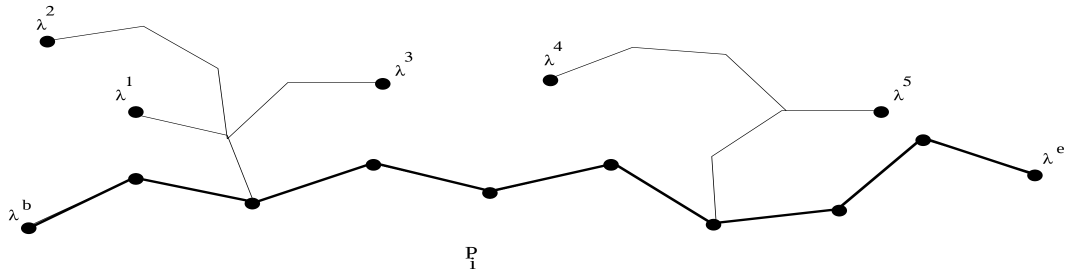

Figure 9.15: Ear with several intersection points

At last we look at $\bar{y}^r$. Here we choose $x^{i_{r-1}}_{j_r} = \bar{\lambda}^{i_{r-1}} - x^{i_{r-1}}_{j_{r-1}}$. If $x^{i_{r-1}}_{j_r} < \bar{y}^r$ holds, we tap $\bar{\lambda}^{i_0}$ as much as we can by means of the shifting process described above. If after that step $\bar{y}^r$ is still not covered, the rest must be taken of $\bar{\lambda}^{i_r}$, i.e., $x^{i_r}_{j_r} = \bar{y}^r - x^{i_{r-1}}_{j_r}$.
So we can eliminate the $\bar{y}$- and the inner $\bar{\lambda}$-variables of the ear and reduce the values of the endpoint variables by $x^{i_0}_{j_1}$ and $x^{i_r}_{j_r}$, respectively.
In the same way we can handle each ear having just the two intersections with the graph at its endnodes.

Now we consider an ear P_i having more than just the two intersection points at its ends. Here we have more liberties to tap other ears. We denote the $\bar{\lambda}$-variables belonging to the endnodes of the ear as $\bar{\lambda}^b$ and $\bar{\lambda}^e$. Further, we denote by $\bar{\lambda}^k$, $k = 1, ..., s$, the other possible tapping-variables arising from inner intersection points of the ear, see Figure 9.15. Note that such intersection points arise from ears P_k, $k > i$, already processed. Let $\bar{\lambda}^b$ belong to the (end) intersection point with the next ear in the list $P_{i-1}, ..., P_1, C$. In principle, we embark on the same strategy as above beginning at the $\bar{y}$-variable adjacent to $\bar{\lambda}^b$ except that if the current variable $\bar{y}^c$ cannot be covered by adjacent $\bar{\lambda}$-variables we have, if we have already hit an inner intersection point, the choice between several tapping variables $\bar{\lambda}^b, \bar{\lambda}^1, ..., \bar{\lambda}^j$. Here we begin the shifting process where we iteratively move as much capacity as we can and need from the tapping variable connected via an already handled ear to the next ear in the list $P_{i-1}, ..., P_1, C$. We shift as long as $\bar{y}^c$ is not covered and we know that this is possible, since otherwise we would find a violated inequality as in the case above. So we find also a conical combination for an ear with several degrees of freedom.

Having outlined these two cases, we are able to iteratively eliminate all ears including their variables.
In a final step we have to look at the cycle C. Again we can assume w.l.o.g. that each edge of the cycle has exactly one positive adjacent $\bar{y}$-variable and each triangle defining the cycle has exactly two positive $\bar{\lambda}$-variables. Note that in the case of a little cycle, i.e., a triangle where all adjacent $\bar{\lambda}$-variables are positive, it is easy to find a conical combination for the variables. So let the indices of the cycle C be given by

$$C^\lambda = \{i_1, ..., i_r\} \quad \text{and} \quad C^y = \{j_1, ..., j_r\},$$

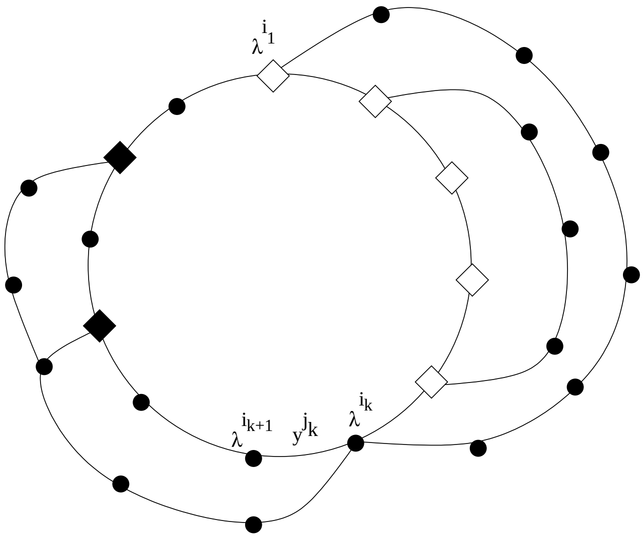

Figure 9.16: Conical combination for cycle variables

where $i_l i_{l+1}$ for $l = 1, ..., r$ (define $r + 1 = 1$) are the edges of the cycle which are at the same time edges of the triangles y^{j_l}. Obviously, $\sum_{k=1}^{r} \bar{\lambda}^{i_k} = \sum_{k=1}^{r} \bar{y}^{j_k}$ holds. We begin the construction of the conical combination at y^{j_1} and always try to cover the $\bar{y}$-variables by adjacent $\bar{\lambda}$-variables, i.e., for $k \in \{1, ..., r\}$ we choose $x^{i_k}_{j_k} \, e^{i_k}_{j_k}$ and $x^{i_{k+1}}_{j_k} \, e^{i_{k+1}}_{j_k}$ with

$$x^{i_k}_{j_k} = \min\{\bar{y}^{j_k}, \bar{\lambda}^{i_k} - x^{i_k}_{j_{k-1}}\} \quad \text{and} \quad x^{i_{k+1}}_{j_k} = \min\{\bar{y}^{j_k} - x^{i_k}_{j_k}, \bar{\lambda}^{i_{k+1}}\}.$$

Note the exceptions $x^{i_1}_{j_0} = 0$ and $x^{i_{r+1}}_{j_r} = x^{i_1}_{j_r} = \min\{\bar{y}^{j_r} - x^{i_r}_{j_r}, \bar{\lambda}^{i_1} - x^{i_1}_{j_1}\}$. If $x^{i_k}_{j_k} + x^{i_{k+1}}_{j_k} = \bar{y}^{j_k}$ holds, we are done. Otherwise, we have to tap other variables. Since we have originally a cycle with ears, we have various tapping variables and sometimes even several alternatives to transport the rest capacity of a $\bar{\lambda}$-variable depending on the complex structure of the non-separable subgraph. Figure 9.16 illustrates this situation. The tapping variables are marked by tilted squares, where a black square signifies that there exists just one way to transport the rest capacity of the variable and a white one stands for several transportation possibilities. For example, for $\bar{\lambda}^{i_1}$ in Figure 9.16 we have a path along the cycle or we transport along the big ear or we make a combination between transport along the cycle and the little ear.

We begin tapping those variables we have already considered. Starting with $\bar{\lambda}^{i_1}$ we shift the rest capacity along all possible alternatives and continue in ascending order. It is clear that we can cover $\bar{y}^{j_k}$ in this way, or else, we would find a violated inequality. If we iteratively handled all $\bar{y}^{j_k}$-variables, we know that also the $\bar{\lambda}^{i_k}$-variables are used up, and we are finished with the cycle.

This completes the conical combination of feasible points for $(\bar{\lambda}_I, \bar{y}_J)$ and the statement is proven. □

This concludes our studies of the SOS 3 polytope. Note that we did not develop a separation algorithm for the facet-defining inequalities given by Theorem 9.3.1. The reason is that instead of following this MIP approach, we incorporate the SOS condition in the branch-and-bound phase. As already mentioned, for the stationary case of gas network optimization it turned out that the implicit handling of SOS conditions yields the best running times [MMM06]. Moreover, it is not that intuitive to separate this class of facets.

In [LW01] the authors specify a separation algorithm for inequalities of the form (9.13) which correspond to our valid inequalities (9.28). For the determination of an violated inequality, an LP must be solved, and since linear programs can be solved efficiently, the separation problem can be solved in polynomial time. Note that this separation algorithm also considers redundant inequalities.

Chapter 10

Computational Results

In this chapter we report on computational results for the solution of the TTO model as described in Section 5. At first, we present the three networks which we used for testing. Then we concentrate on approximation accuracy for the linearized functions. We test three different accuracy levels and several subdivisions whereas we consider the smallest test network as prototype. In the following section, we regard test results for the classical binary approach known from the literature using CPLEX, where we suppose the coarsest accuracy. Thereafter, we start with the presentation of our developed methods. The basis is given in Section 10.4 where we quote the results based on SOS branching. These results can be compared with the binary approach of the previous section as no further algorithmic additions are used. After this, we add one feature after another to our branch-and-cut algorithm and show its benefit. First, we include the heuristic to obtain a feasible solution. Then, we improve the branching scheme using our preprocessing strategies. Thereafter, the two separation algorithms are incorporated. The first one assures runtime conditions and switching processes of the compressors without explicit modeling, the second one tackles SOS conditions. Then, we present results assuming practical conditions, i.e., a very short running time of the solution algorithm. Finally, we give a brief presentation of our developed algorithm considering some examples and conclude with some remarks in the last section. In all test runs, we use seven different sizes of the time horizon, from three hours till one day. In practice, optimization for one day is desirable.

As branch-and-cut framework we include CPLEX Version 9.0. Our described methods, such as heuristic, branching strategies, and separation algorithms, are added via so called `callback` functions that are provided within CPLEX [CPL02]. The computations were done on an AMD Athlon 64 Dual Core processor with 2.4 GHz with 4 GB main memory.

10.1 Test Networks

Our project partner, the German gas company E.ON Ruhrgas AG [RA], provided three test instances to us. These instances do not describe exactly any part of the whole Ruhrgas network, but the most relevant characteristics are involved to check the performance and accuracy of our approach. In the following, we give a brief description of these test networks.

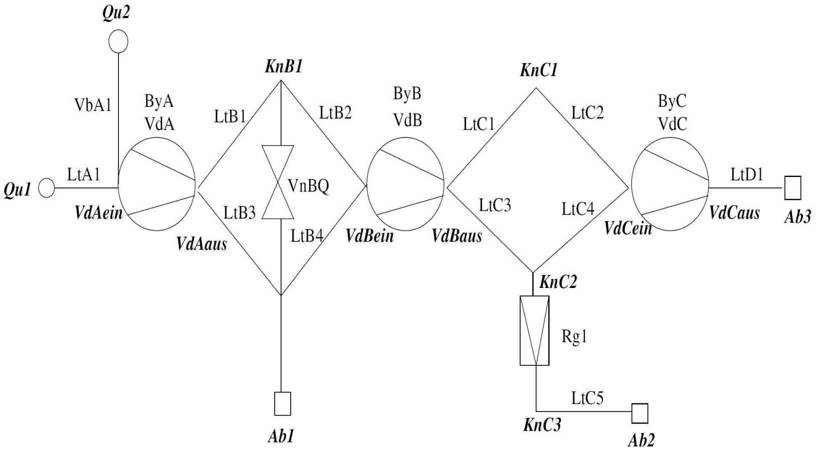

Figure 10.1: Network 1

Let us begin with the first network, the smallest one which was developed for test purposes by the Ruhrgas AG. It includes all network elements specified in Section 5 and thus can be seen as a prototype. In Figure 10.1 a picture of it is shown. As we can see, this network consists of three compressors, denoted with VdA, VdB and VdC, eleven pipes (the lines with label '$Lt*$'), and one connection $VbA1$. Furthermore, there are four ordinary valves (note that the bypass valves ByA, ByB and ByC of the compressors are not shown), and one control valve $Rg1$. Finally, the gas is delivered from two sources $Qu1$ and $Qu2$ and flows via ten innodes to the three consumers $Ab1$, $Ab2$, and $Ab3$.

