Models and Solution Methods for the Pooling Problem

Mohammed Alfaki

Dissertation for the degree of Philosophiae Doctor (PhD)

University of Bergen
Department of Informatics
March 2012

ISBN: 978-82-308-2051-3
University of Bergen, Norway
printed version March 29, 2012

To my parents, wife,
brothers, and sisters

Scientific Environment

The research done in this thesis has been accomplished while I was a PhD-student in the Optimization group at the Department of Informatics, University of Bergen, where I have also been associated with the ICT Research school.

This thesis is part of the RAMONA project, which concerns the regularity and uncertainty analysis and management for the Norwegian gas processing and transportation system. This project is financed by the Norwegian Research Council (80%), StatoilHydro (10%), and Gassco (10%). It is a joint project between three Norwegian universities (University of Stavanger, Norwegian University of Science and Technology, and University of Bergen). One of the sub-projects, devoted to the development of flow allocation models and algorithms under complex constraints, was assigned to University of Bergen, and this thesis is part of this sub-project.

UNIVERSITY OF BERGEN
Department of Informatics

ICT

Research School in
Information and Communication Technology

Acknowledgements

First and foremost, all praise is due to Allah, who gave me the ability to complete this work. While completing my PhD thesis, I have been fortunate to work with many skilled and encouraging people. It gives me great pleasure to acknowledge those many people who have influenced my thinking in some way and contributed to my often inadequate knowledge. I have invariably learned from all my supervisors, colleagues and friends.

I am very grateful to my supervisor, Professor Dag Haugland, for his excellent guidance and the time he afforded me to comment on my ideas and writing. My knowledge has benefited greatly from his expertise, enthusiasm and encouragement. I was fortunate to work under his guidance during my MSc and PhD studies. I also wish to express my warmest gratitude to my co-supervisor, Professor Trond Steihaug, for his numerous invaluable suggestions during this work. Special thanks go to my co-authors, Dr. Lennart Frimannslund and Dr. Mohamed El Ghami, for sharing valuable information and the discussions that contributed in various ways to this work. Warm thanks go to my father-in-law, Dr. Tagelsir Mohammed Suleiman, for reading the thesis and making valuable suggestions regarding the manuscript. I would also like to thank the former members of the Optimization group, Dr. Conrado Borraz-Sánchez and Dr. Geir Gundersen, with whom I have shared offices; they have been wonderful friends. My sincere gratitude goes to Professor Emeritus Sverre Storøy and Dr. Joanna Bauer. My thanks also go to the administration staff at the Department of Informatics, especially to the former Heads of Administration, Signe Knappskog and Ida Holen, for their help in many practicalities. I would like to extend my gratitude to Marta Lopez, Tor Bastiansen, Liljan Myhr and Steinar Heldal. I am enormously grateful to the Norwegian Research Council, Gassco, and Statoil for funding this thesis through the RAMONA project.

Last but not least, on a more personal note, I would like to thank my sisters,

Tahani and Mona, and my brothers, Mubark, Bashir and Abdallah, for their care, constant support and prayers. My thanks extended also to all my friends in Bergen for making life easier and more enjoyable. Above all, I would like to thank my parents, Altoma and Ali, and my wife, Sara, without whose love, support and sacrifices I could not have succeeded; my thesis is dedicated to all my family members.

University of Bergen
Department of Informatics

Models and Solution Methods for the Pooling Problem

Dissertation for the degree of Philosophiae Doctor (PhD)

Abstract

Pipeline transportation of natural gas is largely affected by restrictions regarding gas quality imposed by the market and the actual quality of the gas produced at sources. From the sources, gas flow streams of unequal compositions are mixed in intermediate tanks (pools) and blended again in terminal points. At the pools and the terminals, the quality of the mixture is given as volume-weighted average of the qualities of each mixed gas flow stream. The optimization problem of allocating flow in pipeline transportation networks at minimum cost is referred to as the pooling problem. Such problem is frequently encountered not only in gas transportation planning, but also in the process industries such as petrochemicals.

The pooling problem is a well-studied global optimization problem, which is formulated as a nonconvex (bilinear) problem, and consequently the problem can possibly have many local optima. Despite the strong $\mathcal{NP}$-hardness of the problem, which is proved formally in this thesis, much progress in solving small to moderate size instances to global optimality has recently been made by use of strong formulations. However, the literature offers few approaches to approximation algorithms and other inexact methods dedicated for large-scale instances. The main contribution of this thesis is the development of strong formulations and efficient solution methods for the pooling problem. In this thesis, we develop a new formulation that proves to be stronger than other formulations based on proportion variables for the standard pooling problem. For the generalized case, we proposes a multi-commodity flow formulation, and prove its strength over formulations from the literature.

Regarding the solution methods, the thesis proposes three solution approaches to tackle the problem. In the first methodology, we discuss solving a simplified

version of the standard pooling problem using a solution strategy that based on a sequence of semidefinite programming relaxations. The second approach is based on discretization method in which the pooling problem is approximated by a mixed-integer programming problem. Finally, we give a greedy construction method especially designed to find good feasible solutions for large-scale instances.

March 2012 – Bergen, Norway,
Mohammed Alfaki

Contents

List of publications

This thesis presents the results of my research performed at Department of Informatics, University of Bergen, for the degree of Philosophiae Doctor. The contributions of this thesis are based on the following publications:

A. Alfaki, M. and Haugland, D. (2012). **Strong formulations for the pooling problem**. In: *Journal of Global Optimization*, doi: 10.1007/s10898-012-9875-6.

B. Alfaki, M. and Haugland, D. (2012). **A multi-commodity flow formulation for the generalized pooling problem**. In: *Journal of Global Optimization*, doi: 10.1007/s10898-012-9890-7.

C. Frimannslund, L., El Ghami, M., Alfaki, M. and Haugland, D. (2010). **Solving the pooling problem with LMI relaxations**. In: S. Cafieri, B. G.-Tóth, E. Hendrix, L. Liberti and F. Messine (Eds.), *Proceedings of the Toulouse Global Optimization Workshop* (pp. 51–54).

D. Alfaki, M. and Haugland, D. (2011). **Comparison of discrete and continuous models for the pooling problem**. In: A. Caprara and S. Kontogiannis (Eds.), *11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems* (Vol. 20, pp. 112–121). OpenAccess Series in Informatics (OASIcs). Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, doi: 10.4230/OASIcs.ATMOS.2011.112.

E. Alfaki, M. and Haugland, D. (2011). **Computing feasible solutions to the pooling problem**. Submitted to: *Annals of Operations Research*.

Part I

Overview

Chapter 1

Introduction

Mathematical programming, or optimization, models and techniques have been extensively used to optimize the entire supply chain in the petroleum industry since the mid of the last century. The downstream part of the petroleum industry, which is concerned with turning crude petroleum into finished usable products that are delivered to consumers, is more concerned with mathematical programming than other parts of the industry. In the downstream part, the decision makers examine every possibility to make good decisions regarding a complex system composed of several operations. The operations include: crude purchasing, processing the crude into a variety of products, and transporting the products to the consumers. The primary goal of this business is to minimize the operational costs and maximize the profit, while satisfying customers needs. High economic value and operability benefits, associated with integrating mathematical programming software in the petroleum industry, are the driving forces for theoretical and commercial development of the field.

1.1 Background

Natural gas, which is a subcategory of petroleum, is one of the most widely used energy sources in the world. Its usage has been increasing in recent years. This is due to the facts that it is an efficient fossil fuel, with low cost and relatively low pollutant emissions to the environment. Geologists and chemists agree that petroleum was formed when the remains of organisms that accumulated in the past are compressed under the earth at very high pressure for millions of years.

Since petroleum derivatives are depleted faster than its formation, natural gas is generally considered a nonrenewable source of energy.

Natural gas is extracted from deep rock reservoirs in the Earth's crust, where it exists under high pressure, either alone or associated with heavier hydrocarbons and water. It is produced from the reservoir in the same manner as the crude oil. In general, the gas associated with heavier hydrocarbons and water is found in rock reservoirs at depths ranging between 1000 and 6000 meters, while deeper rock reservoirs produce mainly dry gas. Natural gas is mainly produced from three types of reservoirs:

1. Crude oil reservoir wells, where the gas is produced as a by-product and it is referred to as *associated gas*.
2. Dry gas wells, which typically do not contain any hydrocarbon liquids. The produced gas is called *non-associated gas*.
3. Condensate wells, in which the extracted gas is also non-associated, but contains hydrocarbon liquids. This type of gas is referred to as *unconventional gas* or wet gas.

As a consequence of natural gas market developments and the advances in production technology, the natural gas industry has begun to explore for more challenging condensate reservoirs, which have a high percentage of impurities. Examples of such condensate reservoirs are tight gas which exists in low permeability[1] rock formations, shale, coalbed methane, natural gas hydrates and deep gas. For more detailed treatments of natural gas geological formation and characteristic, the reader is referred to the survey by Mokhatab et al. (2006).

Due to the distinct characteristics of each well, all natural gas produced is not of the same quality. Even gas produced from a particular well may overtime vary in component percentages. Raw natural gas is mainly composed of methane (CH_4) with varying amounts of heavier gaseous hydrocarbons (e.g. ethane (C_2H_6), propane (C_3H_8), butane (C_4H_{10})), acid gases (e.g. carbon dioxide (CO_2), hydrogen sulfide (H_2S)), other gases (e.g. nitrogen (N_2), helium (He)), liquid hydrocarbons, water vapor, mercury, and radioactive gas. Generally, natural gas is classified into two main categories: If the natural gas contains small

[1] Permeability is the measure of the ability of a material to transmit fluids.

amounts of H_2S and CO_2 it is commonly referred to as *sweet* gas, and otherwise it is called *sour* gas. Table 1.1 shows typical chemical components in mole percentages in natural gas extracted from three different sources (Wardzinski et al., 2004). In addition to the chemical components, the gas is also described by its physical properties such as the heating value, which is the amount of energy in mega joule (MJ) per cubic meter, and the Wobbe index[2], which is used to compare the combustion energy output.

Table 1.1: Example of natural gas composition in mole percent.

Composition	Associated	Non-associated	Unconventional
Carbon-Dioxide	0.63	–	–
Nitrogen	3.73	1.25	0.53
Hydrogen-Sulfide	0.57	–	–
Methane	64.48	91.01	94.87
Ethane	11.98	4.88	2.89
Propane	8.75	1.69	0.92
Iso-Butane	0.93	0.14	0.31
n-Butane	2.91	0.52	0.22
iso-Pentane	0.54	0.09	0.09
n-Pentane	0.80	0.18	0.06
Hexanes	0.37	0.13	0.05
Heptanes-plus	0.31	0.11	0.06

Since natural gas is available with relatively affordable prices and low pollutant emissions, it is used as a source of energy as well as raw material in manufacturing. It has been used for heating space and water, air conditioning and cooking, especially in seasonal months. At the industrial level, it is used for example in power generation, hydrogen production, vehicles, and fertilizers. In addition, natural gas is an important raw material in manufacturing of fabrics, glass, and other products. Figure 1.1 shows the natural gas global demand by sector.

In order for the consumers to use the natural gas safely and efficiently in their equipment, it must be within specified quality (we simply refer to the relative content of a component or a physical property as quality) range. Otherwise, serious problems may occur, such as the flame lifting when the Wobbe index is not in its correct range. Table 1.2 shows examples of natural quality ranges

[2]The Wobble index is measured in Btu (British thermal unit of energy), 1 Btu≈ 1055 joules.

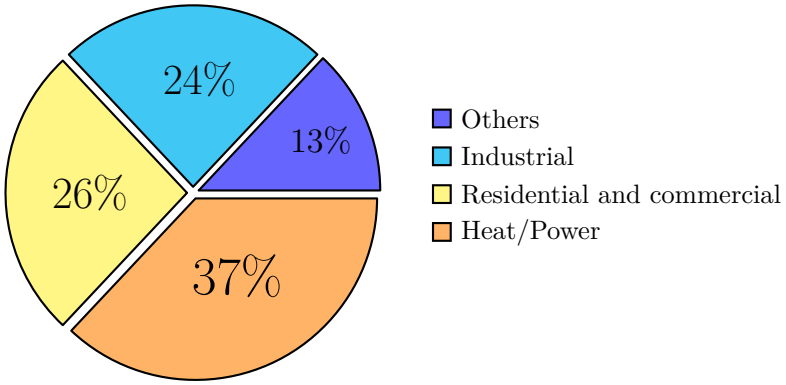


Figure 1.1: Global natural gas demand. The figure is based on data from (Simmons et al., 2006).

in North America and Europe[3]. The quality standards can vary greatly from country to country, and even from consumer to consumer in the same country.

Table 1.2: Typical examples of quality requirements demanded by the markets in North America and Europe (Hubbard, 2009). The table is divided into two groups of rows: composition and physical property requirements.

Quality	North America	Europe
CO_2 concentration	1–3 mol%	2–3 mol%
N_2	2–3 mol%*	2–3 mol%*
Total inerts	3–5 mol%*	NA
H_2S	0.25–1.0 grain/100 scf	5–7 mg/Nm^3
Total S	0.5–20 grain/100 scf	120–150 mg/Nm^3
Mercaptans	0.25–1.0 grain/100 scf*	6–15 mg/Nm^3
Oxygen	10–2000 ppm (mol)	1000–5000 ppm (mol)
Water dew-point	4–7 lbm H_2O/MMscf of gas	-10 to -12°C at 7000 kPa
Hydrocarbon dew-point	14–40°F at specified P	0 to -5°C at P < 7000 kPa
Heating value	950–1200 Btu/scf	40–46 MJ/Nm^3
Wobbe index	NA	51–56 MJ/Nm^3

NA = not applicable
* = often not specified

[3] 1 grain = 64.79891 milligrams (mg), scf ≡ standard cubic feet, Nm^3 ≡ normal cubic meter, ppm ≡ parts-per-million, the pound-mass (lbm) is a unit of mass, 1 MMscf = 10^6 scf, kPa ≡ kilo-pascals is a unit for pressure (P).

As soon as the gas is extracted from the wellheads, it goes through a number of processing operations to remove undesired impurities such as water, liquid hydrocarbons and sulfur. Removing impurities through processing operations is an important step in the natural gas journey to the end consumer. Processing operations can be divided into two major processes: *separation* and *blending*. In the separation process, natural gas is converted into intermediate products known as *pipeline-quality* dry gas, where the compositions of the product are given in fixed proportions of the original ones. In this process, natural gas can go through up to four processing facilities depending upon the level of impurities present in the gas. These processing facilities apply chemical and physical technologies. The main purpose of the separation process is to purify the natural gas to facilitate its transportation through the pipelines or the ship vessels. For more information about the separation process the reader is directed to (Guo and Ghalambor, 2005).

Blending is the physical mixture of different flow gas streams, which takes place in the so-called *pools* (tanks, vessels or through injection), where the quality of this mixture is given as volume-weighted average of the qualities of each mixed gas flow stream. In the blending operation, as opposed to the separation process, the quality of the final product depends on *both* the volume and the quality of the entering natural gas stream. In comparison to the separation process, blending is cheaper, and therefore it can be used to reduce the level of impurities prior to the separation process. If the natural gas is sweet, it can be transported directly to the end consumers, but it may still require further blending to match the consumer quality requirements.

Transporting the natural gas from the production sources to the consumers is a complex process on a transportation network of large number of pipelines and processing units. In this process, a blend of gases from different sources is formed in order to meet the end consumers' requirements, while taking into account the network configuration and its capacity. Optimization models are used to efficiently and effectively to allocate natural gas flow in this pipeline transportation network. Optimal flow allocations require that the constraints imposed by the system are modeled at an appropriate level of detail, and that corresponding solution procedures are available.

1.2 Optimization – in brief

This section briefly introduces some mathematical optimization concepts that are useful in this thesis. For a comprehensive treatment of this subject, the interested reader is referred to Boyd and Vandenberghe (2004) for convex optimization, Nocedal and Wright (2000) and Floudas (2000) or Hendrix and G.-Tóth (2010) for nonconvex optimization, and Wolsey (1998) for integer optimization.

The origin of optimization can be traced back to the work of Euler and Lagrange in the calculus of variations. In the 1940s, the invention of *linear programming* by Kantorovich (1940) and Dantzig (1949), and the subsequent theoretical and practical developments have further shaped the field. An *optimization problem* is to find the best solution for minimizing (or maximizing) an *objective function* subject to inequality and/or equality constraints. Suppose that $x \in \mathbb{R}^n$ is a vector of n *variables*, and $f_i : \mathbb{R}^n \mapsto \mathbb{R}$, where $i = 0, 1, 2, \ldots, m$, are the objective ($i = 0$) and the constraint functions ($i \geq 1$), respectively. The optimization problem can be written as:

$$\left.\begin{array}{lll} \min\limits_{x \in \mathbb{R}^n} & f_0(x) & \\ \text{s.t.} & f_i(x) \leq 0, & i \in \mathcal{I}, \\ & f_i(x) = 0, & i \in \mathcal{E}, \end{array}\right\} \tag{1.1}$$

where the sets $\mathcal{I}$ and $\mathcal{E}$ consist of indices for inequality and equality constraints, respectively. If the sets $\mathcal{I}$ and $\mathcal{E}$ are empty, the optimization problem (1.1) is called an *unconstrained optimization* problem, otherwise the problem is referred to as a *constrained optimization* problem. Since the variables x are assumed to take real values in (1.1), the problem is called a *continuous optimization* problem. However, in many modeling situations, the variables make sense only if they take discrete values, e.g. $x \in S \subseteq \mathbb{Z}^n$ instead of $x \in \mathbb{R}^n$ in (1.1), in which we refer to the problem as an *integer optimization* problem. When some, but not all, of the variables are restricted to be integers the problem is known as a *mixed integer programming* problem.

The *feasible region* Ω of the problem (1.1) is defined as the set of all vectors satisfying the constraints. That is, $\Omega = \{x \in \mathbb{R}^n : f_i(x) \leq 0,\ i \in \mathcal{I}; f_i(x) = 0,\ i \in \mathcal{E}\}$.

We refer to a vector x^* as a *local optimum*, if it has the smaller objective function value among all vectors in a neighborhood of x^*. A local optimum is called a *global optimum*, if it has the smallest objective function value among all local optima. Global optimal solutions are important in many practical applications. Nonetheless, the task of finding them is challenging in many problems.

Optimization problems can be classified depending on the form of the objective and constraint functions. An important concept is the notion of *convexity*, which is a property that makes the optimization problem efficiently solvable in both theory and practice.

Before going further, let us informally explain what we mean by efficiently solvable and hard problems. In the computational complexity theory, we say that a problem is efficiently (polynomially) solvable, if there exists an algorithm which computes its exact solution in a number of arithmetic operations that is bounded above by a polynomial in the instance size, for any instance of the problem. The algorithm is said to have polynomial running time, or to be a *polynomial time algorithm* in short. Here, the instance size of a problem is the number of bits needed to represent the instance on the computer.

Given the optimization problem (1.1), an associated *decision problem* is, for a given number $z \in \mathbb{R}$, to answer 'yes' or 'no' to the question: Is there an $x \in \Omega$ such that $f_0(x) \leq z$?[4] The class $\mathcal{NP}$ (non-deterministic polynomial time) contains decision problems where a given 'yes'-answer can be *verified* in polynomial time. The class of all polynomially solvable decision problems in $\mathcal{NP}$ is referred to as $\mathcal{P}$. The decision problem $\pi_1 \in \mathcal{NP}$ is *polynomially reducible* to $\pi_2 \in \mathcal{NP}$, if we can convert any instance of π_1 to an instance of π_2 in polynomial time. A decision problem $\pi_1 \in \mathcal{NP}$ is $\mathcal{NP}$-complete, if all $\pi \in \mathcal{NP}$ are polynomially reducible to π_1. In other words, if a polynomial time algorithm for any $\mathcal{NP}$-complete problem exists, then all problems in $\mathcal{NP}$ can be solved in polynomial time. No $\mathcal{NP}$-complete problem is currently known to have a polynomial solution algorithm, and a big question in computer science that remains unsolved is the $\mathcal{P}$ versus $\mathcal{NP}$ question, i.e. is $\mathcal{P} = \mathcal{NP}$ or $\mathcal{P} \neq \mathcal{NP}$? Resolution to the question, either way, will have important theoretical and computational consequences. An

[4] By solving the decision problem a number of times and using the bisection on the objective function value, we can find the optimal solution to the original optimization problem.

optimization problem for which its decision problem is $\mathcal{NP}$-complete, is referred to as an $\mathcal{NP}$-hard problem.

Definition 1.1. A set $S \subseteq \mathbb{R}^n$ is convex if the entire line segment between any two points of S lies in S. That is, for all $x, y \in S$ we have $\alpha x + (1-\alpha)y \in S$ for all $\alpha \in [0,1]$.

The notion of convexity also applies to functions. We say that a function f is convex if and only if the set of points above the graph of f is convex. That is, for any $x, y \in S$ (S is convex) we have,

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y), \ \forall \ \alpha \in [0,1].$$

A function f is said to be concave if $-f$ is convex. A class of optimization problems in which the objective function and the feasible region are convex is referred to as *convex optimization* problems. One important property of this class is that any local optimum is also a global optimum, which means that it is sufficient to apply any local optimization algorithm in order to compute the global optimum. An advantage of recognizing a problem as a convex optimization problem, is that the problem can be solved in polynomial time, for example using *interior-point* methods. This means that the convex optimization problems are in $\mathcal{P}$. One of the most widely used subclasses of convex optimization problems is known as the *linear programming* problem, in which all the constraints and the objective function are linear. That is, in problem (1.1), the objective and constraint functions satisfy

$$f_i(\alpha x + \beta y) = \alpha f_i(x) + \beta f_i(y),$$

for all $x, y \in \mathbb{R}^n$ and for all $\alpha, \beta \in \mathbb{R}$. Several effective methods for solving the linear programming problem are used in many practical applications. Among these is the simplex method developed by Dantzig (1949), and the interior-point method introduced by Karmarkar (1984), which later has been extended by Nesterov and Nemirovskii (1994) for solving general convex optimization problems.

The class of problems where the objective function or the feasible region is not necessarily convex is referred to as *nonconvex optimization* problems. As

opposed to convex problems, nonconvex problems may have several local optima with unequal objective function values, and there is no polynomial time algorithm (unless $\mathcal{P} = \mathcal{NP}$) to find a global optimum in general. Computation and characterization of global optima are the subjects of *global optimization*.

Global optimization has traditionally attracted far less attention than local optimization and solution methods for convex problems. Over the last few decades, however, research in the field has emerged, and several textbooks devoted to the subject have been published. Noteworthy among these are (Horst et al., 2000), (Floudas, 2000) and (Hendrix and G.-Tóth, 2010).

Global optimization approaches are typically based upon *relaxation* problems of the original nonconvex problem to provide lower bounds on the optimal objective function value. The relaxation problem is a modification of the objective function and/or the feasible region giving a new problem that is easier to solve, and it is formally defined as follows:

Definition 1.2. A problem $\min_{x \in \tilde{\Omega}} \tilde{f}_0(x)$ is a relaxation of the problem $\min_{x \in \Omega} f_0(x)$ if: (i) $\Omega \subseteq \tilde{\Omega}$, and (ii) $\tilde{f}_0(x) \leq f_0(x)$, for all $x \in \Omega$.