The second network, the middle one, is of similar constitution. Figure 10.2 gives an illustration of it. It has several additional pipes, one additional source and twice as much consumers as network 1. So this network can be seen as an extension of the smallest one. The number of compressors is the same, but anyhow the complexity of this network is much higher due to the increased number of pipes. Remember that three nonlinear functions must be considered for each pipe in each time step. These first two test networks reflect the characteristics of a transmission network.

At last, we regard the third network, the biggest one. Figure 10.3 gives an impression of this test instance, see also [KRS00]. Note that in this network there are parallel compressors (building so called compressor stations) for example at $Elten$ or $Werne$. Even though this test network does not exactly represent a part of the Ruhrgas transmission system, it characterizes the major part of

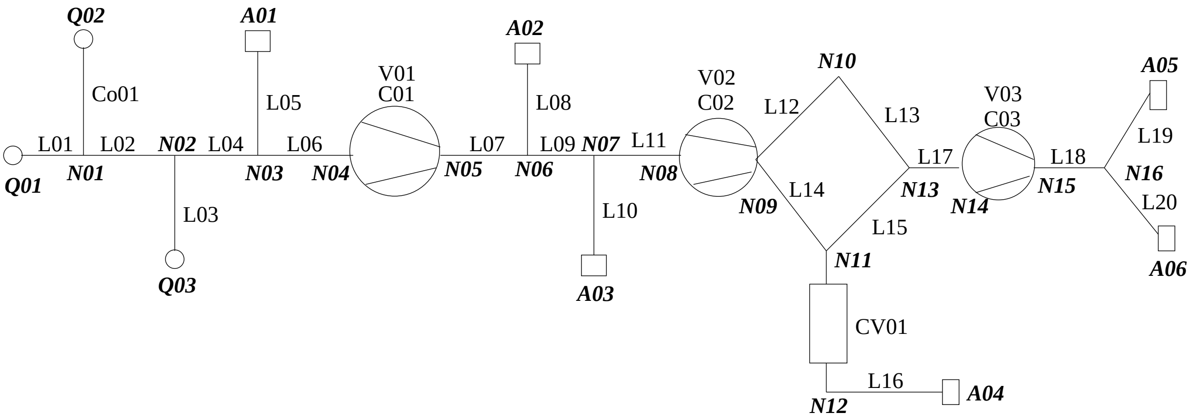

Figure 10.2: Network 2

the Ruhrgas network in Western Germany. Because of its high dimension, it poses a challenge for our optimization algorithm.
Concluding this section, we summarize the size of these networks in the following table, see Table 10.1.

Table 10.1: Size of the three test instances

Test instance	Number of							
	pipes	compr.	ord. valves	ctr. valves	connect.	sources	sinks	innodes
network 1	11	3	4	1	1	2	3	10
network 2	20	3	3	1	1	3	6	16
network 3	29	15	22	7	42	19	11	56

10.2 Approximation Accuracy

In this section we concentrate on the accuracy of the approximated nonlinear functions in our model. As already mentioned in Section 6.1, we consider three different accuracy levels $\varepsilon = 0.15$, $\varepsilon = 0.1$, and $\varepsilon = 0.05$. Remember that in our context the accuracy ε of an approximation grid means that the maximal relative approximation error in the practical relevant part of the function's domain is at most ε. Based on the first network, we test these accuracies where we regard different subdivisions of the two-dimensional grids. Finally, we choose partitions that we use for the succeeding computations.

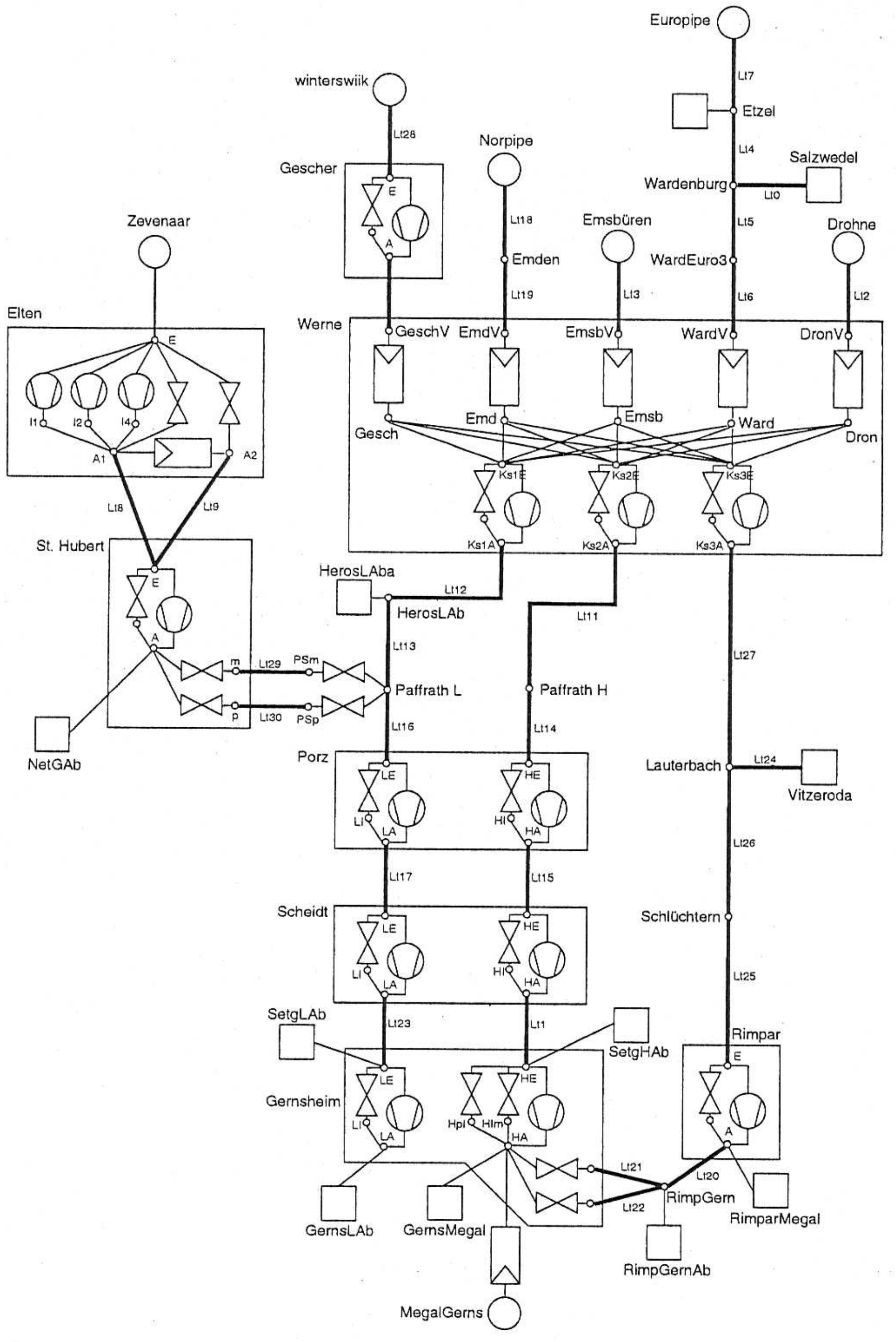

Figure 10.3: Network 3

In Section 6.1 we analyzed that the one dimensional $conti$-function (5.21) that appear in the discretized continuity equation is nearly linear. We can approximate it via one line segment and obtain a maximal relative error of 0.012 for the worst pipe. Note that in this case the SOS condition (of Type 2) is automatically fulfilled and no branching on the one-dimensional approximation grids is necessary.

In case of the two-dimensional approximation grids for the $friction$- and $impact$-function, see (5.25) and (5.26), that arise from the discretized momentum equation, we can choose between several subdivisions (see Section 6.1). Remember that the two-dimensional nonlinear functions of our model just differ by a constant factor which does not influence the relative error. Therefore, we can treat the approximation grids all at once. We determine uniform triangulations of the function domains that are given by rectangles using the software package `KARDOS`. There, we receive different partitions for the same accuracy as we can vary subdivisions in pressure and flow direction, see Figure 6.3. In Table 10.2 we list the subdivisions for the two-dimensional grids that we tested for network 1, where we consider three different partitions for $\varepsilon = 0.15$ and two ones for $\varepsilon = 0.1$ and $\varepsilon = 0.05$.

Table 10.2: Subdivisions for two-dimensional grids

Accuracy	Subdivision of		Number of	
ε	pressure	flow	points	triangles
0.15	1	9	33	40
0.15	2	8	40	54
0.15	3	7	45	64
0.1	2	9	44	60
0.1	5	8	70	108
0.05	3	16	90	136
0.05	6	15	136	224

In the first column, we find the approximation accuracy corresponding to the decomposition. Column 2 and 3 give the number of subdivisions in pressure and flow direction. The last two columns describe the number of the resulting triangles and grid points, obtained by the partition. Note that these two parameters influence our model, as for each point we have to introduce a λ-variable describing the convex combination of our approximated function value. Furthermore, the number of triangles is important in context of the branching strategies, since we have to guarantee that the SOS Type 3 condition is fulfilled, i.e., the positive λ-variables must belong to exactly one triangle (Notice that for the classical binary linearization approach, see for example Section 3.2, the number of triangles reflects the number of binary variables that are needed to formulate the approximation). Based on our computational results, see below, we will choose favorable subdivisions.

Concluding our grid considerations, we outline the three-dimensional case. Here, we determine (nonuniform) approximation grids based on Delaunay triangulation as described in Section 6.1. So we receive one grid for each accuracy level. As there are minor differences in the compressor data for the first two networks and the third network (they vary in the maximal flow allowed in the machine), we obtain different triangulations. In Table 10.3 and Table 10.4 the sizes of these grids for the first two networks and the third one are given. In Column 1, the approximation accuracy is shown. The following two columns specify the number of points and the number of tetrahedra of the grids. As we see from these tables, the difference of the triangulation sizes for the first two and the third network concerning the same accuracy level are marginal. For example, for $\varepsilon = 0.15$ the number of points is the same for all three networks, and for a compressor of the biggest network we need four tetrahedra less than for the smaller networks. Nevertheless, the structure of these approximation grids differs.

Table 10.3: Size of three-dimensional grids for network 1 and 2

Accuracy	Number of	
ε	points	tetrahedra
0.15	36	108
0.1	80	303
0.05	129	527

Table 10.4: Size of three-dimensional grids for network 3

Accuracy	Number of	
ε	points	tetrahedra
0.15	36	104
0.1	86	310
0.05	134	539

Now, we come to our results of the accuracy tests. As already mentioned, we use network 1 as prototype for these calculations. Further on, we can restrict the considered time horizon to three hours, $T = 3$, since the consequences of the grid choice can already be observed for this parameter value. Notice that this is the smallest value for the time horizon that is reasonable for an optimization process. Remember that the first time step is fixed because of the given initial state (see Section 5.4.3) and the last time step is coupled with the first one via the terminal condition (see Section 5.4.5). Therefore, we need at least three time steps to obtain optimization potential.
For an adequate evaluation of the test runs, we incorporate our developed methods as the heuristic,

the SOS branching and preprocessing, and the separation algorithm concerning switching processes in the branch-and-cut algorithm. Here, we disregard the separation for the SOS conditions as this algorithm highly depends on the triangulations, and therefore it becomes difficult to compare the different partitions for the two-dimensional grids as the size of the considered model varies a lot. As our branch-and-cut algorithm does not find an optimal solution within a reasonable running time, we restrict the solution process to one day for the test runs.