Different types of relaxation techniques are used to compute lower bounds for the nonconvex problems, for example convex relaxations and Lagrangian relaxations. One way to construct convex relaxations of problems with some nonconvex constraint function f can be based on the *convex* and *concave envelopes* of the function f. Let S be a convex set, and let $\mathcal{L}(f, S)$ be the set of convex functions $g : \mathbb{R}^n \mapsto \mathbb{R}$ that everywhere in S lie below f, i.e. for all $x \in S$ we have $g(x) \leq f(x)$. Then the convex envelope of f is formally defined as follows:

Definition 1.3. The convex envelope of a function $f : \mathbb{R}^n \mapsto \mathbb{R}$ on a convex set S is defined as a function $\mathrm{vex}_S f : \mathbb{R}^n \mapsto \mathbb{R}$ such that for all $x \in S$

$$\mathrm{vex}_S f(x) = \sup_g \{g(x) : g \in \mathcal{L}(f, S)\}.$$

Hence, $\mathrm{vex}_S f(x)$ is the pointwise supremum of $\mathcal{L}(f, S)$. The concave envelope of f is defined as $\mathrm{cav}_S f(x) = -\mathrm{vex}(-f)_S(x)$, which thus becomes the smallest concave function which throughout S lies above f.

1.3 Problem statement

In this work, we consider an abstraction of natural gas transportation networks consisting of supply nodes, intermediate nodes and terminal nodes, as well as links representing pipelines used to transport the gas between two nodes. The gas enters the transportation network at the supply nodes and flows through the intermediate nodes, and finally, it leaves the network through the terminal nodes. In this network, blending operations occur in the intermediate and terminal nodes.

1.3.1 The standard pooling problem

Because of the blending operations and the quality constraints at the terminals, the problem of allocating gas flow to the network is equivalent to a problem frequently occurring in planning oil refinery operations. Haverly (1978) defined the *pooling problem* in terms of the instance depicted in Figure 1.2. A problem instance of the pooling problem is characterized by a network with, at the reception side, source streams with different qualities that can enter the network. Flow from the sources is fed into a limited number of available storage tanks (pools), where the entering flow is mixed to form intermediate blends with new qualities. The pool contents are subsequently used to form final blends at the terminals, where specific quality requirements to the blend are imposed by the market. Due to the blending operations, the optimization model for this problem involves nonconvex constraints, and consequently the pooling problem can possibly have many local optima.

In the Haverly example, we have three types of oils (the source nodes) denoted s_1, s_2 and s_3 with different concentrations of sulfur contents (for simplicity, assume that sulfur is the only quality parameter) given as 3%, 1% and 2%, respectively. The oils s_1 and s_2 are blended in an intermediate node (pool) denoted p_1, whereas the oil in s_3 is transported directly to two consumers (terminal nodes) denoted t_1 and t_2. The output of the pool p_1 is transported to the same terminals. Consumers t_1 and t_2 will only buy the oil if it contains no more than

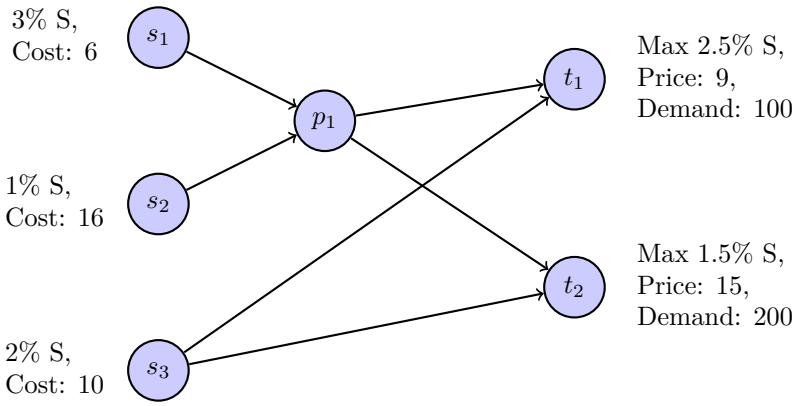


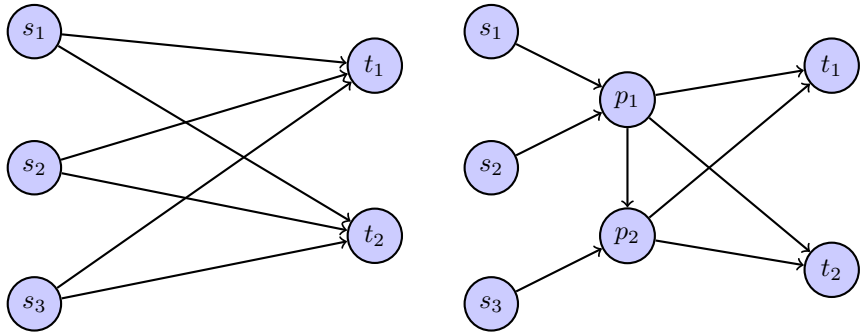
Figure 1.2: Haverly's pooling problem instance.

2.5% and 1.5% of sulfur, respectively. The oil price at the sources, the consumer demand, and sale prices are given in Figure 1.2.

The objective function is to minimize the total cost, while satisfying the consumers quality standard and demand. Dealing with the general instantiations of the pooling problem is the subject of this thesis. Its translation to transportation networks for natural gas is quite direct: Wells and processing units correspond to the sources, junction points are represented by pools, and reception units are modeled as terminals. Although our motivation is the natural gas industry, we will study the pooling problem and its extensions independently of their particular applications.

1.3.2 The blending problem

A special case of the problem under study occurs when the intermediate nodes are not needed, in other words, the flow streams are directly blended at the terminal nodes, leading to a problem known as the *blending problem*. This problem can be modeled as a linear program and can hence be solved fast. An example of this problem is shown in Figure 1.3(a), which is a modification of Haverly's instance illustrated in Figure 1.2, where the pool p_1 is removed.



(a) Blending (b) Generalized

Figure 1.3: Examples of the blending problem, and the generalized pooling problem.

1.3.3 The generalized pooling problem

In the literature, when referring to the pooling problem, it is frequently assumed that the only connections allowed in the network are the connections from sources to pools, from sources to terminals, and from pools to terminals. We refer to this as the *standard pooling problem* in this thesis. An example is given in Figure 1.2. Audet et al. (2004) introduced the *generalized* pooling problem, where connections between pools are allowed as well. Figure 1.3(b) shows an example constructed by modification of Haverly's instance depicted above.

Driven by increased consumption of energy in the world, coupled with the introduction of unconventional gas in the transportation network, pipeline transportation of natural gas has become a complex system. These conditions have posed several challenges that require more accurate optimization models and more efficient solution methods for the pooling problem. Although a number of optimization models and solution approaches have been applied to the pooling problem, solving large-scale instances in reasonable time with acceptable accuracy is still challenging. This thesis contributes to both modeling and algorithmic methods in order to approach this problem.

1.4 Structure of the thesis

This thesis is divided into two parts. Part I provides motivations, overview of the field, related works, and a summary of the scientific contributions. In Part II, the five publications documenting the results of the thesis are provided in separate attachments in their published form, with the exception of their style which is changed to suite this thesis format.

Part I is further composed of five chapters, including this introductory chapter, and the remaining chapters are organized as follows: In Chapter 2 and 3, we give, along with the thesis contributions, a literature review of model formulations and solution methods for pooling problems, respectively. In Chapter 4, summaries of the included papers are given. Finally, Chapter 5 gives some concluding remarks and proposes possible future research directions.

Chapter 2

Model formulations for pooling problems

The pooling problem is an important optimization problem that has been extensively studied mainly because of its industrial applications. In addition to the applications in the transportation of natural gas and in oil refining discussed in the previous chapter, it is frequently encountered in waste-water treatment and general petrochemicals industries. The research on pooling problems focuses on two directions: The first is developing mathematical formulations that exhibit favorable properties, and the second is designing efficient solution methods (see Chapter 3).

This chapter reviews the different formulations to our problem and its extensions. They are categorized into models for the standard and the generalized pooling problems, which in their turn can also be classified as quality and proportion based models. All of these have bilinear constraints. In a global optimization context, pooling problems are often approached by branch-and-bound algorithms (studied in more details in Section 3.2.1), which rely mainly on the construction of linear relaxations. Building stronger/tighter formulations to provide strong lower bounds on the optimal solution are crucial to the convergence of the algorithms.

2.1 Introduction

The *minimum-cost flow problem* with node capacities is defined on a directed graph $G = (N, A)$ where N is the set of nodes and A is the set of arcs, where each node $i \in N$ has a known *capacity* b_i, and each arc $(i, j) \in A$ has a unit *cost* c_{ij}. For any node $i \in N$, let $N_i^+ = \{j \in N : (i, j) \in A\}$ and $N_i^- = \{j \in N : (j, i) \in A\}$ denote the set of out- and in-neighbors of i, respectively. We assume that G has

non-empty sets $S, T \subseteq N$ of *sources* and *terminals*, respectively, where $N_s^- = \emptyset$, $\forall\ s \in S$ and $N_t^+ = \emptyset$, $\forall\ t \in T$. The optimization problem is to determine the minimum-cost plan for sending flow through the network to satisfy supply and demand requirements. The arc flows must be nonnegative and respect the node capacities, and they must satisfy conservation of flow at the intermediate nodes. The minimum-cost flow problem is a transportation model that allocates *single* commodity flow in the network, in other words, the flow entering the network at different sources has the same quality. As already known, there are efficient algorithms to solve this problem which can be modeled as a linear programming problem.

In natural gas transportation networks, the gas flow is coming from several sources that have different qualities, which means that we have *multi-commodity* flow in the network. Consequently, the minimum-cost flow model in this situation needs to be extended to handle this kind of flow. The pooling problem can be viewed as an extension of the minimum-cost flow problem where the quality of the flow depends on the sources from which it originates. At each source, the quality is known, whereas in all other nodes, the quality of the flow blends linearly (see Definition 2.1). In addition to the minimum-cost flow problem parameters, let K be the set of all *quality attributes*. With each $i \in S \cup T$, we define a real constant q_i^k for each $k \in K$. If $s \in S$, q_s^k is referred to as the *quality parameter* of attribute k at that source, and if $t \in T$, q_t^k is referred to as the *quality bound* of attribute k at terminal t. We refer to all nodes in $I = N \setminus (S \cup T)$ as *pools*.

Definition 2.1 (*Linear blending*)**.** The quality at node $i \in I \cup T$ is defined as a weighted average of the qualities at entering arcs, where the corresponding arc flows constitute the weights. The quality at any arc $(i, j) \in A$ is defined as the quality at node i.

Definition 2.2. The *pooling problem* is to assign flow values to all arcs $(i, j) \in A$ such that, in addition to the constraints of the minimum-cost flow problem, the quality bounds at the terminals are respected while the total flow cost is minimized.

In the literature, when referring to the pooling problem, it is frequently assumed that all maximal paths in G have exactly one source and one terminal,

and at most one pool. We refer to this as the *standard pooling problem*, which means that G is a tripartite graph, i.e. $A \subseteq (S \times I) \cup (I \times T) \cup (S \times T)$. The optimization problem arising when no longer assuming the tripartite network structure is referred to (Audet et al., 2004; Misener and Floudas, 2009) as the *generalized pooling problem*.

Different variants of optimization models (or formulations) for the pooling problem and its extensions exist in the literature. Generally, one can divide these formulations into two main categories. The first one consists of flow and quality variables, whereas the other uses flow proportions instead of quality variables. In the next sections, we discuss these formulations and their extensions in more details.