Table 10.5: Accuracy tests for network 1 with time horizon $T = 3$ and running time 24 hours

Accuracy	Subdivision of						Number of
ε	press.	flow	Nodes	Best Sol.	Lower Bd	Gap	Feas. Sol.
0.15	1	9	2688500	26.5930	17.5329	34.07%	3
0.15	2	8	3256090	23.8092	17.5329	26.36%	6
0.15	3	7	3381480	25.2633	17.5329	30.60%	4
0.1	2	9	3905324	28.1437	17.5329	37.70%	1
0.1	5	8	4656390	28.3058	17.5329	38.06%	1
0.05	3	16	3516607	28.1945	17.5329	37.81%	1
0.05	6	15	3328758	27.9782	17.5329	37.33%	1

In Table 10.5 the obtained results are summarized. The first column specifies the accuracy that must be fulfilled for all approximation grids. In Column 2 and 3, the subdivisions in pressure and flow direction for the two-dimensional grids can be found (see also Table 10.2). The fourth column indicates the number of nodes in the branch-and-bound tree. Column 5 gives the best feasible solution, and Column 6 the best lower bound. In the following column the percentage deviation of the best solution from the best lower bound is given, i.e., if UB and LB denotes the value in Column 5 and 6, respectively, then it contains the value $\frac{UB-LB}{UB}$. Here, we choose the upper bound as reference value since we suppose that the best solution found by our algorithm is near the optimal solution. Finally, the last column gives the number of feasible solutions that were found during execution of the algorithm. Note that for all test runs at least one feasible solution is determined by the simulated annealing heuristic.

At first, we consider the accuracy level $\varepsilon = 0.15$. In all cases, the branch-and-cut algorithm was able to find better feasible solutions than that given by the heuristic. For the parameters 1 and 9 for pressure and flow subdivisions two further solutions, for the parameters 2 and 8 even five feasible solutions, and for the last partition three better solutions were detected. For the partition 2 and 8, three of the five feasible solutions were found within about two hours, whereas for the other partitions it took more than nine or eleven hours before a better incumbent was found. If we look at the lower bound in the sixth column, we see that the values are the same for all partitions considered. This lower bound is reached already at the beginning of the branch-and-bound tree. But if we take the gap into account, the middle partition yields definitely the best results because

of the better feasible solution.
Now, we look at $\varepsilon = 0.1$. The results for both partitions are almost the same. The best lower bounds are equal and again determined early in the branching process. The feasible solutions that are given by our heuristic do not differ a lot due to the stochastic component involved. There are no qualitative differences for the partitions within the solution algorithm.
For accuracy $\varepsilon = 0.05$ the results are similar. The lower bounds, received early in the tree, are the same and the best feasible solutions, again given by the heuristic, do not differ a lot.
Altogether, the lower bounds for all accuracy levels are the same, and this value is already fixed at the beginning of the solution process. The branch-and-cut algorithm could not further improve this lower bound because of the SOS conditions that have to be fulfilled via branching. Only for the lowest accuracy value we could improve the results as the algorithm found better feasible solutions.

We decide to consider two accuracy levels in the following sections. We choose accuracy $\varepsilon = 0.15$ with partition values 2 and 8 as in that case we receive the best results (see Table 10.5). Since this is a coarse accuracy value, we also test accuracy level $\varepsilon = 0.05$. For this value we obtained no qualitative differences for the two considered partitions, therefore we take subdivisions 3 and 16 as these parameter values imply a smaller size of the resulting triangulation, see Table 10.2. Considering two accuracy levels we can observe how the individual methods, developed for our branch-and-cut framework, affect different approximation accuracies for our nonlinear functions. Further on, we restrict the running time to two hours when testing our features. As mentioned above, we found better feasible solutions than those given by the heuristic within this time limit for $\varepsilon = 0.15$.

10.3 Binary Approach using CPLEX

Before we consider the results given by our developed methods, we look at a binary approach for the linearization of functions as a comparison with our SOS approach. Here, we choose the classical binary approach that prevails in the literature, so to speak as benchmark, the so called lambda method. This method models the SOS condition explicitly using additional binary variables (see Chapter 3). There, a binary variable is introduced for each simplex of the triangulation (hence for each line segment, triangle or tetrahedron depending on the dimension) indicating if this simplex is chosen for the approximation of the nonlinear function. Thus exactly one of all these binary variables is positive and if so, only the λ-variables belonging to its vertices can be positive. The mathematical formulation of this approach can be found in Chapter 3.
We use this binary approach to formulate the approximation of all nonlinear functions in our model. We restrict to the lowest accuracy $\varepsilon = 0.15$ (already for this value the binary approach fails) and consider all three test networks. We choose seven different values for the time horizon, namely $T = 3, 4, 5, 6, 9, 12, 24$. We also consider these values in the following sections.
We used CPLEX Version 9.0 to solve the resulting mixed integer programs assuming default pa-

Table 10.6: Binary tests for all networks with approximation accuracy $\varepsilon = 0.15$ and running time 24 hours

Test Instance	T	Number of all var.	bin.	constr.	Nodes	Lower Bd
Network 1	3	6353	3417	4051	6640881	13.3610
	4	9160	4954	5845	1036360	18.6216
	5	11967	6491	7639	218461	23.0318
	6	14774	8028	9439	246974	27.1946
	9	23195	12639	14839	61235	35.7168
	12	31616	17250	20239	50997	46.3959
	24	65300	35694	41839	3276	54.6510
Network 2	3	10325	5385	6381	400546	1.9871
	4	15020	7902	9315	69992	2.9681
	5	19715	10419	12249	61551	2.9932
	6	24410	12936	15189	77683	2.8208
	9	38495	20487	24009	8197	3.1875
	12	52580	28038	32829	6734	3.9944
	24	108920	58242	68109	328	7.3179
Network 3	3	20474	11223	13544	912124	5.0306
	4	29116	16018	19242	304243	6.7075
	5	37758	20813	24940	197836	8.3843
	6	46400	25608	30668	56853	8.6350
	9	72326	39993	47852	10717	8.2248
	12	98252	54378	65036	2684	9.2304
	24	201956	111918	133772	37	9.4249

rameter settings. We restrict the running time to one day. In Table 10.6 we list the results. Column 1 states the test network. The second column gives the considered time horizon. The following three columns specify the size of the MIP, the number of variables, the number of binaries among them, and the constraints of the problem. Finally, the number of nodes in the branch-and-bound tree and the best lower bound are given in the last columns.

The results of the test runs for the binary approach are as expected. In none of the examples CPLEX could find a feasible solution within the given running time. This was already observed for bigger test instances in stationary gas network optimization, see [MMM06]. As we deal with the transient gas optimization problem, which means a coupling of multiple stationary models, this behavior is transferable. Furthermore, the number of nodes in the branch-and-bound tree decreases with increasing time horizon. There are only two exceptions in case of $T = 6$ for network 1 and 2, here the number of nodes increases slightly. Obviously, the best lower bound grows with increasing time horizon. But this growth is not in appropriate proportion to the increasing number of time steps, this means that for growing time horizon the fuel costs approximately increase proportional with the number of time steps, and thus also the best lower bound must be higher. For the increasing lower bounds, we find two irregularities for $T = 6$ for the second network and $T = 9$ for the third one. In these cases the lower bound slightly decreases. We cannot explain this phenomenon, maybe it is because of the growing complexity of the model. Finally, we have to stress that these lower bounds are mostly attained at the end or nearly the end of the restricted running time. Only for small values of T the best lower bound presented in the table could be reached earlier, for the small network even within some minutes. Note that we can compare the lower bounds in Table 10.6 with the results given in the next section where we use SOS branching to guarantee the SOS conditions (see Tables 10.7, 10.8 and 10.9).

10.4 SOS Branching

In this section, we begin with the presentation of our developed methods. At first, we incorporate the branching methods for the SOS conditions as described in Section 6.2 in the branch-and-cut framework given by CPLEX. Therefore, we neglect the binary variables needed for the approximation of the nonlinear functions in the previous section.

We use the following settings and conditions as standard for further computational tests. We test the accuracy levels $\varepsilon = 0.15$ and $\varepsilon = 0.05$ for all three test networks. As mentioned above, we choose $T = 3, 4, 5, 6, 9, 12, 24$ as different time horizons and restrict the running time to two hours. Furthermore, we take min-up and min-down times as well as switching processes for the compressors into account (see Section 5.4.1 and 5.4.2). At the beginning, these conditions are modeled explicitly via the constraints given in Chapter 8 (see Lemma 8.3.2 and equations (8.3)). Later, we include these conditions implicitly using a separation algorithm, see Section 10.7. We suppose four hours as minimum runtime and downtime, i.e., if a compressor is switched on, it must operate for at least four hours and if it is switched off, it must stay off for at least four hours. These values are given by the E.ON Ruhrgas AG. Unfortunately, our project partner could not provide real data concerning switching costs for compressors to us. It is very difficult to estimate these costs, especially as they influence the optimization process since they appear in the objective function (see for example [Hei02]). For our following presentation we decided to set all switching costs to zero by choosing $C^t_{e,up} = C^t_{e,down} = 0$ for $e \in E_C$ and for $t \in \mathbb{T} \setminus \{1\}$ in the objective function (5.38).

We implemented the following branching sequence in our solution algorithm. At first, we branch on the variables suggested by CPLEX. Note that our model includes only few binary variables, the switching variables for compressors and valves, but contains a big number of SOS conditions that must be fulfilled via branching strategies. Therefore, we begin with the SOS branching, if there is no more integer infeasibility. Note that this way of proceeding is reasonable, because otherwise we would for example branch on a fuel gas consumption grid of a compressor, even if the machine does not operate. After the default branching function of CPLEX, we continue with the SOS

branching (see Section 6.2). If there is any candidate for hyperplane branching we start with this strategie, since it is very efficient, especially in connection with our preprocessing techniques (see below). Thereafter we continue with variable branching to guarantee that all SOS conditions are fulfilled. A candidate for hyperplane or variable branching is chosen based on the criterion of a balanced tree, i.e., given a LP-solution, we require that the sum of the positive λ-variables in the right son is about the sum of positive λ's in the left son for the considered grid.
Further on, we add an additional feature to our SOS branching. Note that for each LP-solution we can calculate the relative approximation errors for the nonlinear functions. If this relative error is less than a certain threshold δ for an approximated function, we consider this approximation grid being feasible even if it does not satisfy the SOS condition. On the basis of several test runs we observed that $\delta = \frac{1}{2}\varepsilon$ yields a good choice for this threshold parameter, where ε is the approximation accuracy of the nonlinear function.

Table 10.7: SOS branching tests for network 1 with running time two hours

			Number of			
Accuracy	T	all var.	bin.	constr.	Nodes	Lower Bd
$\varepsilon = 0.15$	3	2972	36	584	391369	14.2115
	4	4256	50	819	179753	18.6216
	5	5540	64	1054	93664	23.0318
	6	6824	78	1295	163679	28.4229
	9	10676	120	2018	48508	41.6534
	12	14528	162	2741	13026	49.6183
	24	29936	330	5633	2951	50.2402
$\varepsilon = 0.05$	3	6009	36	584	356305	14.2115
	4	8672	50	819	209967	18.6216
	5	11335	64	1054	81855	23.0318
	6	13998	78	1295	105455	27.5668
	9	21987	120	2018	24462	41.6534
	12	29976	162	2741	5874	43.9234
	24	61932	330	5633	54	44.1449

In Table 10.7 we list the branching results for the first test network. The designation of the columns is the same as for the binary tests (see Table 10.6). Column 1 gives the approximation accuracy, and Column 2 the considered time horizon. Column 3 to 5 state the dimension of the resulting MIP. In these columns the number of all variables, the number of binaries among them, and the number of constraints are given. Finally, we find the number of nodes of the tree and the best lower bound in the two last columns.
As for the binary approach in the previous section, our SOS branching concept cannot determine

a feasible solution within the running time of two hours. This shows the complexity of the time-dependent gas optimization problem in contrast to the stationary model. In that case, where just one time step is considered, the problem could be solved to optimality, see [MMM06]. If we look at the number of nodes, we notice that it decreases with increasing planning horizon T. Only in case of $T = 5$, there is an exception. Here we receive a smaller tree, which results from the fact that for $T = 5$ it is the first time that the minimum runtime and downtime of four hours for the compressors have an effect on the model. Hence, these conditions help us to tighten the tree. If we regard the best lower bounds, we recognize increasing values with growing time horizon, as expected. If we compare these bounds for the different accuracy levels, they are almost always equal. Only for bigger planning horizons, we obtain worse bounds for the higher accuracy value. Finally, we can compare the lower bounds for $\varepsilon = 0.15$ with those given by the binary approach, see Table 10.6. In nearly all cases our SOS branching yields the same or even a better lower bound. Only for a time horizon of one day, we receive a minor value. But remember that we allowed a running time of 24 hours in the previous section in contrast to two hours for the SOS branching (we devote a longer running time to the binary approach). For a longer running time, SOS branching can improve this lower bound.

Table 10.8: SOS branching tests for network 2 with running time two hours

		Number of				
Accuracy	T	all var.	bin.	constr.	Nodes	Lower Bd
$\varepsilon = 0.15$	3	4973	33	880	69216	1.9871
	4	7164	46	1247	30235	2.9681
	5	9355	59	1614	18009	2.9932
	6	11546	72	1987	11367	2.9681
	9	18119	111	3106	6969	4.7210
	12	24692	150	4225	572	4.2844
	24	50984	306	8701	1019	7.7800
$\varepsilon = 0.05$	3	9810	33	880	122311	1.9871
	4	14280	46	1247	51231	2.9681
	5	18750	59	1614	43166	2.9932
	6	23220	72	1987	19540	2.9681
	9	36630	111	3106	1124	3.3256
	12	50040	150	4225	7328	5.8317
	24	103680	306	8701	1445	7.3223

The results for network 2 are given in Table 10.8. Obviously, our algorithm does not find a feasible solution. Regarding the lower bounds, we observe that especially for $T = 6$ the SOS branching strategie has problems to increase this best value. Note that with growing planning horizon the number of nonlinear functions in the model multiply. Thus the algorithm has much more possibilities to violate SOS conditions in a favorable way, i.e., resulting in a low objective function value. To fulfill SOS conditions a big number of branching steps is necessary, and hence, the lower bound only increases slightly. If we look at the bigger time horizons $T = 9, 12, 24$, the lower bounds become better. This is due to the min-up and min-down times for the compressors which only have a significant effect for bigger planning horizons. Comparing the lower bounds for the different accuracy levels, they are the same for a planning horizon up to six coupled time steps. Otherwise, we obtain worse values for higher approximation accuracy except for $T = 12$. In that case, a finer approximation near the LP-solution can improve the best lower bound. For an accuracy of $\varepsilon = 0.15$, we receive the same bounds up to five time steps and even stronger bounds for the other parameters in comparison to the binary approach solved by CPLEX, see Table 10.6.

Table 10.9: SOS branching tests for network 3 with running time two hours

		Number of				
Accuracy	T	all var.	bin.	constr.	Nodes	Lower Bd
$\varepsilon = 0.15$	3	9443	192	2265	154073	5.0306
	4	13364	266	3140	53312	6.7075
	5	17285	340	4015	23879	8.3843
	6	21206	414	4920	11829	9.8224
	9	32969	636	7635	2256	10.3281
	12	44732	858	10350	1221	11.4217
	24	91784	1746	21210	79	8.9028
$\varepsilon = 0.05$	3	19653	192	2265	134075	5.0306
	4	27944	266	3140	80421	6.7075
	5	36235	340	4015	45754	7.5459
	6	44526	414	4920	21061	8.3843
	9	69399	636	7635	969	6.5887
	12	94272	858	10350	428	5.9869
	24	193764	1746	21210	42	6.7018

Finally, Table 10.9 shows the results for the biggest network. They are similar to the results of the two smaller networks. The best lower bound increases slightly with growing time horizon T, whereas the number of nodes decreases. For this instance we can observe that the growing complexity of the model poses problems to the SOS branching methods. For $\varepsilon = 0.15$ and $T = 24$ and for $\varepsilon = 0.05$ and nine or more coupled time steps, we obtain worse lower bounds than for shorter planning horizons, especially for higher approximation accuracy. Note that the number of pipes and compressors increases drastically in comparison to the first test instances. Hence, we have to taken much more nonlinear functions into account that must be approximated. This means that we have to handle a bigger number of SOS conditions via our branching strategies. Thus, we see that SOS branching attains its limits.

Altogether, we can conclude that our branching strategies beat the binary approach solved with CPLEX. For an approximation accuracy of $\varepsilon = 0.15$, we obtain the same or even tighter lower bounds than in the binary approach, whereas we restrict the running time to two hours instead of one day in the binary tests. Only in two cases with a time horizon of $T = 24$, our branching methods yield worse values, which can be improved assuming longer running times. Nevertheless, we reached the limits of SOS branching, thus further features are needed to improve the results.
In the following tables concerning computational results we omit the columns specifying the dimension of the considered MIP, i.e., the number of variables, the number of binary variables , and the number of constraints, as these values do not change. Furthermore, we neglect the column giving the number of nodes in the tree as the behavior of this number is always the same, the number of nodes in the branch-and-bound tree decreases with increasing time horizon.

10.5 Heuristic

In a next step, we include the simulated annealing (SA) algorithm as heuristic in the branch-and-cut algorithm as described in Chapter 7.
Again we use our standard settings, which are the approximation accuracies $\varepsilon = 0.15$ and $\varepsilon = 0.05$, planning horizons $T = 3, 4, 5, 6, 9, 12, 24$, restricted running time of two hours, min-up and min-down times of four hours, and zero switching costs.
Regarding the SA algorithm for the gas network optimization problem, we integrate the following parameters which were tested in [Mah05]. For the relaxed continuity and momentum equations, we combine static and dynamic penalty costs in the objective function, see (7.3). Here, we choose the static cost coefficients $R_j = 100$ and the dynamic coefficients $C_j = 0.0005$ and $\alpha = 2$. For the adaptive step size, we select step size ranges $R_q = 5$ for flow variables and $R_p = 0.05$ for pressure variables, see (7.5). The initial value $\mathcal{T}_0$ of the control parameter is calculated using Dekkert's and Aart's approach see (7.6). We apply the adaptive decrement function (7.8) for the control parameter $\mathcal{T}$ using reduction parameter $\delta = 20$. Finally, the number of iterations L for each value of the control parameter $\mathcal{T}$ is computed via $L = 500 \cdot (T - 1)$, where T is the planning horizon.

In Table 10.10, we find the results for the first test network. In contrast to the tables of the previous section we delete the columns indicating the dimension of the considered MIP as well as the number of nodes (as mentioned above), and we add three columns. In Column 3, the best feasible solution given by the SA algorithm is shown. The time that the heuristic needs to determine this solution is stated in the last column. In the column before last, the gap is indicated. As mentioned above, it is calculated via $\frac{UB-LB}{UB}$, where UB is the best feasible solution and LB the best lower bound found.

Table 10.10: Simulated annealing algorithm for network 1 with running time two hours

Accuracy	T	Best Sol.	Lower Bd	Gap	Time
$\varepsilon = 0.15$	3	28.1806	14.2115	49.57%	2.69 s
	4	38.0955	18.6216	51.12%	4.73 s
	5	48.6365	23.0318	52.65%	6.92 s
	6	58.5815	28.4218	51.48%	10.60 s
	9	89.7588	41.6534	53.59%	23.88 s
	12	122.7367	49.8296	59.40%	40.88 s
	24	247.0880	61.5010	75.11%	120.81 s
$\varepsilon = 0.05$	3	28.1945	14.2115	49.60%	3.51 s
	4	38.2616	18.6216	51.33%	4.94 s
	5	49.2139	23.0318	53.20%	11.16 s
	6	58.4344	27.5976	52.77%	12.27 s
	9	89.4267	41.6534	53.42%	24.09 s
	12	121.3873	48.0180	60.44%	50.49 s
	24	245.2796	61.8004	74.80%	165.94 s

If we look at the last column of this table, we see that the SA algorithm is very fast. The running time is from some seconds for small planning horizons up to two or three minutes for a whole day. Considering the gap, it grows from about 50% for values up to $T = 6$ to 75% for 24 hours. Nevertheless, we suppose that the feasible solutions given by the heuristic are good. If we take the optimal function value in the stationary case for network 1, which is about 8.71, as a benchmark for a typical time step, the heuristic solutions seem to be reasonable. If we look at the best solution values for the different accuracies, we see that they only differ slightly due to the stochastic component involved and the different approximation grids for the fuel gas consumption. Finally, we compare the lower bounds here with the lower bounds of the previous section, given by the SOS branching strategies. We remark that in the majority of cases the heuristic solution could not help us to improve the best lower bound. Only for bigger time horizons, $T = 12, 24$, we observe that the values increase.

Now we look at the results for the second network in Table 10.11. In the last column we see that the running time for the heuristic increases for this test instance. It needs up to a minute for small planning horizons, and even up to ten minutes for a day. Especially, we recognize growing time values for the higher accuracy. Note that for the determination of the linearized function values, the heuristic has to determine the right simplex of the triangulation for each nonlinear function. Since for a higher accuracy the number of simplices in all triangulations grows, see Section 10.2, it is clear that more time is needed to find the simplices by which the nonlinear functions are approximated. Regarding the gaps, varying from 88% to about 95%, they are worse than the

Table 10.11: Simulated annealing algorithm for network 2 with running time two hours

Accuracy	T	Best Sol.	Lower Bd	Gap	Time
$\varepsilon = 0.15$	3	17.3455	1.9871	88.54%	5.15 s
	4	22.2530	2.9681	86.66%	9.56 s
	5	27.1449	2.9932	88.97%	15.59 s
	6	32.4963	2.9681	90.87%	61.85 s
	9	51.4124	4.9300	90.41%	57.20 s
	12	69.8742	4.8893	93.00%	89.55 s
	24	140.5824	7.7800	94.47%	301.46 s
$\varepsilon = 0.05$	3	16.9595	1.9871	88.28%	9.36 s
	4	23.3355	2.9681	87.28%	16.45 s
	5	28.4046	2.9932	89.46%	31.25 s
	6	33.9530	2.9681	91.26%	50.80 s
	9	51.7342	4.9300	90.47%	151.52 s
	12	70.9011	5.8189	91.79%	188.35 s
	24	143.2503	7.3223	94.89%	579.42 s

gaps for the first network. This reflects the fact that the second network poses a higher degree of difficulty as it is an extension of network 1. Compared with the stationary case we again expect that the heuristic provides quite good solutions. Considering Table 10.11 we can calculate an average objective function value of about 5.5 for one time step. For the stationary case we obtain an objective function value of about 3.52, but only two compressors operate. In the time-dependent case, all three compressors are needed to guarantee the gas transport. Therefore an average fuel gas consumption value of 5.5 for a time step is appropriate. Remark that the total transport costs for network 2 are less than the costs for the smaller network. On the one hand this is due to the fact that at the beginning of the considered time horizon just two of the three compressors work. On the other hand, we see in the picture of the second network, Figure 10.2, that the three additional consumers can be found close to the suppliers. The first one is located directly behind the sources and two other consumers are situated behind the first compressor. Further on, the minimal gas demand of all consumers is almost the same for both networks. Thus, it becomes clear that we receive less transport costs, even though the network is bigger. Notice that as in case of network 1, the heuristic solutions could not help to increase the lower bound.

Table 10.12 shows the results for network 3. Again, we observe very fast running times of the SA algorithm in the last column. Surprisingly, we receive very good results, even though the network is much bigger than the previous instances. We obtain a gap of about 1% for up to six coupled time steps for the coarser accuracy level. For growing planning horizons the gap increases. We also recognize that for the higher approximation accuracy, it becomes more difficult to improve the gap

Table 10.12: Simulated annealing algorithm for network 3 with running time two hours

Accuracy	T	Best Sol.	Lower Bd	Gap	Time
$\varepsilon = 0.15$	3	5.1203	5.0306	1.75%	4.08 s
	4	6.7953	6.7075	1.29%	5.35 s
	5	8.4642	8.3843	0.94%	6.08 s
	6	10.1661	10.0612	1.03%	7.20 s
	9	15.3002	12.3972	18.97%	15.10 s
	12	20.3615	11.0362	45.80%	14.76 s
	24	40.7211	8.4460	79.26%	41.53 s
$\varepsilon = 0.05$	3	5.0575	5.0306	0.53%	8.36 s
	4	6.8698	6.7075	2.36%	18.57 s
	5	8.5175	7.5459	11.41%	31.03 s
	6	10.3402	8.3843	18.91%	37.78 s
	9	15.3785	9.7604	36.53%	81.02 s
	12	20.5679	8.5768	58.30%	129.76 s
	24	41.1995	6.7018	83.73%	270.40 s

because of the growing complexity of the model.
These good results are due to the fact that a lot of compressors in this network are parallel, see Figure 10.3, in contrast to the smaller networks, where the compressors are in series. For optimization, it is much easier to decide which of the parallel machines must be operated than to regulate the successive compressors in an optimal manner. So it is comprehensible that the results are much better for this test instance. Considering the several time horizons, we obtain an average fuel gas consumption of about 1.7 per time step. In the stationary case, this value is about 1.68. Thus we can say that the heuristic solution is quite good. Especially for bigger planning horizons, we suppose that the feasible solution is near the optimum and the big gap results from the bad lower bound. It is amazing that we receive the lowest fuel cost for the biggest test network. These results are due to the fact that just two compressors are needed to guarantee the gas transport. Furthermore, there are some shorter pipes in this network, and the pressure offered by the sources yields a good basis for transport.