2.2 Formulations for the standard pooling problem

2.2.1 The quality formulation

The most straightforward formulation is achieved by spelling out the definition given in Section 2.1, which in the literature is commonly referred to as the P-formulation. Define f_{ij} as the flow along the arc $(i,j) \in A$, and w_i^k $(k \in K)$ as the quality of the flow leaving node $i \in S \cup I$ (if $i \in S$ let $w_i^k = q_i^k$). Then the P-formulation can be written as:

$$[\text{P}] \qquad \min_{f,w} \sum_{(i,j)\in A} c_{ij} f_{ij}, \tag{2.1}$$

$$\text{s.t.} \qquad \sum_{j\in N_i^+} f_{ij} \le b_i, \qquad i \in N \setminus T, \tag{2.2}$$

$$\sum_{j\in N_t^-} f_{jt} \le b_t, \qquad t \in T, \tag{2.3}$$

$$\sum_{j\in N_i^-} f_{ji} - \sum_{j\in N_i^+} f_{ij} = 0, \qquad i \in I, \tag{2.4}$$

$$\sum_{j\in N_i^-} w_j^k f_{ji} - \sum_{j\in N_i^+} w_i^k f_{ij} = 0, \qquad i \in I,\ k \in K, \tag{2.5}$$

$$\sum_{j \in N_t^-} w_j^k f_{jt} - q_t^k \sum_{j \in N_t^-} f_{jt} \leq 0, \qquad t \in T,\ k \in K, \tag{2.6}$$

$$f_{ij} \geq 0, \qquad (i,j) \in A. \tag{2.7}$$

Constraints (2.2)–(2.3) and (2.4) express the flow capacity bound at all nodes and the flow conservation around pool nodes, respectively. Constraint (2.5) is the result of direct application of Definition 2.1 to all pool nodes, whereas constraint (2.6) follows by the application of the same definition to terminal $t \in T$, and the quality bound constraint, $w_t^k \leq q_t^k$ for all $k \in K$, from the problem definition. The number of bilinear terms in the P-formulation is proportional to the number of quality attributes. The P-formulation is originally derived for the standard pooling problem. Nevertheless, it can easily be generalized to handle more general pooling networks than the formulation (2.1)–(2.7) does.

Haverly (1978) is the first to use the P-formulation to model the pooling problem, and from that time, many researchers have used it. Among them are Lasdon et al. (1979), Floudas and Aggarwal (1990), Foulds et al. (1992) and Fieldhouse (1993). A practical application of the P-formulation has been shown by Baker and Lasdon (1985) and Amos et al. (1997), who use this formulation at Exxon refineries and New Zealand Refining company, respectively.

2.2.2 Proportion formulations

An alternative formulation relies on variables that represent proportions (fractions) of flow instead of explicit quality variables. However, these types of formulation are applicable only for the standard pooling problem. We can use two types of proportion variables, source or terminal proportions, to replace the quality variables. The idea of using proportion variables was first suggested by Ben-Tal et al. (1994) who derived a formulation that relies on source proportion variables, and they referred to it as the Q-formulation. Recently, by building on the same idea, we gave two new formulations based on terminal proportions (see Paper A).

2.2.2.1 Formulations with source proportions

Define the proportion variables y_i^s ($s \in S$, $i \in I$) as the fraction of the flow through pool i that originates from source s. That is, if the flow through i is non-zero, we have $y_i^s = f_{si} / \sum_{t \in N_i^+} f_{it}$. We keep variable f_{ij} as the flow along the arc $(i,j) \in A$ as in Section 2.2.1. We observe that, due to the introduction of the new variables, the flow along arc (s,i), where $s \in S$ and $i \in I$, can be represented by $\sum_{t \in N_i^+} y_i^s f_{it}$. Using this observation in constraint (2.5), the quality variables can be expressed as $w_i^k = \sum_{s \in N_i^-} q_s^k y_i^s$ for all $i \in I$, $k \in K$. Combining these observations and the proportion variables with the flow variables, we arrive at the Q-formulation written as:

$$
\begin{aligned}
[\mathrm{Q}] \quad \min_{f,y} \quad & \sum_{i \in I} \sum_{s \in N_i^-} c_{si} y_i^s \sum_{t \in N_i^+} f_{it} + \sum_{t \in T} \sum_{j \in N_t^-} c_{jt} f_{jt} \\
\text{s.t.} \quad & \sum_{i \in I \cap N_s^+} y_i^s \sum_{t \in N_i^+} f_{it} + \sum_{t \in T \cap N_s^+} f_{st} \le b_s, && s \in S, \\
& \sum_{t \in N_i^+} f_{it} \le b_i, && i \in I, \qquad (2.8) \\
& \sum_{j \in N_t^-} f_{jt} \le b_t, && t \in T, \\
& \sum_{s \in S \cap N_t^-} q_s^k f_{st} + \sum_{i \in I \cap N_t^-} \sum_{s \in N_i^-} q_s^k y_i^s f_{it} \le q_t^k \sum_{j \in N_t^-} f_{jt}, && t \in T,\ k \in K, \\
& \sum_{s \in N_i^-} y_i^s = 1, && i \in I, \qquad (2.9) \\
& f_{jt} \ge 0, && t \in T,\ j \in N_t^-, \\
& 0 \le y_i^s \le 1, && i \in I,\ s \in N_i^-.
\end{aligned}
$$

The number of nonlinear variables in the Q-formulation is independent of the number of quality attributes, making this formulation more practical as the number of quality attributes increases. Tawarmalani and Sahinidis (2002) extended the Q-formulation by applying the reformulation-linearization technique (see Section 3.2.1) to constraints (2.8) and (2.9). That is, multiplying (2.9) by

f_{it} yields (2.10), similarly, (2.11) is obtained by multiplying (2.8) by y_i^s. The new formulation is referred to as the PQ-formulation. The new constraints,

$$f_{it} - \sum_{s \in N_i^-} y_i^s f_{it} = 0, \quad i \in I,\ t \in N_i^+, \tag{2.10}$$

$$\sum_{t \in N_i^+} y_i^s f_{it} - b_i y_i^s \leq 0, \quad i \in I,\ s \in N_i^-, \tag{2.11}$$

were already derived by Quesada and Grossmann (1995). Tawarmalani and Sahinidis (2002) proved that the linear programming relaxation (constructed by the McCormick envelopes, see Section 3.2.1) of the PQ-formulation dominates the linear programming relaxation of both the P- and the Q-formulation.

2.2.2.2 Formulations with terminal proportions

The PQ-formulation has proportion variables corresponding to sources, and flow variables on arcs entering terminals. Symmetric to the PQ-formulation, we suggest in Paper A (Alfaki and Haugland, 2012b) a formulation with proportion variables corresponding to terminals, and flow variables on arcs leaving sources.

Define for all pools $i \in I$, y_i^t as the proportion of the flow at i destined for terminal $t \in T$. That is, we let $y_i^t = f_{it} / \sum_{s \in S} f_{si}$ when the latter sum is positive. Hence, the new formulation, referred to as the TP-formulation, is given as follows:

$$\begin{aligned}
[\text{TP}] \quad \min_{f,y} \quad & \sum_{s \in S} \sum_{j \in N_s^+} c_{sj} f_{sj} + \sum_{i \in I} \sum_{t \in N_i^+} c_{it} y_i^t \sum_{s \in N_i^-} f_{si} \\
\text{s.t.} \quad & \sum_{j \in N_s^+} f_{sj} \leq b_s, && s \in S, \\
& \sum_{s \in N_i^-} f_{si} \leq b_i, && i \in I, \\
& \sum_{s \in S \cap N_t^-} f_{st} + \sum_{i \in I \cap N_t^-} \sum_{s \in N_i^-} f_{si} y_i^t \leq b_t, && t \in T, \\
& \sum_{s \in S \cap N_t^-} q_s^k f_{st} + \sum_{i \in I \cap N_t^-} \sum_{s \in N_i^-} q_s^k f_{si} y_i^t \leq q_t^k \sum_{j \in N_t^-} f_{jt}, && t \in T,\ k \in K, \\
& \sum_{t \in N_i^+} y_i^t = 1, && i \in I,
\end{aligned}$$

$$\sum_{t \in N_i^+} f_{si} y_i^t - f_{si} = 0, \qquad s \in N_i^-,\ i \in I, \quad (2.12)$$

$$\sum_{s \in N_i^-} f_{si} y_i^t - b_i y_i^s \leq 0, \qquad i \in I,\ s \in N_i^-, \quad (2.13)$$

$$f_{sj} \geq 0, \qquad s \in S,\ j \in N_s^+,$$

$$0 \leq y_i^t \leq 1, \qquad i \in I,\ t \in N_i^+.$$

The interpretation of the constraints is analogous to the PQ-formulation. Constraints (2.12) and (2.13) are redundant. Following the pattern of the PQ-formulation, they are included to strengthen the relaxation. A comparison to the PQ-formulation showed that the formulations do not in general have equal strength, but none dominates the other (see Paper A).

In Paper A (Alfaki and Haugland, 2012b), we develop a new proportion formulation by combining both source and terminal proportions in one model. It follows from the definition of y_i^s and y_i^t that $y_i^s f_{it}$ and $y_i^t f_{si}$ both can be interpreted as the flow along the unique path connecting source $s \in S$, pool $i \in I$ and terminal $t \in T$. Given this observation, the STP-formulation can be derived by combining the variables and the constraints from both models.

In the same paper, it has been shown that the linear relaxation of the STP-formulation is at least as tight as the relaxations of both the PQ-, and the TP-formulations. The experiments presented in the paper showed that the STP-formulation in some instances is tighter than both its competitors (see Paper A for complete details).

2.3 Formulations for the generalized pooling problem

2.3.1 A hybrid formulation

Audet et al. (2004) applied their branch-and-cut algorithm to both the P- and Q-formulation and found that the Q-formulation is the more favorable in their algorithm. They also observed that the Q-formulation is not applicable to networks where flow streams leaving one pool may be blended in some pools further downstream in the network. Therefore, the Q-formulation in such networks is no

longer a bilinear model. In fact, the model will contain terms that are products of two proportion variables. In order to avoid terms where proportion variables are squared, Audet et al. (2004) introduced such variables exclusively for pools that only have sources as in-neighbors, and quality variables for the remaining pools. Denote the former subset of pools I_1. For each pool which in the sense defined above is close to the sources, the hybrid model thus makes use of a proportion variable y_i^s for each neighboring source s (as defined in Section 2.2.2.1). For other pools, a quality variable w_i^k for each attribute $k \in K$ (as defined in Section 2.2.1) is used. The hybrid formulation denoted [HYB] can be written as:

$$\text{[HYB]} \quad \min_{f,y,w} \sum_{i\in N} \sum_{j\in N_i^+\setminus I_1} c_{ij} f_{ij} + \sum_{s\in S} \sum_{i\in N_s^+\cap I_1} \sum_{j\in N_i^+} c_{si} y_i^s f_{ij}, \tag{2.14}$$

$$\text{s.t.} \quad \sum_{j\in N_s^+\setminus I_1} f_{sj} + \sum_{i\in N_s^+\cap I_1} \sum_{j\in N_i^+} y_i^s f_{ij} \le b_s, \qquad s \in S, \tag{2.15}$$

$$\sum_{i\in N_t^-} f_{it} \le b_t, \qquad t \in T, \tag{2.16}$$

$$\sum_{j\in N_i^+} f_{ij} \le b_i, \qquad i \in I, \tag{2.17}$$

$$\sum_{j\in N_i^+} f_{ij} - \sum_{j\in N_i^-} f_{ji} = 0, \qquad i \in I \setminus I_1, \tag{2.18}$$

$$\sum_{j\in N_i^-\cap I_1} \sum_{s\in N_j^-} q_s^k y_j^s f_{ji} + \sum_{j\in N_i^-\cap(I\setminus I_1)} w_j^k f_{ji} - \sum_{j\in N_i^+} w_i^k f_{ij} = 0, \qquad i \in I \setminus I_1,\ k \in K, \tag{2.19}$$

$$\sum_{j\in N_t^-\cap I_1} \sum_{s\in N_j^-} q_s^k y_j^s f_{jt} + \sum_{j\in N_t^-\cap(I\setminus I_1)} w_j^k f_{jt} - q_t^k \sum_{j\in N_t^-} f_{jt} \le 0, \qquad t \in T,\ k \in K, \tag{2.20}$$

$$\sum_{s\in N_i^-} y_i^s = 1, \qquad i \in I_1, \tag{2.21}$$

$$f_{ij} \ge 0, \qquad (i,j) \in A,\ j \notin I_1, \tag{2.22}$$

$$0 \le y_i^s \le 1, \qquad (s,i) \in A,\ i \in I_1. \tag{2.23}$$

In Paper B (Alfaki and Haugland, 2012a), we have found a flaw in the original hybrid formulation given by Audet et al. (2004). The authors gave this formulation only in terms of an example of the generalized pooling problem. Therefore, (2.14)–(2.23) is not only a correction but also a generalization of their formulation.