Recapitulating these results, the simulated annealing algorithm finds good feasible solutions in very short running times. For the biggest network, we receive minor gaps for planning horizons up to six coupled time steps. Otherwise, further work has to be done, to diminish the gap.

10.6 Preprocessing

In this section we want to improve the SOS branching with the preprocessing strategies described in Section 6.3. If the branching algorithm performs a hyperplane branching step, we add the induced cuts to the left and right son. Moreover, we determine λ-variables in other grids that must be zero assuming these cuts, and pass these informations to the sons. Beyond, we use informations of the input data, such as the initial state, the consumer demands, and the deliveries of the sources, to obtain further restrictions for linearizing variables of the grids.

Table 10.13: Preprocessing for network 1 with running time two hours

Accuracy	T	Best Sol.	Lower Bd	Gap
$\varepsilon = 0.15$	3	26.2979	17.5329	33.33%
	4	38.0955	21.9431	42.40%
	5	48.6365	27.3342	43.80%
	6	58.5815	31.7444	45.81%
	9	89.7588	45.9558	48.80%
	12	122.7367	54.6001	55.51%
	24	247.0880	68.2705	72.37%
$\varepsilon = 0.05$	3	28.1945	17.5329	37.81%
	4	38.2616	21.9431	42.65%
	5	49.2139	26.3532	46.45%
	6	58.4344	31.7444	45.68%
	9	89.4267	45.9558	48.61%
	12	121.3873	54.6001	55.02%
	24	245.2796	67.5922	72.44%

In Table 10.13, we list the results for the smallest test network. We have to emphasize one value in this table, namely the best solution value for $T = 3$ and accuracy $\varepsilon = 0.15$. In that case, our algorithm finds a feasible solution which is better than the solution given by the heuristic. This proves the advantage of our preprocessing strategies. Further on, the lower bounds are improved. For $T = 3$, now we obtain a gap below 40% and for the lower accuracy even 33.33% due to the better incumbent. For up to five coupled time steps, the gaps could be decreased by about 10%. For bigger planning horizons the gaps diminished by about 5%. Only if the considered time horizon comprises a day, there are just minor improvements. But altogether, we receive gaps below 50% for a time horizon up to $T = 9$.

Table 10.14 shows the results for the second network. In that case none of the heuristic solutions

Table 10.14: Preprocessing for network 2 with running time two hours

Accuracy	T	Best Sol.	Lower Bd	Gap
$\varepsilon = 0.15$	3	17.3455	4.5212	73.93%
	4	22.2530	4.5212	79.68%
	5	27.1449	5.5021	79.73%
	6	32.4963	4.4960	86.16%
	9	51.4124	6.9342	86.51%
	12	69.8742	6.6882	90.43%
	24	140.5824	8.8944	93.67%
$\varepsilon = 0.05$	3	16.9595	4.4960	73.49%
	4	23.3355	4.5212	80.63%
	5	28.4046	5.5021	80.63%
	6	33.9530	4.4960	86.76%
	9	51.7342	6.5145	87.41%
	12	70.9011	7.5211	89.39%
	24	143.2503	9.1530	93.61%

Table 10.15: Preprocessing for network 3 with running time two hours

Accuracy	T	Best Sol.	Lower Bd	Gap
$\varepsilon = 0.15$	3	5.1203	5.0306	1.75%
	4	6.7953	6.7075	1.29%
	5	8.4642	8.3843	0.94%
	6	10.1661	10.0612	1.03%
	9	15.3002	12.5765	17.80%
	12	20.3615	11.4897	43.57%
	24	40.7211	8.7314	78.56%
$\varepsilon = 0.05$	3	5.0575	5.0306	0.53%
	4	6.8698	6.7075	2.36%
	5	8.5175	7.5459	11.41%
	6	10.3402	8.3843	18.91%
	9	15.3785	10.2646	33.25%
	12	20.5679	9.7521	52.59%
	24	41.1995	6.8818	83.30%

could be improved using our preprocessing strategies. But the algorithm could tighten the gaps. For $T = 3$, there is an improvement of around 14%. For four and five time steps, the gap diminishes by around 8%. For more coupled time steps, the gap reduces just by around 4%. As in case of the smaller network, for a whole day only minor improvement can be observed. Summarizing, we obtain gaps below and around 80% for up to five coupled time steps.

Finally, we regard the results for the biggest network in Table 10.15. We do not observe any changes in the results for small planning horizons up to six hours. But with a gap of around 1% the results are already good and in case of better approximation accuracy our strategies do not help. Moreover, for higher values of T only minor improvement are received. For $\varepsilon = 0.15$ we gain around $1 - 2\%$ and for the better accuracy about $3 - 5\%$ (except $T = 24$).

Concluding these results, we can say that the preprocessing strategies improve our branch-and-cut algorithm. Note that these strategies originate from very simple and intuitive ideas, and no theoretical mathematical background is used. Nevertheless, they help us to find a better feasible solution in case of the smallest network considering three coupled times steps. Moreover, they decrease the gaps for other test runs, except for network 3, where already good results are given.

10.7 Separation for Runtime Conditions and Switching Processes

In the following, we concentrate on separation algorithms. At first, we consider runtime conditions as well as switching processes of compressors. Until now, we modeled these conditions explicitly via equations (8.3) and inequalities (8.10) to (8.13). Now the idea is to omit all these constraints in our model and to incorporate them implicitly using a separation algorithm. But here, the following difficulties arise.

If we omit the constraints modeling the runtimes and switching processes of the compressors, the start-up and shut-down variables s^t_{up} and s^t_{down} only appear in the objective function (or if we do not suppose switching costs they do not appear at all). Therefore, assuming default settings, CPLEX would eliminate these variables from the model fixing them to zero. To avoid this elimination, we must turn off CPLEX presolving. But this poses a problem. Notice that SOS branching affects the continuous λ-variables. Remember that using SOS branching, some λ-variables in each son are fixed to zero. But CPLEX itself does not branch on continuous variables, hence it adds an extra constraint of the form $\lambda^i \leq 0$, if variable λ^i is supposed to be zero in the branch. In the course of our branch-and-cut algorithm, a lot of SOS branching steps are performed. So we can imagine that adding all these (nonpositivity) constraints, the considered problems in the nodes increase drastically. If we disregard presolve techniques of CPLEX, this results in a memory capacity problem. Hence, we implement two things to handle this problems. At first, if Λ_{son} defines the set of indices of λ-variables that must be zero in a son, we pass the equation $\sum_{i \in \Lambda_{son}} \lambda^i = 0$ to the branch instead of adding all the 'nonpositivity' constraints. Furthermore, we turn preprocessing of CPLEX on. To avoid elimination of variables s^t_{up} and s^t_{down}, we continue including some of the constraints defining the switching polytope explicitly in our model. Here, we choose equations (8.3). The start-up and shut-down variables appear in these equations, hence CPLEX cannot remove these variables from the model. Therefore, we just separate constraints (8.12) and (8.13) during the solution algorithm. Notice that these cuts must be separated throughout the branch-and-bound tree.

Table 10.16: Separation of runtime conditions and switching processes for network 1 with running time two hours

Accuracy	T	Best Sol.	Lower Bd	Cuts	Gap
$\varepsilon = 0.15$	3	24.9665	17.5329	3	29.77%
	4	38.0955	21.9431	3	42.40%
	5	48.6365	27.3342	3	43.80%
	6	58.5815	31.7444	7	45.81%
	9	89.7588	45.9558	15	48.80%
	12	122.7367	54.6001	27	55.51%
	24	247.0880	68.2205	91	72.39%
$\varepsilon = 0.05$	3	28.1945	17.5329	3	37.81%
	4	38.2616	21.9431	3	42.65%
	5	49.2139	26.3532	3	46.45%
	6	58.4344	31.7444	7	45.68%
	9	89.4267	45.9558	13	48.61%
	12	121.3873	54.6001	28	55.02%
	24	245.2796	67.5371	101	72.47%

Let us consider the results for the smallest network given in Table 10.16. Based on the tables of the previous section, we insert a new column in front of the last one, where the number of cuts, added via the switching separation algorithm, is shown. We can stress one value in this table namely for $T = 3$ and accuracy $\varepsilon = 0.15$ the separation algorithm helps us to improve the best feasible solution. Remember that the heuristic solution for this setting could already be improved using SOS preprocessing, hence the branch-and-cut algorithm determines a second feasible solution. Beyond there are no considerable improvements. For $T = 24$ the lower bound slightly decreases in comparison with Table 10.13. Finally, we see that the number of violated cuts of the form (8.12) or (8.13) grows with increasing planning horizon. This results from the fact that min-up and min-down times just have an effect if a bigger time horizon is considered.

Table 10.17: Separation of runtime conditions and switching processes for network 2

Accuracy	T	Best Sol.	Lower Bd	Cuts	Gap
$\varepsilon = 0.15$	3	17.3455	4.5212	3	73.93%
	4	22.2530	4.5212	3	79.68%
	5	27.1449	5.5021	3	79.73%
	6	32.4963	4.4960	8	86.16%
	9	51.4124	6.8800	18	86.62%
	12	69.8742	6.9728	32	90.02%
	24	140.5824	9.2885	69	93.39%
$\varepsilon = 0.05$	3	16.9595	4.4960	3	73.49%
	4	23.3355	4.5212	3	80.63%
	5	28.4046	5.5021	3	80.63%
	6	33.9530	4.4960	7	86.76%
	9	51.7342	6.6838	18	87.08%
	12	70.9011	7.4107	35	89.55%
	24	143.2503	9.1704	78	93.60%

Table 10.18: Separation of runtime conditions and switching processes for network 3

Accuracy	T	Best Sol.	Lower Bd	Cuts	Gap
$\varepsilon = 0.15$	3	5.1203	5.0306	12	1.75%
	4	6.7953	6.7075	14	1.29%
	5	8.4642	8.3843	14	0.94%
	6	10.1661	10.0612	29	1.03%
	9	15.3002	12.5765	79	17.80%
	12	20.3615	11.6756	118	42.66%
	24	40.7211	8.0639	124	80.20%
$\varepsilon = 0.05$	3	5.0575	5.0306	10	0.53%
	4	6.8698	6.7075	14	2.36%
	5	8.5175	7.5459	16	11.41%
	6	10.3402	8.3843	32	18.91%
	9	15.3785	10.1410	80	34.06%
	12	20.5679	7.6352	109	62.88%
	24	41.1995	6.0895	107	85.22%

Table 10.17 shows the results for the second network. As in case of the smallest network, the implicit incorporation of runtime conditions and switching processes via a separation algorithm cannot really help us to improve the lower bounds. Here the results are even worse. Only in some cases with growing planning horizon, we gain very small improvements for the gaps.

Finally, we look at the results for network 3 in Table 10.18. As well as for the other test instances, the separation algorithm does not improve the results. For $\varepsilon = 0.05$, the results worsen for bigger planning horizons, and for $T = 12$ the gap decreases even by around 10%.

Summarizing the results of this section, we can say that this separation algorithm does not really improve the results. Only in case of network 1 we can improve the incumbent for three coupled time steps and the lower accuracy level. In the test runs, where we separate all constraints describing runtimes and switching processes, i.e., separating also equations (8.3), we receive much better results. For example for up to six coupled time steps, we obtain gaps below 40% for the first network and below 60% for the second one. For bigger planning horizons the gaps can be tighten for all three test instances. Thus, it seems to be more effective, if CPLEX just adds the constraints violated during optimization instead of including some (or all) of them explicitly, even if the number of constraints is quite small. But as aforesaid, in some cases we obtain memory capacity problems since we must turn off CPLEX presolve. Therefore, as default setting, we add equations (8.3) to the model formulation. Perhaps, in a future release of CPLEX, it is possible to branch on continuous variables just changing their bounds. Then we can benefit from this separation algorithm.

10.8 Separation for SOS Conditions

Finally, we incorporate the separation algorithm for the SOS conditions as described in Section 6.4 in the branch-and-cut framework. Remember that this algorithm results from linking two approximation grids and hence it combines the linearizing λ-variables of two SOS constraints. In our test runs, it turned out that it is the best to add the cuts in the root node only. In the stationary case of gas network optimization, these cuts are also used in the root node, see [MMM06].

Table 10.19 shows the results for the first network. Notice that in Column 5 there are two numbers for the cuts. The first number gives the number of cuts resulting from separation for SOS conditions and the second one specifies the number of cuts determined by the previous separation algorithm concerning runtimes and switching processes. At first, we remark that the better feasible solution found for $T = 3$ and $\varepsilon = 0.15$ gets lost adding the SOS cuts. As the lower bound is improved, the gap still increases slightly by around 1%. Due to the separation algorithm, the gap is improved by around $6 - 7\%$ for up to nine time steps. For twelve coupled time steps the gap even decreases by around 11%. Only for the biggest horizon, $T = 24$, the improvement is not so good. For the lower accuracy we gain around 2%. For the higher accuracy the gap increases slightly, this is because of

Table 10.19: Separation for SOS conditions for network 1 with running time two hours

Accuracy	T	Best Sol.	Lower Bd	Cuts	Gap
$\varepsilon = 0.15$	3	28.1806	19.4948	76/3	30.82%
	4	38.0955	24.8860	28/3	34.67%
	5	48.6365	30.2771	51/3	37.75%
	6	58.5815	35.6682	50/4	39.11%
	9	89.7588	51.8416	165/8	42.24%
	12	122.7367	68.0150	327/13	44.58%
	24	247.0880	74.7055	45/74	69.77%
$\varepsilon = 0.05$	3	28.1945	19.4948	69/1	30.86%
	4	38.2616	24.8860	102/3	34.96%
	5	49.2139	30.2771	151/3	38.48%
	6	58.4344	35.6682	241/6	38.96%
	9	89.4267	51.8416	371/5	42.03%
	12	121.3873	68.0150	608/8	43.97%
	24	245.2796	64.5732	144/101	73.67%

Table 10.20: Separation for SOS conditions for network 2 with running time two hours

Accuracy	T	Best Sol.	Lower Bd	Cuts	Gap
$\varepsilon = 0.15$	3	17.3455	5.6279	15/3	67.55%
	4	22.2530	9.6524	27/3	56.62%
	5	27.1449	8.4702	33/3	68.80%
	6	32.4963	13.7020	46/8	57.84%
	9	51.4124	13.6265	56/13	73.50%
	12	69.8742	12.6707	50/22	81.87%
	24	140.5824	46.2248	89/23	67.12%
$\varepsilon = 0.05$	3	16.9595	4.4960	19/3	73.49%
	4	23.3355	4.5212	31/3	80.63%
	5	28.4046	5.5021	40/3	80.63%
	6	33.9530	4.4960	48/7	86.76%
	9	51.7342	6.5187	76/18	87.40%
	12	70.9011	7.5234	104/35	89.39%
	24	143.2503	8.7960	204/75	93.86%

the time needed to determine the SOS cuts. Altogether we obtain a gap of around 30% for $T = 3$, and a gap below 40% for up to six coupled time steps. If we consider the number of SOS cuts, we see that it increases with growing planning horizon. An exception is $T = 24$, there are less cuts than for $T = 12$, but here we obtain a bigger number of cuts resulting from runtimes and switching processes. Finally, we observe that - except for a whole day - we obtain the same lower bounds for the accuracy levels. Thus, this value is not influenced by the approximation grids.

In Table 10.20 we present the results for the second network. We again see that the number of cuts increases with growing planning horizon. For the lower accuracy level $\varepsilon = 0.15$ the separation for the SOS conditions really help us to improve the results. We gain from 6% up to 28% improvement for the gaps. For accuracy level $\varepsilon = 0.05$, the lower bounds cannot be improved using SOS cuts. For $T = 9$ and $T = 24$ the gap even increases slightly because of the running time needed for the separation of the SOS cuts. Thus, for network 2 we see that the accuracy level influences the solution algorithm.

Table 10.21: Separation for SOS conditions for network 3 with running time two hours

Accuracy	T	Best Sol.	Lower Bd	Cuts	Gap	Time
$\varepsilon = 0.15$	3	5.1203	5.1203	22/1	0.00%	9.38s
	4	6.7953	6.7953	32/0	0.00%	10.95s
	5	8.4642	8.4642	43/1	0.00%	13.64s
	6	10.1661	10.1661	57/1	0.00%	18.54s
	9	15.3002	15.3002	95/1	0.00%	35.56s
	12	20.3615	20.3615	144/4	0.00%	1m:13.56s
	24	40.7211	40.7211	304/4	0.00%	11m:06.12s
$\varepsilon = 0.05$	3	5.0575	5.0306	32/11	0.53%	
	4	6.8698	6.7075	53/15	2.36%	
	5	8.5175	8.5175	86/10	0.00%	3m:07.89s
	6	10.3402	10.3402	109/23	0.00%	4m:49.99s
	9	15.3785	14.0882	143/45	8.39%	
	12	20.5679	7.3707	92/114	64.16%	
	24	41.1995	6.8863	143/101	83.29%	

The results for the biggest network are listed in Table 10.21. Note that we add a column, since in some cases the branch-and-cut algorithm yields the optimal solution of the test instance. First, we observe that for accuracy value $\varepsilon = 0.15$, all test instances are solved to optimality using SOS cuts. Remember that for up to six coupled time steps the gap was already small, but especially for $T = 24$ the gap is reduced by 80%. Obviously the time needed to solve the problems increases with growing planning horizon, see last column in in Table 10.21. As for the second network, we see that the solution algorithm depends on the accuracy value. For $\varepsilon = 0.05$ only two test instances,

$T = 5$ and $T = 6$, are solved to optimality. For $T = 9$ the gap can be improved by 25%. In all other cases the separation algorithm for the SOS conditions does not help us to improve the results. For the small planning horizons, $T = 3$ and $T = 4$, the gap is already small, but for $T = 12$ and $T = 24$ we still receive a gap of 64% and 83%, respectively.

This concludes the presentation of our features incorporated in the branch-and-cut framework of CPLEX to solve the time-dependent case of gas network optimization. Recapitulating, we can say that the addition of the SOS separation algorithm helps us to improve the lower bound.

10.9 Tests under practical Conditions

In this section, we consider our developed branch-and-cut algorithm under practical conditions, that is, we restrict the running times to 15 minutes. In a gas company the so called dispatchers operate the gas transmission network. They decide which compressors to set in and on which level the machines have to run. Further on, they switch the valves and determine the way of the gas to flow. All decisions are based on their expert knowledge. The idea of a gas network optimization tool is to support the dispatcher at work. First the dispatcher specifies the initial state of the gas network for the optimization process, and then the software tool computes an optimal operation of the gas network over a certain time horizon. Here an hourly consideration of a whole day is desirable. Knowing a good or even optimal solution, given by the optimization tool, the dispatcher can make his decisions for the next hour. Thereafter, he can restart the optimization process to receive suggestions for the gas transmission in the following steps. Obviously, such an optimization tool must yield good or optimal solutions in very short running time. In practice a running time of 15 minutes is acceptable. Therefore, we regard the branch-and-cut algorithm for the problem of TTO with respect to this restricted running time.

Table 10.22 shows the results of the solution algorithm for the smallest network after a quarter of an hour. If we compare these results with Table 10.19, where a running time of two hours is assumed, we see that we obtain the same gaps except for the biggest planning horizon $T = 24$. So the lower bounds in Table 10.19 are attained early in the optimization process, mostly, the gap is achieved within some minutes. Thus, we can present the dispatcher good feasible solution within very short running times for a time horizon up to twelve hours. Finally we remark that the results are not worse for increasing accuracy. So the approximation accuracy does not influence the solution quality.

Now, we have a look at the second network in Table 10.23. The results are similar to the results for the smaller network. With the exception of the biggest time horizons $T = 12, 24$ (and $T = 3, 9$ for accuracy $\varepsilon = 0.05$), the same lower bounds are attained as under the consideration of the longer running time, see Table 10.20. Again, the corresponding gaps are received at the beginning of the

Table 10.22: Practice test for network 1 with running time 15 minutes

Accuracy	T	Best Sol.	Lower Bd	Cuts	Gap
$\varepsilon = 0.15$	3	28.1806	19.4948	76/2	30.82%
	4	38.0955	24.8860	28/1	34.67%
	5	48.6365	30.2771	51/2	37.75%
	6	58.5815	35.6682	50/3	39.11%
	9	89.7588	51.8416	165/8	42.24%
	12	122.7367	68.0150	327/13	44.58%
	24	247.0880	58.3910	45/55	76.37%
$\varepsilon = 0.05$	3	28.1945	19.4948	69/1	30.86%
	4	38.2616	24.8860	102/3	34.96%
	5	49.2139	30.2771	151/3	38.48%
	6	58.4344	35.6682	241/5	38.96%
	9	89.4267	51.8416	371/4	42.03%
	12	121.3873	68.0150	608/6	43.97%
	24	245.2796	49.0910	144/56	79.99%

Table 10.23: Practice test for network 2 with running time 15 minutes

Accuracy	T	Best Sol.	Lower Bd	Cuts	Gap
$\varepsilon = 0.15$	3	17.3455	5.6279	15/3	67.55%
	4	22.2530	9.6524	27/3	56.62%
	5	27.1449	8.4702	33/3	68.80%
	6	32.4963	13.7020	46/8	57.84%
	9	51.4124	13.6265	56/10	73.50%
	12	69.8742	11.5232	50/20	83.51%
	24	140.5824	34.9815	89/13	75.12%
$\varepsilon = 0.05$	3	16.9595	3.8220	19/3	77.46%
	4	23.3355	4.5212	31/3	80.63%
	5	28.4046	5.5021	40/3	80.63%
	6	33.9530	4.4960	48/6	86.76%
	9	51.7342	5.7118	76/16	88.96%
	12	70.9011	5.5880	104/24	92.12%
	24	143.2503	8.7222	197/1	93.91%

solution algorithm within some minutes. For this network, the results are not as good as for the first one. For the lower accuracy value $\varepsilon = 0.15$, we receive gaps varying from $56\% - 69\%$ for up to six coupled time steps. For bigger planning horizons, we receive gaps of more than 70%. For the higher accuracy level $\varepsilon = 0.05$, the results are worse. We only obtain gaps around 80% for up to five coupled time steps. Otherwise the gaps are even worse. Nevertheless, we suppose that the best solution given by the simulated annealing algorithm is good, but our branch-and-cut algorithm has problems to improve the lower bound. In contrast to the first network, the accuracy parameter influences the solution quality.

Finally, we consider the practical tests for network 3 in Table 10.24. As for the two test networks before, the branch-and-cut algorithm terminates after 15 minutes with the same results as in Table 10.21 except for the bigger time horizons at higher accuracy level. There, the lower bounds are improved until the end of the considered running times. In the other cases, the best lower bound is determined within some minutes. Especially, we receive optimal solutions for all considered planning horizons assuming accuracy $\varepsilon = 0.15$. Thus, we are very content with the results for the biggest network as we are able to find very good or even optimal solutions for the gas network optimization problem. Furthermore, we suppose that the heuristic solution for bigger planning horizons and accuracy $\varepsilon = 0.05$ is also near the optimum. Remember that even if network 3 is the biggest test network, it poses less problems to the optimization process due to parallel compressors.