2.3.2 A multi-commodity flow formulation

In order to extend the PQ-formulation to the generalized pooling problem, we suggest in Paper B (Alfaki and Haugland, 2012a) a multi-commodity flow formulation. We associate a flow *commodity* with each source $s \in S$, where at most b_s units of the commodity can enter the network. The commodity can leave the network at any $t \in T$. At all other nodes, the commodity neither enters nor leaves the network. Now, the variable f_{ij} defines the total flow of all commodities along arc $(i,j) \in A$. For each $i \in N$, let S_i be the set of sources from which there exists a path to i in G (let $S_s = \{s\}\ \forall\ s \in S$). Relative to the total flow leaving node $i \in S \cup I$, let the variable y_i^s (this is a generalization of the proportion variable introduced in Section 2.2.2.1) denote the proportion of commodity s. Define $y_i^s = 0$ if $s \notin S_i$ and $y_s^s = 1$ for all $s \in S$. Therefore, the quantity $y_i^s f_{ij}$ defines the flow of commodity s (meaning the commodity associated with source s, we simply refer to s as a commodity whenever convenient) along the arc (i,j). Based on this multi-commodity flow idea, we have the following formulation:

$$
\begin{aligned}
\text{[MCF]} \qquad & \min_{f,y,x} \sum_{(i,j)\in A} c_{ij} f_{ij} & & \\
\text{s.t.} \qquad & \sum_{j \in N_i^+} f_{ij} \le b_i, & & i \in N \setminus T, \\
& \sum_{j \in N_t^-} f_{jt} \le b_t, & & t \in T, \\
& \sum_{j \in N_i^-} y_j^s f_{ji} - \sum_{j \in N_i^+} y_i^s f_{ij} = 0, & & i \in I,\ s \in S_i, \qquad (2.24) \\
& \sum_{j \in N_t^-} \sum_{s \in S_j} \left(q_s^k - q_t^k\right) y_j^s f_{jt} \le 0, & & t \in T,\ k \in K, \qquad (2.25)
\end{aligned}
$$

$$\sum_{s \in S_i} y_i^s = 1, \qquad i \in I,$$

$$\sum_{s \in S_i} y_i^s f_{ij} - f_{ij} = 0, \qquad (i,j) \in A,\ i \in I, \tag{2.26}$$

$$\sum_{j \in N_i^+} y_i^s f_{ij} - y_i^s b_i \leq 0, \qquad i \in I,\ s \in S_i, \tag{2.27}$$

$$f_{ij} \geq 0, \qquad (i,j) \in A,$$

$$0 \leq y_i^s \leq 1, \qquad i \in I,\ s \in S_i.$$

Constraints (2.25) express the quality bound at the terminals, and the constraints (2.24) impose the flow proportions y_i^s on all arcs with start node i. Analogous to (2.10)–(2.11), constraints (2.26)–(2.27) are redundant, but are added for the same reason as (2.10)–(2.11) were added to the PQ-formulation. The linear relaxation of the MCF-formulation dominates the corresponding linear relaxations of the HYB-formulation and the generalized version of the P-formulation. In Paper B (Alfaki and Haugland, 2012a), we present computational experiments with this formulation and the P- and the HYB-formulation applied to 40 instances of the generalized pooling problem. Experiments demonstrate that the suggested formulation enables faster computation of the global optimum.

2.4 Formulations for extensions

Meyer and Floudas (2006) and Misener and Floudas (2010) introduced an extension of the pooling problem where the network topology is treated as decision variables: There is a fixed charge for opening the arcs and activating the treatment plants. Therefore, binary variables are needed and the model becomes a mixed integer nonlinear program (MINLP). Such a problem has applications to the design of wastewater treatment networks (Takama et al., 1980). The authors use the P-formulation, since the flow of water may undergo reduction of contamination through several stages of treatment plants. In general, the model for the *wastewater treatment problem* defers from the pooling problem in two aspects: First, there is a fixed cost for opening arcs and installing treatment plants.

Second, each treatment plant has a removal ratio, which represents the removal technologies on this plant, for each contamination (quality) parameter.

Consider the parameters defined in Section 2.1. Let the set of sources represent the effluent streams which usually come form industrial plants. The pools no longer play the role of storage tanks or mixers only, in addition they may be used to reduce the contaminant levels in the wastewater streams. For each treatment plant $i \in I$, define the constant r_i^k as the *removal ratio* of quality (contaminant) $k \in K$. The terminals represent the exit side into which the treated wastewater flows. Define c_{ii} as the unit cost of the flow going through treatment plant $i \in I$. For each arc $(i,j) \in A$ define d_{ij} as the *fixed cost* for opening this arc and d_{ii} as the fixed cost of using the treatment plant $i \in I$. Define the binary variable z_{ij} indicating whether arc $(i,j) \in A$ is used, and z_{ii} as binary variable for using plant $i \in I$. The formulation for the wastewater treatment problem given in (Meyer and Floudas, 2006) can be written as:

$$\min \quad \sum_{i \in I} \left(\sum_{j \in N_i^+} c_{ii} f_{ij} + d_{ii} z_{ii} \right) + \sum_{(i,j) \in A} (c_{ij} f_{ij} + d_{ij} z_{ij}), \tag{2.28}$$

$$\begin{aligned}
\text{s.t.} \quad & \sum_{j \in N_s^+} f_{sj} \leq b_s, && s \in S, \\
& \sum_{j \in N_i^+} f_{ij} \leq b_i z_{ii}, && i \in I, \\
& \sum_{j \in N_t^-} f_{jt} \leq b_t, && t \in T, \\
& \sum_{j \in N_i^+} f_{ij} - \sum_{j \in N_i^-} f_{ji} = 0, && i \in I, \\
& \left(1 - r_i^k\right) \sum_{j \in N_i^-} w_j^k f_{ji} - \sum_{j \in N_i^+} w_i^k f_{ij} = 0, && i \in I,\ k \in K, \qquad (2.29) \\
& \sum_{j \in N_t^-} w_j^k f_{jt} - q_t^k \sum_{j \in N_t^-} f_{jt} \leq 0, && t \in T,\ k \in K, \\
& w_i^k \leq \left(1 - r_i^k\right) \max_{s \in S} q_s^k, && i \in I,\ k \in K, \qquad (2.30) \\
& z_{ij},\ z_{ii} \in \{0,1\}, && i \in I,\ j \in N_i^+, \\
& f_{ij} \geq 0, && (i,j) \in A.
\end{aligned}$$

The objective (2.28) is minimization of the total fixed and variable cost associated with use of the pipelines and installation of the treatment plants. We observe that each quality $k \in K$ of the flow leaving treatment plant $i \in I$ will be reduced by r_i^k percent, and hence the quality balance constraint (2.29) follows. Constraint (2.30) has been added to strengthen the formulation.

Another interesting extension of the standard pooling problem was proposed by Misener and Floudas (2010). The purpose of their model is to maximize the profit of blending reformulated gasoline, subject to environmental standards that involve complex emission constraints.

In an oil and gas production planning context, sometimes the decision makers must take into account newly discovered and developed oil and gas fields, and consider these in the pipeline transportation system. However, knowledge of different quality levels and capacities of the wells is needed in advance. To handle such situations, Armagan (2009) and Li et al. (2011b) proposed the *stochastic pooling problem*, which is an extension of the pooling problem accounting for uncertain parameters present in the planning model. Examples of such uncertain parameters are the quality parameters in the raw gas, the capacity bounds of production sources, and the demands at the consumers side. Li et al. (2011a) presented a stochastic MINLP formulation for this generalization of the pooling problem, where the uncertainty in the parameters is given by a limited number of scenarios.

Chapter 3

Solution methods

Many solution techniques have been suggested for the pooling problem. They merely vary in how they deal with the bilinear terms that appear in tracking of the regulated qualities. We have divided the solution methods proposed in the literature into inexact and exact solution approaches.

3.1 Inexact/heuristic techniques

Heuristic algorithms are targeting large problem instances to find good solutions at reasonable computational cost without guaranteeing global optimality. Usually, these solutions are found by iteratively trying to improve a candidate solution with regard to a given measure.

3.1.1 Improvement heuristics

One of the earliest heuristic algorithm proposed to solve the pooling problem is the iterative method proposed by Haverly (1978). This method starts by estimating and fixing the pool qualities, and then the resulting linear programming is solved. The new qualities are calculated using the flow values from the solution of the linear program (LP). If the new and the old qualities coincide the method stops, otherwise it constructs a new LP using the new qualities and repeats these steps until it converges.

The solution returned by the iterative method depends on the initial guess of the values of the quality variables. Moreover, as pointed out in (Haverly,

1979, 1980), this method may not provide a feasible solution, and if it does, the solution it provides is not always the global optimum. Main (1993) observed that the iterative method is unstable in large instances. Practical implementations of the iterative method have been discussed by White and Trierwiler (1980) who used the distributive recursion, which is an improved version of Haverly's iterative method, at SoCal[1], where they managed to model and solve practical instances. As shown by Lasdon and Joffe (1990), the distributive recursion in some sense is more closely related to the successive linear-programming technique, a method that will be discussed in Section 3.1.3.

A more general heuristic for the iterative method has been suggested by Audet et al. (2004), and is referred to as the alternate heuristic (ALT). This heuristic is a two step algorithm that, starting from a feasible point, the first step freezes one set of the variables appearing in the bilinear terms, and solves the resulting for the remaining variables in the model. For example in the P-formulation (see Section 2.2.1), it fixes w_i^k $(i \in I,\ k \in K)$, and solves for f_{ij} for all $(i,j) \in A$. In the second step, the flow variables on arcs leaving the pools are fixed to the values given by the solution to the LP solved in the first step. The resulting LP is then solved for the quality variables and the flow variables on arcs leaving the sources. These two steps are repeated until a fixed point is reached.

The ALT heuristic hence alternates between two linear programs, each of which corresponds to fixing one set of variables occurring in bilinear terms. This contrasts the method of Haverly (1978), which corresponds to fixing the flow on all arcs in the second step.

All heuristic algorithms discussed so far are improvement heuristics, which, based on the fact that freezing one set of the variables that participate in the bilinear terms, results in a linear program (LP). Audet et al. (2004) also suggested a variable neighborhood search (VNS) heuristic, where the local search procedure is provided by the ALT heuristic. VNS initially defines a set of pre-selected neighborhood structures by modifying the feasible extreme points of the LP resulting from ALT. Starting with one neighborhood, this method moves from the current solution by finding a new solution using local search, where the starting point is drawn randomly within a neighborhood of the current solution. If the

[1]Standard oil Company of California (SoCal) is the old name of Chevron U.S.A. Corporation.

new solution does improve the current solution, it becomes the new current solution. Otherwise, it selects the next neighborhood and proceeds with the current solution. The algorithm repeats this until the maximum number of iterations is reached.