Table 10.24: Practice test for network 3 with running time 15 minutes

Accuracy	T	Best Sol.	Lower Bd	Cuts	Gap
$\varepsilon = 0.15$	3	5.1203	5.1303	22/1	0.00%
	4	6.7953	6.7953	32/0	0.00%
	5	8.4642	8.4642	43/1	0.00%
	6	10.1661	10.1661	57/1	0.00%
	9	15.3002	15.3002	95/1	0.00%
	12	20.3615	20.3615	144/4	0.00%
	24	40.7211	40.7211	304/4	0.00%
$\varepsilon = 0.05$	3	5.0575	5.0306	32/9	0.53%
	4	6.8698	6.7075	53/15	2.36%
	5	8.5175	8.5175	86/10	0.00%
	6	10.3402	10.3402	109/23	0.00%
	9	15.3785	10.5553	143/33	31.36%
	12	20.5679	5.2938	92/58	74.26%
	24	41.1995	5.5513	143/10	86.53%

10.10 Concluding Remarks

For a brief presentation of our developed features we consider one test instance for each network. We begin with network 1 in Table 10.25 for time horizon $T = 12$ and accuracy $\varepsilon = 0.05$. In the first column of this table we give the feature added to the branch-and-cut algorithm. We start with SOS branching, then we add successively the simulated annealing heuristic, the SOS preprocessing and the separation algorithms. As we see in Table 10.25 the results are improved step by step. The heuristic yields a feasible solution. Preprocessing as well as separation strategies increase the lower bound such that the gap can be reduced by around 16%.

Table 10.25: Network 1 for $T = 12$ and $\varepsilon = 0.05$

Feature	Best Sol.	Lower Bd	Gap
Branching		43.9234	
Heuristic	121.3873	48.0180	60.44%
Preprocessing	121.3873	54.6001	55.02%
Separation	121.3873	68.0150	43.97%

For the second network we consider six coupled time steps and accuracy $\varepsilon = 0.15$, see Table 10.26. In this case especially the separation strategies help to improve the lower bound. Thus, the gap of around 91% obtained by means of the heuristic solution can be decreased to about 58%.

Table 10.26: Network 2 for $T = 6$ and $\varepsilon = 0.15$

Feature	Best Sol.	Lower Bd	Gap
Branching		2.9681	
Heuristic	32.4963	2.9681	90.87%
Preprocessing	32.4963	4.4960	86.16%
Separation	32.4963	13.7020	57.84%

For the biggest network we choose a planning horizon of six time steps with accuracy $\varepsilon = 0.05$. In Table 10.27 the results are listed. Here, the preprocessing ideas cannot help to improve the lower bound. But by means of the separation strategies the heuristic solution is proven to be optimal.

Recapitulating all results, we can say that we developed an acceptable optimization tool for the gas network transmission problem. Only in some cases we can determine optimal solutions, but in

Table 10.27: Network 3 for $T = 6$ and $\varepsilon = 0.05$

Feature	Best Sol.	Lower Bd	Gap
Branching		8.3843	
Heuristic	10.3402	8.3843	18.91%
Preprocessing	10.3402	8.3843	18.91%
Separation	10.3402	10.3402	0.00%

general the algorithm yields good feasible solutions.

For network 1, the results are satisfactory with gaps below 40% for up to six time steps and around 40% for $T = 9$ and $T = 12$. For the middle network, we obtain the worst results. In that case our branch-and-cut algorithm has problems to tighten the lower bounds, especially for the higher accuracy we obtain still big gaps. In case of the biggest network, we successfully solved the transient gas network optimization problem assuming accuracy $\varepsilon = 0.15$. For the higher accuracy, we can say that we solved it for up to six coupled time steps. For bigger planning horizons, where we still receive big gaps, we suppose that the heuristic solutions are near the optimal solutions.

To check the performance of the branch-and-cut algorithm, we also started longer test runs for the smallest planning horizon of three coupled time steps, since we wanted to know if the results can be improved. But even for running times of up to several days, nothing changed at all. The algorithm neither found a better feasible solution nor the lower bound could be increased.

Furthermore, we tested longer running times for some other planning horizons. We observed that the results presented in the tables above could only be improved considering bigger planning horizons as twelve or 24 coupled time steps, or for nine time steps in case of higher approximation accuracy. In that cases the lower bound could be tightened and thus the gaps decreased. But this process of improvement always stopped after some time.

Regarding the approximation accuracy of the nonlinear functions, we see that the results for network 2 and network 3, yielded by the branch-and-cut algorithm, depend on this parameter. For the first network, the smallest one, the results do not differ for both accuracy levels. Hence, with growing complexity of the test network our solution algorithm is influenced by the accuracy parameter for the nonlinear functions.

Furthermore, the branch-and-cut algorithm has problems with the fulfillment of the SOS conditions for all nonlinear functions. Remember that in the course of the presentation of our results, the algorithm could improve the heuristic solution given by simulated annealing two times. This succeeds always for the smallest network with planning horizon $T = 3$ and accuracy $\varepsilon = 0.15$. One better solution is found adding the preprocessing strategies, another one considering the separation algorithm for runtime and switching conditions of the compressors. But in general the algorithm hardly succeeds to determine feasible solutions. Note that in order to fulfill the SOS condition for one nonlinear function in a certain time step, several branching steps are needed in general. And thus guaranteeing the SOS conditions for all nonlinear functions in all considered time steps requires a great number of branching steps, even if we improve the SOS branching via preprocessing strategies. Thus further ideas are needed to handle SOS conditions.

Chapter 11

Conclusions

In this thesis we considered the problem of Transient Technical Optimization (TTO). In TTO, the gas transmission in a network must be operated in such a way that the consumer demands are satisfied and the fuel consumption of the compressors is minimized. We proposed a mixed integer approach for this problem concentrating on time-dependent and discrete aspects, where the nonlinearities are piece-wise linear approximated via SOS conditions. Within the framework of our branch-and-cut algorithm an adequate handling of these SOS conditions was necessary. To get a feasible solution, we included a heuristic based on the idea of simulated annealing. The lower bound in our solution algorithm was increased using cutting planes resulting from theoretical studies of switching polytopes and the linking of SOS conditions.

The simulated annealing algorithm yields good solutions in very short running times. We extended the SOS branching strategies developed within the scope of the stationary gas network optimization to the transient case and supported them using additional preprocessing strategies. These preprocessing ideas manage to diminish the gap. The separation algorithm given by switching processes and runtime conditions did not actually help us to improve the lower bound. But we could not completely exploit this separation idea. Because of problems with the CPLEX preprocessing we had to include some constraints defining the switching polytope explicitly in our model. This fact worsened the results. A separation of all constraints would yield much better results. Finally, cutting planes arising from coupled SOS conditions increase the lower bound in our branch-and-cut framework.

Regarding our solution algorithm under the restricted running time of 15 minutes which is necessary for practical applications, we achieved satisfactory results. For the smaller network we obtained good solutions for planning horizons of up to twelve time steps, whereas the second network posed most problems to our solution algorithm. Surprisingly, we received very good results for the biggest network. In this case optimal solutions could also be determined, especially for a lower accuracy level all considered planning horizons are solved to optimality. From our computational results we saw that the approximation accuracy has an influence on the solution quality with growing complexity of the test network. Furthermore the branching routine hardly succeeds to fulfill all SOS conditions. We suppose that our feasible solutions are good and that the lower bounds in our algorithm can be tightened.

In our application the concept of SOS conditions attained its limits. In comparison to the stationary case of gas network optimization, where simple SOS branching sufficed to obtain optimal solutions, even improved branching strategies were not sufficient to solve the problem of TTO. This is due to the fact that there is a larger number of nonlinearities that, in addition, depend on each other. Thus, more strategies are necessary to continue the approach via SOS approximations. A first step is the consideration of adaptive grids which pay more attention to the properties of the nonlinear function, especially in the practically relevant part of the domain. Furthermore, a combination of SOS conditions with binary variables could help to improve the results. The idea is that an approximation grid is divided into some parts, hence, several simplices of the triangulation are aggregated, each one identified by a binary variable. Via these binary variables the solution process first chooses the rough region in the grid, thereafter the values are calibrated using exact SOS branching. If the subdivision is chosen in such a way that additional cuts are induced as in the case of our preprocessing strategies, this could also accelerate the algorithm.
A next step from the technical point of view is the evaluation of the feasible solutions received from our branch-and-cut algorithm using a simulation tool. Thereafter, the modeling of the gas dynamics, where we considered rough discretizations, must be adapted.

The methods developed in this thesis are also useful for other applications. Any nonlinear function can be piece-wise linear approximated using SOS conditions. For an adequate number of nonlinearities the presented methods are sufficient to solve the problem in dependence on the approximation accuracy. Besides, the concept of SOS can be improved as pointed out above.
There are a lot of practical applications including switching processes and runtime conditions for a machine, for example in the field of power production or regenerative energies. The polyhedral studies of the switching polytopes can help to speed up solution algorithms for such problems.
Furthermore, the studies of the SOS 2 and SOS 3 polytopes are interesting from a theoretical point of view.

Concluding this discussion, we can say that we presented good results for the challenging task of solving the transient gas network optimization. But still much research remains to be done in order to solve the problem of TTO.

Wie war zu Köln es doch vordem
Mit Heinzelmännchen so bequem!
Denn war man faul - man legte sich
Hin auf die Bank und pflegte sich:
Da kamen bei Nacht,
Eh man’s gedacht,
Die Männlein und schwärmten
Und klappten und lärmten,
Und rupften
Und zupften
Und hüpften und trabten
Und putzten und schabten,
Und eh ein Faulpelz noch erwacht,
War all sein Tagewerk
bereits gemacht!

August Kopisch, Die Heinzelmännchen von Köln

Bibliography

[AC00] J.M. Arroyo and A.J. Conejo. Optimal response of a thermal unit to an electricity spot market. *IEEE Transactions on Power Systems*, 15(3):1098 – 1104, 2000.

[AdBHvL86] E.H.L. Aarts, F.M.J. de Bont, E.H.A. Habers, and P.J.M. van Laarhoven. Parallel implementations of the statistical cooling algorithm. *Integration, the VLSI Journal*, 4:209 – 238, 1986.

[AK89] E.H.L. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines. A Stochastic Approach to Combinatorial Optimization and Neural Computing*. Wiley, 1989.

[Bal05] P. Bales. Hierarchische Modellierung der Eulerschen Flussgleichungen in der Gasdynamik. Master's thesis, Technische Universität Darmstadt, 2005.

[BCC93] E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed $0-1$ programs. *Mathematical Programming*, 58:295 – 324, 1993.

[Bea79] E.M.L. Beale. Branch and bound methods for mathematical programming systems. *Annals of Discrete Mathematics*, 5:201 – 219, 1979.

[BF76] E.M.L. Beale and J.J.H. Forrest. Global optimization using special ordered sets. *Mathematical Programming*, 10:52 – 69, 1976.

[BGR] BGR. Bundesanstalt für Geowissenschaften und Rohstoffe. http://www.bgr.bund.de.

[BHK06] M.K. Banda, M. Herty, and A. Klar. Gas flow in pipeline networks. *Networks and Heterogenous Media*, 1(1):41 – 56, 2006.

[BJS86] I.O. Bohachevsky, M.E. Johnson, and M.L. Stein. Generalized simulated annealing for function optimization. *Technometrics*, 28:209 – 217, 1986.

[BSRM05] C. Borraz-Sánchez and R.Z. Ríos-Mercado. A hybrid meta-heuristic approach for natural gas pipeline network optimization. In M. J. Blesa, C. Blum, A. Roli, and

M. Sampels, editors, *Hybrid Metaheuristics: Second International Workshop, HM 2005, Barcelona, Spain, August 29-30*, pages 54 – 65. Springer, 2005.

[BT70] E.M.L. Beale and J.A. Tomlin. Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. In J. Lawrence, editor, *Proceedings of the Fifth International Conference on Operations Research*, pages 447 – 454. Tavistock Publications, London, 1970.

[Car98] R. Carter. Pipeline optimization: Dynamic programming after 30 years. In *Pipeline Simulation Interest Group*, URL: www.psig.org, 1998.

[Čer85] V. Černý. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45:41 – 51, 1985.

[CGM03] K.L. Croxton, B. Gendron, and T.L. Magnanti. A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems. *Management Science*, 49:1268 – 1273, 2003.

[CL00] T. Christof and A. Löbel. `PORTA`*: POlyhedron Representation Transformation Algorithm, Version 1.3*. Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2000.