3.1.2 A construction heuristic

In Paper E (Alfaki and Haugland, 2011b), we propose a construction heuristic for the pooling problem. The heuristic considers a sequence of subgraphs, each of which contains a single terminal, and an associated bilinear program for optimizing the flow to the terminal. The optimal solution to each subproblem serves as a feasible augmentation of the total flow accumulated so far. Experimental results on 20 large-scale standard pooling problem instances indicate that, in large instances, our heuristic algorithm outperforms multi-start local optimization techniques provided by commercially available software. Our heuristic can also easily be extended to tackle generalized pooling problem instances.

3.1.3 Successive linear programming

Successive linear programming (SLP) has traditionally been used to solve the pooling problem in the petrochemical industries. This technique is also referred to as the method of approximate programming (MAP) by Griffith and Stewart (1961) of Shell oil company, who originally proposed and tested the approach on petroleum refinery optimization. Perhaps the major reasons for this popularity are its ability to employ available linear programming codes and solve large instances (Baker and Lasdon, 1985). The method starts with an initial guess of the variable values, approximates the bilinear terms using the Taylor's first order expansion at the initial guess, and then solves the resulting LP. The procedure is repeated with the LP solution as the new base of the Taylor expansion, until convergence to a fix point is obtained. Lasdon et al. (1979) applied SLP and the generalized reduced gradient algorithms to the pooling problem, where they showed some advantages over the iterative method of Haverly. Some improvements of the SLP are reported by Palacios-Gomez et al. (1982), Zhang et al. (1985), Baker and Lasdon (1985) and Sarker and Gunn (1997).

Many successful applications of the SLP technique in the leading oil and gas companies were reported in the literature. Among these is the work of Simon and Azma (1983) at Exxon, where the authors documented Exxon experience with the SLP technique implemented in the system PLATOFORM. Later, Baker and Lasdon (1985) described Exxon's attempt to unify the treatment of nonlinear functions appearing in their mathematical programming system, which was accomplished by the introduction of nonnegative deviation variables in the SLP linearized subproblem.

3.1.4 Benders decomposition

Benders (1962) has proposed a well-known and popular decomposition method for solving nonlinear optimization problems, where the variables are partitioned into complicating and non-complicating variables. The partition is made such that fixing the complicating variables reduces the problem to a linear program in the remaining variables, parametrized by the value of the complicating ones.

Based on Geoffrion's generalization of Benders decomposition (Geoffrion, 1972), Floudas and Aggarwal (1990) proposed a method that searches for a global solution to the pooling problem. Following the variable partition of Benders, the original problem can be partitioned into a subproblem where the complicating variables are fixed, and a master problem in the complicating variables. The method iterates between the subproblem and the master problem to identify an optimal solution. Despite the satisfactory behavior in some instances, this method could not guarantee convergence to a global solution.

3.1.5 Discretization approaches

To approximate the bilinear constraints, Tomasgard et al. (2007) and Rømo et al. (2009) discretized the quality variables which resulted in a mixed integer programming problem (MILP). A similar approach is used in (Faria and Bagajewicz, 2008) for the wastewater treatment problem, and replaced the bilinear constraints by "big M" constraints. Pushing in the same direction, Pham et al. (2009) and Pham (2007) eliminated the bilinear terms by discretizing the quality

variables. Consequently, the pooling problem is approximated by a mixed-integer programming problem.

In Paper D (Alfaki and Haugland, 2011a), we propose a method that linearizes the bilinear terms by discretizing the domain of the proportion variables into a fixed number of points. The resulting model serves as an approximation to the pooling problem. This approach is a generalization of the discretization approach proposed by Pham et al. (2009). Computational experiments on a set of large-scale generalized pooling problem instances show that this approach outperforms traditional solution methods where continuous models are used, even when a very coarse discretization is applied. With a fine discretization, however, the discretization approach implies a large computational effort.

3.2 Exact solution techniques

3.2.1 Branch-and-bound algorithms

Most exact global optimization methods are based on a branch-and-bound framework. In general, the branch-and-bound algorithm starts by partitioning the feasible region of the problem into two or more sub-regions (branching process), and constructs a relaxation for each sub-region. This yields a lower bound on the global minimum cost, possibly also an upper bound (bounding process). The branching process is then applied recursively, and defines a search tree in which the nodes represents the sub-regions. Nodes in the resulting search tree are pruned when its lower bounds exceed the best upper bound found so far. The algorithm halts when the tree is empty or the best lower and upper bounds are sufficiently close.

3.2.1.1 Primal-dual decomposition methods

In an effort to improve the method described in Section 3.1.4, Visweswaran and Floudas (1990) suggested the first global optimization algorithm based on a decomposition technique and branch-and-bound. The problem is decomposed into primal and dual subproblems to provide upper and lower bounds on the global solution. Gradient information of the Lagrange function is used to partition the

current domain into sub-domains, and the procedure is repeated until it converges to the global solution. Some improvements of this method are observed in e.g. (Visweswaran and Floudas, 1993) and (Androukakis et al., 1996).

3.2.1.2 Linear relaxation based algorithms

Linear relaxations of problems involving some bilinear function $f(x, y) = xy$, where $(x, y) \in D$, are obtained by the *convex* and *concave envelopes* (see Section 1.2) of f, denoted $\text{vex}_D f(x, y)$ and $\text{cav}_D f(x, y)$, respectively. It can be shown (see Al-Khayyal and Falk, 1983; McCormick, 1976) that the convex and concave envelopes of f on the rectangle $D = [\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}]$ are given by, respectively,

$$\text{vex}_D f(x, y) = \max\left\{\underline{y}x + \underline{x}y - \underline{x}\underline{y}, \overline{y}x + \overline{x}y - \overline{x}\overline{y}\right\}, \tag{3.1}$$

$$\text{cav}_D f(x, y) = \min\left\{\underline{y}x + \overline{x}y - \overline{x}\underline{y}, \overline{y}x + \underline{x}y - \underline{x}\overline{y}\right\}. \tag{3.2}$$

Linear relaxations of the pooling problem formulations given in Section 2.2 are obtained by replacing all occurrences of the bilinear terms by new variables, and by bounding each new variable between its corresponding envelopes. A branch-and-bound algorithm based on such relaxations, where in each iteration the rectangle is divided into four sub-rectangles, was first applied to the pooling problem by Foulds et al. (1992). Audet et al. (2004) suggested a branch-and-cut algorithm, which is an improvement of the above branch-and-bound technique.

The reformulation-linearization technique (RLT) (Sherali and Alameddine, 1992) is a methodology for constructing tight linear relaxations of a nonconvex problem. The second step, linearization, was already discussed above. The first step, reformulation, is to add new valid constraints obtained by multiplying two original constraints.

It is interesting to note that one can arrive to the McCormick's convex and concave envelopes (3.1)–(3.2) by applying the RLT to the bound constraints of x and y. For example, multiplying $(x - \underline{x}) \geq 0$ and $(\overline{y} - y) \geq 0$ yields the constraint $xy \leq \overline{y}x + \underline{x}y - \underline{x}\overline{y}$, which is one of the constraints suggested by (3.2).

Quesada and Grossmann (1995) applied the RLT to obtain a relaxation which is used within a spatial branch and bound algorithm that uses a nonlinear solver

to provide upper bounds. The result in several instances showed that a few branch-and-bound nodes were needed to verify the global solutions.

Sahinidis and Tawarmalani (2005) applied their branch-and-reduce algorithm, which uses the McCormick's relaxation as lower-bounding and local and random search as upper-bounding techniques. In addition, it uses various range reduction techniques. This algorithm is implemented in the generic global optimization code, BARON (Sahinidis, 1996), by use of the PQ-formulation. When applying their code to standard instances from the literature, they were able to reduce the running time and the size of the search tree significantly.

Liberti and Pantelides (2006) proposed an improved relaxation technique referred to as the reduced reformulation linearization technique (RRLT), and incorporated it in a spatial branch-and-bound algorithm. The authors also suggested an algorithm that automatically constructs this relaxation for large and sparse NLPs such as the pooling problem. They applied their algorithm to common pooling instances where the results showed that tight linear relaxations and hence faster convergence are provided.

Piecewise-linear relaxations have been proposed by Wicaksono and Karimi (2008) and Gounaris et al. (2009), who utilize piecewise linearization schemes by partitioning the original domain of the variables involved in the bilinear terms into smaller sub-domains. Applying the McCormick relaxation for each of the resulting sub-domains, and using binary variables to select the optimal sub-domain, resulted in an efficient relaxation that can be used in the branch-and-bound framework to accelerate convergence.

3.2.1.3 Lagrangian relaxation based algorithms

The Lagrangian relaxation is a useful technique when the problem's constraints can be decomposed into "difficult" and "easy" ones. The difficult constraints are relaxed by adding them to the objective with weight (*Lagrange multipliers*), and thereby the solution provides a lower bound on the global solution of the original problem. In the pooling problem, the difficult constraints are the bilinear ones. As shown in Chapter 2, these constraints arise from quality balances around pools and the quality bounds at terminals.

Ben-Tal et al. (1994) studied the Lagrangian relaxation of their Q-formulation (see Section 2.2.2.1 for details). The associated Lagrangian dual, which gives a lower bound on the minimum cost, is solved by analyzing the simplex $\left\{y \in \mathbb{R}^{S \times I} : \sum_{s \in N_i^-} y_i^s = 1\right\}$. This relaxation is integrated in a branch-and-bound algorithm that divides the simplex into smaller ones. Upper bounds on the global minimum cost are found by local search.

Adhya et al. (1999) introduced a Lagrangian relaxation by dualizing all the constraints in the P-formulation except for the variable bounds. The solution of the resulting Lagrangian subproblem is approximated by solving a sequence of MILPs. They also proved that the Lagrangian relaxation provides tighter lower bounds than standard linear relaxation does in the case of more than one quality parameter. A similar Lagrangian relaxation was suggested by Almutairi and Elhedhli (2009).

3.2.2 Semidefinite programming relaxations

In Paper C (Frimannslund et al., 2010), we suggest a technique based on a series of semidefinite programs (or linear matrix inequality (LMI) relaxations) to solve the pooling problem. LMI relaxations are used to turn general (nonconvex) optimization problems, where the objective and the constraints are polynomials, into a sequence of convex positive semidefinite programs (Lasserre, 2001a,b).

The general idea of this technique is as follows. Consider the optimization problem,

$$f^* = \min_{x \in \mathbb{R}^n} \{f_0(x) : x \in \Omega\}, \tag{3.3}$$

and assume for simplicity that Ω is compact and f_0 is continuous. Then the problem (3.3) can be turned into a convex problem by minimizing over the set, $\mathcal{B}(\Omega)$, of all Borel probability measures μ supported on Ω. The resulting optimization problem,

$$\mu^* = \min_{\mu \in \mathcal{B}(\Omega)} \int f_0(x) d\mu,$$

has the same global optimum value as the original problem. However, finding the probability distribution μ^* on the support Ω is done by characterizing its moment sequences, which is an infinite-dimensional convex optimization problem

known as the *moment problem* (Lasserre, 2010). Instead of solving an infinite-dimensional problem, a truncated moment sequences are determined, which can be cast as an LMI relaxation. By increasing the order of the moment sequences, a tighter relaxation is obtained.

By applying the above technique to the pooling problem with a single quality parameter, Frimannslund et al. (2010) show that if the feasible set has a nonempty interior, then we have a finite sequence of LMI relaxations with increasing order that converges to the global optimum. For a fixed relaxation order, this technique thus provides tight lower bounds for the global minimum cost. Based on the experiments, we show that for low order relaxations, the lower bound provided by this technique matches the true global optimum in several small instances.

3.3 Summary

A review of the literature on solution techniques proposed to solve the pooling problem is given in this chapter. These techniques are classified as improvement heuristics, successive linear programming, decomposition techniques and branch-and-bound algorithms. Global optimization algorithms are quite effective for instances of small to moderate size. In larger instances, however, global optimizers fail to converge in reasonable time, while existing local optimizers depend largely on good initial guesses.

Chapter 4

Summary of papers

In this chapter, we give an overview of the five papers constituting the thesis. Two of the papers are focused on modeling (see Section 4.1), while the topic of the others is solution methods (see Section 4.2). With the exception of Paper E, each paper was presented in at least one international conference. Papers A and C are direct extensions of conference papers (not included).

4.1 Strong formulations

4.1.1 The standard pooling problem (Paper A)

Paper A, entitled "Strong formulations for the pooling problem" and authored by Mohammed Alfaki and Dag Haugland, is published in the *Journal of Global optimization*, doi: `10.1007/s10898-012-9875-6`. A short version of the paper is presented at *Toulouse Global Optimization Workshop* (TOGO 2010) in Toulouse, France, and published in the same conference proceedings (Alfaki and Haugland, 2010). The paper was also presented at the *4th Nordic Optimization Symposium*, 2010, in Århus, Denmark.

In this paper, we develop new formulations for the standard pooling problem based on terminal proportion variables (see Section 2.2.2.2). In the strongest model, we combine source and terminal proportion variables. This formulation is proved to be stronger than other formulations based uniquely on source proportions or quality variables. A new branching strategy that performs well with the strongest formulation is presented.

Main contributions:

- We give a formal proof of the *strong* $\mathcal{NP}$-hardness[1] of the pooling problem by constructing a polynomial reduction from the *maximum independent vertex set problem* (MIVS)[2]. We also prove that the strong $\mathcal{NP}$-hardness persists in two interesting special cases: (i) If the networks have only one pool, (ii) If we consider maximizing the total flow instead of minimizing the total cost.
- We extend the idea of proportion variables in the PQ-formulation (see Section 2.2.2.1), and give two new formulations. The first is the TP-formulation (see Section 2.2.2.2), which contrasts the PQ-formulation in that it uses terminal proportions instead of source proportions. This formulation is comparable to the PQ-formulation in terms of strength. The second new formulation is the STP-formulation (see Section 2.2.2.2), which combines source and terminal proportions, and proves to have stronger linear relaxation than both the PQ-, and TP-formulations.
- The strength of the STP-formulation is tested on a set of well-studied pooling problem instances from the literature, and on large randomly generated instances. Computational experiments have confirmed the strength of the STP-formulation. In most of the instances, the STP-formulation turned out to yield stronger lower bounds than the competing formulations.
- We develop a special branching rule suitable for the STP-formulation, which is incorporated in a branch-and-bound algorithm. Computational experiments with an implementation of the algorithm indicate that this branching rule helps to improve the lower bound on the global optimum.

4.1.2 The generalized pooling problem (Paper B)

Paper B, entitled "A multi-commodity flow formulation for the generalized pooling problem" and authored by Mohammed Alfaki and Dag Haugland, is published in *Journal of Global optimization*, doi: `10.1007/s10898-012-9890-7`. A prelimi-

[1] A problem is a *strongly* $\mathcal{NP}$-hard if it remains $\mathcal{NP}$-hard even when all of its parameters are bounded by a polynomial in the input size.

[2] The MIVS problem is to find a largest possible subset of the vertices in a graph, such that all pairs of vertices in the subset are non-neighbors.

nary version was presented at the *3rd Nordic Optimization Symposium*, 2009, in Stockholm, Sweden.

In this paper, we develop a multi-commodity flow formulation for the generalized pooling problem (see Section 2.3.2). This formulation is a generalization of a well-established formulation for the standard problem, and we demonstrate that it is stronger than alternative, popular formulations from the literature.

Main contributions:

- We develop a multi-commodity flow formulation for the generalized pooling problem. When applied to the standard pooling problem, this formulation coincides with PQ-formulation.
- We prove that our new formulation is stronger than both the P-formulation (see Section 2.2.1) and the HYB-formulation (see Section 2.3.1).
- The paper also presents computational experiments of 40 instances with up to 35 nodes and 12 quality attributes confirming that the suggested formulation performs better. They confirm that the suggested formulation enables faster computation of (strong bounds on) the global optimum.

4.2 New solution methods

4.2.1 LMI relaxations (Paper C)

A short version of Paper C, which is entitled "Solving the pooling problem with LMI relaxations" and authored by Lennart Frimannslund, Mohamed El Ghami, Mohammed Alfaki, and Dag Haugland, is presented at *Toulouse Global Optimization Workshop* (TOGO 2010) in Toulouse, France, and published in the conference proceedings (Frimannslund et al., 2010) (reviewed). In the thesis, we include the complete version of the paper.

In this paper, we suggest a solution framework based on a sequence of LMI relaxations (see Section 3.2.2) to solve standard pooling problems with a single quality parameter. We have considered both the maximum flow and the minimum cost versions of the problem. Based on our experiments, we show that standard pooling instances with a single quality parameter can be solved at low LMI

relaxation orders. However, solving such relaxation implies a large computational effort, which for large instances makes the method impractical to use.

Main contributions:

- We suggest a technique that provides tight lower bounds for the global objective function value, which, under certain conditions, converges monotonically to the global optimum. The method applies to the standard pooling problem with a single quality parameter.
- The experiments show that, in several small instances, the maximum flow and minimum cost versions of the problem can be solved at LMI relaxation order 2 and 3, respectively.

4.2.2 A discretization approach (Paper D)

Paper D, entitled "Comparison of discrete and continuous models for the pooling problem" and authored by Mohammed Alfaki and Dag Haugland, is presented at the *11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems* (ATMOS'11), 2011, in Saarbrücken, Germany. The paper is reviewed and published in the conference proceedings.

In this paper, we generalize the discretization approach proposed in (Pham et al., 2009) to the generalized pooling problem. This method approximates the problem by discretizing the proportion variables into a given number of points, and consequently, the resulting model is a mixed-integer programming problem (see Section 3.1.5). Through numerical experiments on large scale instances, we compare our discrete formulation with a continuous formulation. The purpose of this is to investigate whether discrete models are more suitable for finding good solutions when the global optimum is out of reach. By lower bounding techniques, we also aim to estimate the error introduced by discretizing the solution space.

Main contributions:

- We propose a method that linearizes the bilinear terms at the cost of introducing binary variables. The resulting model serves as an approximation to the generalized pooling problem.

- Computational experiments on a set of large-scale instances show that this approach is superior to approaches that use continuous variables, even when a very coarse discretization is applied.

4.2.3 A construction heuristic method (Paper E)

Paper E, entitled "Computing feasible solutions to the pooling problem" and authored by Mohammed Alfaki and Dag Haugland, is submitted to *Annals of Operations Research* (the first review is currently in progress).

In this paper, we give a construction heuristic for the standard pooling problem. It constructs a sequence of subgraphs, each of which contains a single terminal, and associated bilinear programs for optimizing the flow to the terminal. The optimal solution to each bilinear program serves as a feasible augmentation of the total flow accumulated so far. The suggested method is designed to give good feasible solutions, particularly in large instances, and does not guarantee to find the optimal solution. In order to keep the work focused, we confine the study to the standard version of the problem, but it is straightforward to extend the proposed method to the generalized version.

Main contributions:

- We develop a greedy construction heuristic method for the pooling problem (see Section 3.1.2).
- Experimental results on 20 large-scale instances indicate that, in large instances, our heuristic algorithm outperforms multi-start local optimization techniques provided by commercially available software.

Chapter 5

Conclusion and future work

5.1 Conclusion

In the last several decades, many inexact and exact solution methods to approach the pooling problem have been suggested in the literature. Although inexact solution methods are intended for large problem instances, most of these are under the influence of starting points, and hence can easily be trapped at weak solutions. On the other hand, most exact solution methods are based on branch-and-bound algorithms requiring strong relaxations in order to converge fast. The thesis has contributed to both modeling and algorithmic aspects by developing strong formulations and efficient solution methods for the pooling problem.

By constructing a polynomial reduction from the MIVS problem, we have proved formally that the pooling problem is strongly $\mathcal{NP}$-hard. We have also showed that the problem remains strongly $\mathcal{NP}$-hard when there is only one pool and no direct arcs from sources to terminals. Concerning the development of strong formulation for the standard pooling problem, we have derived new formulations based on terminal proportion variables. In the strongest model, referred to as the STP-formulation, we have combined source and terminal proportion variables. This formulation is proved to be stronger than competing formulations from the literature. Along with this model, we have suggested a new branching strategy that exploits the redundancy in the STP-formulation. The strength of the formulation and effectiveness of the branching strategy are demonstrated experimentally.

For the generalized pooling problem, we have proposed a multi-commodity flow formulation. The proposed model is an extension of the PQ-formulation for the standard version of the problem. We have proved that our multi-commodity flow formulation has stronger relaxation than state-of-the-art formulations from the literature. Experiments also confirm that ours performs better.

Three solution methods have been suggested in the thesis. A procedure based on a sequence of LMI relaxations has been proposed to solve the pooling problem with a single quality parameter. The experiments indicate that small instances of this problem can be solved with LMI relaxation of order 2 or 3. However, solving such relaxations implies a large computational effort.

We have also developed a mixed integer programming model, serving as an approximation to the pooling problem, by discretizing the proportion variables. Hence, bilinear constraints are transformed into linear ones, at the computational cost represented by the introduction of binary variables. Computational experiments on a set of large-scale instances show that a discrete model is superior to its continuous ancestor.

In the last approach, we have developed a construction heuristic for the pooling problem. It considers a sequence of subgraphs, each of which contains a single terminal, and an associated bilinear program for optimizing the flow to the terminal. The optimal solution serves as a feasible augmentation of the total flow accumulated so far. Experimental results indicate that, in large instances, our heuristic algorithm outperforms multi-start local optimization techniques provided by commercially available software.

5.2 Future work

Regarding the method developed in Paper D, a topic for future research is to develop an adaptive discretization rule. Computations can be saved if the number of discretization points can be kept small, while gradually focusing the search on solution sets of decreasing size. This can also be integrated in a branch-and-bound algorithm to provide tighter upper bounds on the global optimum.

In Paper E, instead of iterating the construction heuristic on the set of terminals, we will investigate a similar heuristic that considers the set of sources.

Extending the idea for other extensions of the pooling problem is also a possible direction for future work.

In the models for pipeline transportation of natural gas, we have extended the traditional flow models by adding constraints that track the flow quality throughout the system. Additional type of constraints that seem to be of particular importance when allocating flow in natural gas transportation networks is the interrelation between pipeline flow and pressure. That is, the inlet pressure of a pipeline is a function of upstream flow, and the outlet pressure is given by the inlet pressure and the pipeline flow. Increasing the flow implies a reduction of the outlet pressure, which in its turn reduces the flow capacity of downstream links. Inlet and outlet pressure variables must be introduced for each pipeline, and the pipeline flow must be modeled as a (nonlinear) function of these. In the ideal case, this is accomplished by use of e.g. the Weymouth equation (Osiadacz, 1987), which states that the square of the flow is proportional to the difference of the squares of inlet and outlet pressures. That takes the following form:

$$f_{ij}^2 = W_{ij}\left(p_i^2 - p_j^2\right), \quad (i,j) \in A.$$

where f_{ij} is the flow through pipeline $(i,j) \in A$, W_{ij} is a constant which depends on pipeline physical properties, and p_i is the pressure at node $i \in N$. Under certain conditions, such relations can be formulated in terms of convex constraints. However, when inhomogeneous flow streams are pooled, this is no longer realistic, and the capacity constraints become nonconvex.