[CMMR87] A. Corana, M. Marchesi, C. Martini, and S. Ridella. Minimizing multimodal functions of continuous variables with the “simulated annealing” algorithm. *ACM Transactions on Mathematical Software*, 13:262 – 280, 1987.

[CPL02] ILOG CPLEX Division, 889 Alder Avenue, Suite 200, Incline Village, NV 89451, USA. *Using the CPLEX Callable Library*, 2002. Information available at URL http://www.cplex.com.

[DA91] A. Dekkers and E. Aarts. Global optimization and simulated annealing. *Mathematical Programming*, 50:367 – 393, 1991.

[Dan63] G.B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.

[dFJZZ05] I.R. de Farias Jr., M. Zhao, and H. Zhao. A special ordered set approach to discontinuous piecewise linear optimization. Technical report, State University of New York at Buffalo, 2005.

[Die00] R. Diestel. *Graphentheorie*. Springer, 2000.

[ELR02] B. Erdmann, J. Lang, and R. Roitzsch. KARDOS user’s guide. Technical Report ZR 02-42, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2002.

[ES03] K. Ehrhardt and M. Steinbach. Nonlinear optimization in gas networks. Technical Report ZR 03-46, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2003.

[ES05] K. Ehrhardt and M. Steinbach. Nonlinear optimization in gas networks. In H.G. Bock, E. Kostina, H.X. Phu, and R. Ranacher, editors, *Modeling, Simulation and Optimization of Complex Processes*, pages 139 – 148, Berlin - Heidelberg - New York, 2005. Springer.

[GAS] GASAG. Berliner Gaswerke AG. http://www.gasag.de.

[GJ99] E. Gawrilow and M. Joswig. `polymake`*: a Framework for Analyzing Convex Poltytopes*, 1999.

[GNRS00] R. Gollmer, M.P. Nowak, W. Römisch, and R. Schultz. Unit commitment in power generation – a basic model and some extensions. *Annals of Operations Research*, 96:167 – 189, 2000.

[Gop79] V.N. Gopal. Techniques to optimize fuel and compressor combination selection. In *American Gas Association Transmission Conference*, 1979.

[Hac02] P. Hackländer. *Integrierte Betriebsplanung von Gasversorgungsunternehmen*. PhD thesis, Bergische Universität - Gesamthochschule Wuppertal, 2002.

[Hei02] T. Heidenreich. Linearisierungen in transienten Optimierungsmodellen des Gastransports. Master’s thesis, Gerhard-Mercator-Universität Duisburg, 2002.

[Her06] S. Herrmann, 2006. Personal communication and documents.

[HNNS04] E. Handschin, F. Neise, H. Neumann, and R. Schultz. Optimal operation of dispersed generation under uncertainty using mathematical programming. Technical report, Universität Dortmund, Universität Duisburg-Essen, 2004.

[JC00] H.H. Rachford Jr. and R. Carter. Optimizing pipeline control in transient gas flow. In *Pipeline Simulation Interest Group*, URL: www.psig.org, 2000.

[Jen93] T. Jeníček. Steady-state optimization of gas transport. In *Proceedings of the 2nd International Workshop SIMONE on Innovative Approaches to Modelling and Optimal Control of Large Scale Pipeline Networks*, September 1993.

[KdFJN04] A.B. Keha, I.R. de Farias Jr., and G.L. Nemhauser. Models for representing piecewise linear cost functions. *Operations Research Letters*, 32:44 – 48, 2004.

[KJV83] S. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671 – 680, 1983.

[Kol] O. Kolb. Physikalisches Modell des Gastransports in vernetzten Pipelines. Working Paper.

[Krá93] J. Králik. Compressor stations in SIMONE. In *Proceedings of the 2nd International Workshop SIMONE on Innovative Approaches to Modelling and Optimal Control of Large Scale Pipeline Networks*, September 1993.

[KRS00] C. Kelling, K. Reith, and E. Sekirnjak. A practical approach to transient optimization for gas networks. Technical report, Ruhrgas AG, PSI AG, 2000.

[LA87] P.J.M. van Laarhoven and E.H.L. Aarts. *Simulated Annealing: Theory and Applications*. D. Reidel Publishing Company, 1987.

[LLM04] J. Lee, J. Leung, and F. Margot. Min-up/min-down polytopes. *Discrete Optimization*, 1:77 – 85, 2004.

[LP79] A. Land and S. Powell. Computer codes for problems of integer programming. *Annals of Discrete Mathematics*, 5:221 – 269, 1979.

[LP86] L. Lovász and M.D. Plummer. *Matching Theory*. Annals of Discrete Mathematics 29. North-Holland Mathematics Studies, 1986.

[LW01] J. Lee and D. Wilson. Polyhedral methods for piecewise-linear functions I: The lambda method. *Discrete Applied Mathematics*, 108:269 – 285, 2001.

[Mah05] D. Mahlke. Der Simulated Annealing Algorithmus zur transienten Optimierung von Gasnetzen. Master's thesis, Technische Universität Darmstadt, 2005.

[Mar05] P. Marcinkowski. Schaltbedingungen bei der Optimierung von Gasnetzen: Polyedrische Untersuchungen und Schnittebenen. Master's thesis, Technische Universität Darmstadt, 2005.

[MF00] Z. Michalewicz and D.B. Fogel. *How to Solve It: Modern Heuristics*. Springer, 2000.

[MHF03] M. Miki, T. Hiroyasu, and T. Fushimi. Parallel simulated annealing with adaptive neighborhood determined by GA. *IEEE International Conference on Systems, Man and Cybernetics*, 1:26 – 31, 2003.

[MM57] H.M. Markowitz and A.S. Manne. On the solution of discrete programming problems. *Econometrica*, 25:84 – 110, 1957.

[MMM06] A. Martin, M. Möller, and S. Moritz. Mixed integer models for the stationary case of gas network optimization. *Mathematical Programming*, 105:563 – 582, 2006.

[MMMar] D. Mahlke, A. Martin, and S. Moritz. A simulated annealing algorithm for transient optimization in gas networks. *Mathematical Methods of Operations Research*, to appear.

[Möl04] M. Möller. *Mixed Integer Models for the Optimisation of Gas Networks in the Stationary Case*. PhD thesis, Technische Universität Darmstadt, 2004.

[MRR$^+$53] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087 – 1092, 1953.

[Nei04] F. Neise. Transportkostenminimierung in Gasnetzen mittels nichtlinearer optimierung. Master's thesis, Universität Duisburg-Essen, 2004.

[NNR$^+$00] M.P. Nowak, R. Nürnberg, W. Römisch, R. Schultz, and M. Westphalen. Stochastic programming for power production and trading under uncertainty. Technical Report SM-DU-471, Gerhard-Mercator-Universität Duisburg, 2000. Schriftenreihe des Fachbereichs Mathematik.

[NS00] A. Nolte and R. Schrader. A note on the finite time behavior of simulated annealing. *Mathematics of Operations Research*, 25(3):476 – 484, 2000.

[NW88] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1988.

[NW03] M.P. Nowak and M. Westphalen. A linear model for transient gas flow. Technical Report STF38 S03601, SINTEF Industrial Management, Norway, 2003.

[OK96] I.H. Osman and J.P. Kelly, editors. *Meta-Heuristics: Theory and Applications*. Kluwer, 1996.

[Oos98] F. van Oostvoorn. European gas market developments - opportunities and threats -. In *Supplement to the IAEE/GEE Conference "Energy Markets, What's New?"*, pages 29 – 44, Berlin, 9-10 September 1998.

[Pad00] M. Padberg. Approximating separable nonlinear functions via mixed zero-one programs. *Operations Research Letters*, 27:1 – 5, 2000.

[PW84] K.F. Pratt and J.G. Wilson. Optimization of operation of gas transmission systems. *Transactions of the Institute of Measurement and Control*, 6(4):261 – 269, October 1984.

[PW06] Y. Pochet and L.A. Wolsey. *Production Planning by Mixed Integer Programming*. Springer, 2006.

[RA] E.ON Ruhrgas AG. http://www.eon-ruhrgas.com.

[Ree93] C.R. Reeves, editor. *Modern Heuristic Techniques for Combinatorial Problems*. Halstead Press (Wiley), 1993.

[RS01] K. Reith and E. Sekirnjak. Transiente Technische Optimierung (TTO-Basisdokument.). Technical report, PSI AG, Berlin; Ruhrgas AG, Essen, April 2001. Vertrauliche Dokumentation.

[RT05] D. Rajan and S. Takriti. Minimum up/down polytopes of the unit commitment problem with start-up costs. Technical report, IBM Research Division, 2005.

[Sch86] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1986.

[Sek98] E. Sekirnjak. *Mixed Integer Optimization for Gas Transmission and Distribution Systems*. INFORMS Meeting, Seattle, October 1998. Lecture notes.

[Sek00] E. Sekirnjak. Transiente Technische Optimierung (TTO-Prototyp). Technical report, PSI AG, Berlin, November 2000. Vertrauliche Dokumentation.

[Sek01] E. Sekirnjak. Stationäre Leitungshydraulik für lange Segmente. Technical report, PSI AG, Berlin, November 2001. Vertrauliche Dokumentation.

[Sek03] E. Sekirnjak. Transiente Leitungsmodelle für lange Segmente und Perioden. Technical report, PSI AG, Berlin, Oktober 2003. Vertrauliche Dokumentation.

[SW00] Y. Smeers and D. De Wolf. The gas transmission problem solved by an extension of the simplex algorithm. *Management Science*, 46:1454 – 1465, 2000.

[TKW00] S. Takriti, B. Krasenbrink, and L.S.-Y. Wu. Incorporating fuel constraints and electricity spot prices into the stochastic unit commitment problem. *Operations Research*, 48(2):268 – 280, 2000.

[Tom81] J.A. Tomlin. A suggested extension of special ordered sets to non-separable nonconvex programming problems. *Annals of Discrete Mathematics*, 11:359 – 370, 1981.

[Tom88] J.A. Tomlin. Special ordered sets and an application to gas supply operations planning. *Mathematical Programming*, 42:69 – 84, 1988.

[Vos93] Z. Vostrý. Transient optimization of gas transport and distribution. In *Proceedings of the 2nd International Workshop SIMONE on Innovative Approaches to Modelling and Optimal Control of Large Scale Pipeline Networks*, September 1993.

[WC00] B.W. Wah and Y.X. Chen. Optimal anytime constrained simulated annealing for constrained global optimization. In *Proceedings Sixth International Conference on Principles and Practice of Constraint Programming*, pages 425 – 440, 2000.

[Wes04] M. Westphalen. *Anwendungen der stochastischen Optimierung im Stromhandel und Gastransport*. PhD thesis, Universität Duisburg-Essen, 2004.

[Whi32] H. Whitney. Non-separable and planar graphs. *Transactions of the American Mathematical Society*, 34:339 – 362, 1932.

[Wil98] D. Wilson. *Polyhedral Methods for Piecewise-Linear Functions*. PhD thesis, University of Kentucky, 1998. Thesis only available via www.umi.com.

[Wol98] L.A. Wolsey. *Integer Programming*. Wiley, 1998.

[WSD98] S. Wright, M. Somani, and C. Ditzel. Compressor station optimization. In *Pipeline Simulation Interest Group*, Denver, Colorado, 1998.

[WW99] B.W. Wah and T. Wang. Constrained simulated annealing with applications in nonlinear continuous constrained global optimization. In *Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence*, pages 381 – 388, 1999.

[Záw] J. Záworka. Project SIMONE - achievements and running development. Institute of Information Theory and Automation, Academy of Sciences of the Czech Republik.

[Zim75] H.I. Zimmer. Calculating optimum pipeline operations. In *American Gas Association Transmission Conference*, 1975.

Akademischer Werdegang

Susanne Moritz

geboren am 24. August 1973 in Köln

1980 - 1984	Katholische Grundschule Merten, Bornheim
1984 - 1993	Max-Ernst-Gymnasium, Brühl
1993	Abitur
1993 - 2000	Studium Diplom Mathematik mit Nebenfach Informatik und Lehramt Französisch an der Universität zu Köln
August 1996	Zwischenprüfung Französisch
Februar 2000	Diplom in Mathematik
Seit April 2000	Wissenschaftliche Mitarbeiterin am Fachbereich Mathematik der Technischen Universität Darmstadt