When two or more pipelines meet at a junction node (pool), flow streams of unequal pressure are leveled to the smallest pressure value. In order to model this, a binary variable is required for each valve that can be either open or closed. Consequently, an otherwise continuous flow model is transferred into a discrete model, and thereby becomes much more difficult to solve. In fact, taking the above mentioned constraints into account, the pipeline transportation model of natural gas becomes a mixed integer nonlinear program (MINLP). In the suggested future work, we will investigate how the flow problems mentioned above can be formulated and solved. We will in particular study how solution approaches based on models with proportion variables can be applied.

Bibliography

Adhya, N., Tawarmalani, M., and Sahinidis, N.V. (1999). A Lagrangian approach to the pooling problem. *Industrial & Engineering Chemistry Research*, **38** (5), 1956–1972.

Al-Khayyal, F.A., and Falk, J.E. (1983). Jointly constrained biconvex programming. *Mathematics of Operations Research*, **8** (2), 273–286.

Alfaki, M., and Haugland, D. (2010). Strong formulations for the pooling problem. In S. Cafieri, B. G.-Tóth, E. Hendrix, L. Liberti and F. Messine (Eds.), *Proceedings of the Toulouse Global Optimization Workshop* (pp. 15–18).

Alfaki, M., and Haugland, D. (2011a). Comparison of discrete and continuous models for the pooling problem. In A. Caprara and S. Kontogiannis (Eds.), *11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems* (Vol. 20, pp. 112–121). OpenAccess Series in Informatics (OASIcs). Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

Alfaki, M., and Haugland, D. (2011b). Computing feasible solutions to the pooling problem. *Submitted to: Annals of Operations Research*. First review in progress.

Alfaki, M., and Haugland, D. (2012a). A multi-commodity flow formulation for the generalized pooling problem. *Journal of Global Optimization*. doi:10.1007/s10898-012-9890-7

Alfaki, M., and Haugland, D. (2012b). Strong formulations for the pooling problem. *Journal of Global Optimization*. doi:10.1007/s10898-012-9875-6

Almutairi, H., and Elhedhli, S. (2009). A new Lagrangian approach to the pooling problem. *Journal of Global Optimization*, **45** (2), 237–257.

Amos, F., Rönnqvist, M., and Gill, G. (1997). Modelling the pooling problem at the New Zealand Refining Company. *Journal of the Operational Research Society*, **48** (8), 767–778.

Androukakis, I.P., Visweswaran, V., and Floudas, C.A. (1996). Distributed decomposition-based approaches. In C. Floudas and P. Pardalos (Eds.), *State of the Art in Global Optimization: Computational Methods and Applications* (pp. 285–301). Kluwer Academic, Dordrecht.

Armagan, E. (2009). *Decomposition Algorithms for Global Solution of Deterministic and Stochastic Pooling Problems in Natural Gas Value Chains*. (Master's thesis, Massachusetts Institute of Technology, Cambridge, USA).

Audet, C., Brimberg, J., Hansen, P., Le Digabel, S., and Mladenović, N. (2004). Pooling problem: Alternate formulations and solution methods. *Management science*, **50** (6), 761–776.

Baker, T.E., and Lasdon, L.S. (1985). Successive linear programming at Exxon. *Management Science*, **31** (3), 264–274.

Ben-Tal, A., Eiger, G., and Gershovitz, V. (1994). Global minimization by reducing the duality gap. *Mathematical Programming*, **63** (2), 193–212.

Benders, J.F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, **4** (1), 238–252.

Boyd, S.P., and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge, UK: Cambridge University Press.

Dantzig, G.B. (1949). Programming of interdependent activities: II mathematical model. *Econometrica, Journal of the Econometric Society*, 200–211.

Faria, D.C., and Bagajewicz, M.J. (2008). A new approach for the design of multi-component water/wastewater networks. *Computer Aided Chemical Engineering*, **25**, 43–48.

Fieldhouse, M. (1993). The pooling problem. In T. Ciriani and R. Leachman (Eds.), *Optimization in Industry: Mathematical Programming and Modeling Techniques in Practice* (pp. 223–230). New York, USA: John Wiley & Sons Ltd.

Floudas, C.A. (2000). *Deterministic Global Optimization: Theory, Methods, and Applications*. Springer.

Floudas, C.A., and Aggarwal, A. (1990). A decomposition strategy for global optimization search in the pooling problem. *Operations Research Journal On Computing*, **2** (3), 225–235.

Foulds, L.R., Haugland, D., and Jörnsten, K. (1992). A bilinear approach to the pooling problem. *Optimization*, **24** (1), 165–180.

Frimannslund, L., El Ghami, M., Alfaki, M., and Haugland, D. (2010). Solving the Pooling Problem with LMI Relaxations. In S. Cafieri, B. G.-Tóth, E. Hendrix, L. Liberti and F. Messine (Eds.), *Proceedings of the Toulouse Global Optimization Workshop* (pp. 51–54).

Geoffrion, A.M. (1972). Generalized benders decomposition. *Journal of Optimization Theory and Applications*, **10** (4), 237–260.

Gounaris, C.E., Misener, R., and Floudas, C.A. (2009). Computational comparison of piecewise–linear relaxation for pooling problems. *Industrial & Engineering Chemistry Research*, **48** (12), 5742–5766.

Griffith, R.E., and Stewart, R.A. (1961). A nonlinear programming technique for the optimization of continuous processing systems. *Management Science*, **7**, 379–392.

Guo, B., and Ghalambor, A. (2005). *Natural Gas Engineering Handbook*. Houston, USA: Gulf Publishing Company.

Haverly, C.A. (1978). Studies of the behavior of recursion for the pooling problem. *ACM SIGMAP Bulletin*, **25**, 19–28.

Haverly, C.A. (1979). Behavior of recursion model-more studies. *ACM SIGMAP Bulletin*, **26**, 22–28.

Haverly, C.A. (1980). Recursion model behavior: More studies. *ACM SIGMAP Bulletin*, **28**, 39–41.

Hendrix, E.M.T., and G.-Tóth, B. (2010). *Introduction to Nonlinear and Global Optimization*. Cambridge, UK: Springer.

Horst, R., Pardalos, P.M., and Thoai, N.V. (2000). *Introduction to Global Optimization* (Second). Dordrecht, The Netherlands: Kluwer Academic Publishers.

Hubbard, R. (2009). The role of gas processing in the natural-gas value chain. *Journal of Petroleum Technology*, **61** (8), 65–71.

Kantorovich, L.V. (1940). A new method of solving some classes of extremal problems. *Doklady Academy of Sciences USSR*, **28**, 211–214.

Karmarkar, N. (1984). A new polynomial time algorithm for linear programming. *Combinatorica*, **4**, 373–395.

Lasdon, L.S., and Joffe, B. (1990). The relationship between distributive recursion and successive linear programming in refining production planning models. In *Proceedings of the NPRA Computer Conference, October 29–31, Seattle, Washington*.

Lasdon, L.S., Waren, A.D., Sarkar, S., and Palacios, F. (1979). Solving the pooling problem using generalized reduced gradient and successive linear programming algorithms. *ACM Sigmap Bulletin*, **27**, 9–15.

Lasserre, J.B. (2001a). Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, **11** (3), 796–817.

Lasserre, J.B. (2001b). New positive semidefinite relaxations for nonconvex quadratic programs. In H. Hadjisavvas and P. Paradalos (Eds.), *Advances in Convex Analysis and Global Optimization: Honoring the Memory of C. Caratheodory (1873-1950)* (pp. 161–189). Kluwer Academic Publishers.

Lasserre, J.B. (2010). *Moments, Positive Polynomials and Their Applications*. London, UK: Imperial College Press.

Li, X., Tomasgard, A., and Barton, P.I. (2011a). Decomposition strategy for the stochastic pooling problem. *Journal of Global Optimization*, 1–26.

Li, X., Armagan, E., Tomasgard, A., and Barton, P.I. (2011b). Stochastic pooling problem for natural gas production network design and operation under uncertainty. *AIChE Journal*, **57** (8), 2120–2135.

Liberti, L., and Pantelides, C.C. (2006). An exact reformulation algorithm for large nonconvex NLPs involving bilinear terms. *Journal of Global Optimization*, **36** (2), 161–189.

Main, R.A. (1993). Large recursion models: Practical aspects of recursion techniques. In T. Ciriani and R. Leachman (Eds.), *Optimization in Industry: Mathematical Programming and Modeling Techniques in Practice* (pp. 241–249). New York, USA: John Wiley & Sons Ltd.

McCormick, G.P. (1976). Computability of global solutions to factorable nonconvex programs: part I - convex underestimating problems. *Mathematical Programming*, **10** (1), 147–175.

Meyer, C.A., and Floudas, C.A. (2006). Global optimization of a combinatorially complex generalized pooling problem. *AIChE Journal*, **52** (3), 1027–1037.

Misener, R., and Floudas, C.A. (2009). Advances for the pooling problem: Modeling, global optimization, and computational studies. *Applied and Computational Mathematics*, **8** (1), 3–22.

Misener, R., and Floudas, C.A. (2010). Global optimization of large-scale generalized pooling problems: Quadratically constrained MINLP models. *Industrial & Engineering Chemistry Research*, **49** (11), 5424–5438.

Mokhatab, S., Poe, W.A., and Speight, J.G. (2006). *Handbook of Natural Gas Transmission and Processing*. Gulf Professional Publishing.

Nesterov, Y., and Nemirovskii, A. (1994). *Interior-point Polynomial Algorithms in Convex Programming*. SIAM Studies in Applied Mathematics. Philadelphia, USA: Society for Industrial and Applied Mathematics (SIAM).

Nocedal, J., and Wright, S.J. (2000). *Numerical Optimization*. Springer Verlag.

Osiadacz, A. (1987). *Simulation and analysis of gas networks*. Houston, USA: Gulf Publishing Company.

Palacios-Gomez, F., Lasdon, L.S., and Engquist, M. (1982). Nonlinear optimization by successive linear programming. *Management Science*, **28** (10), 1106–1120.

Pham, V. (2007). *A Global Optimization Approach to Pooling Problems in Refineries*. (Master's thesis, Department of Chemical Engineering, Texas A&M University, Texas, USA).

Pham, V., Laird, C., and El-Halwagi, M. (2009). Convex hull discretization approach to the global optimization of pooling problems. *Industrial & Engineering Chemistry Research*, **48** (4), 1973–1979.

Quesada, I., and Grossmann, I.E. (1995). Global optimization of bilinear process networks with multi-component flows. *Computers & Chemical Engineering*, **19** (12), 1219–1242.

Rømo, F., Tomasgard, A., Hellemo, L., Fodstad, M., Eidesen, B.H., and Pedersen, B. (2009). Optimizing the Norwegian natural gas production and transport. *Interfaces*, **39** (1), 46–56.

Sahinidis, N.V. (1996). BARON: A general purpose global optimization software package. *Journal of Global Optimization*, **8** (2), 201–205.

Sahinidis, N.V., and Tawarmalani, M. (2005). Accelerating branch-and-bound through a modeling language construct for relaxation-specific constraints. *Journal of Global Optimization*, **32** (2), 259–280.

Sarker, R.A., and Gunn, E.A. (1997). A simple SLP algorithm for solving a class of nonlinear programs. *European Journal of Operational Research*, **101** (1), 140–154.

Sherali, H.D., and Alameddine, A. (1992). A new reformulation-linearization technique for bilinear programming problems. *Journal of Global Optimization*, **2**, 379–410.

Simmons, D., Horlings, H., Cronshaw, I., and Others. (2006). *Natural Gas Market Review 2006: Towards a Global Gas Market*. International Energy Agency.

Simon, J.D., and Azma, H.M. (1983). Exxon experience with large scale linear and nonlinear programming applications. *Computers & Chemical Engineering*, **7** (5), 605–614.

Takama, N., Kuriyama, T., Shiroko, K., and Umeda, T. (1980). Optimal water allocation in a petroleum refinery. *Computers & Chemical Engineering*, **4** (4), 251–258.

Tawarmalani, M., and Sahinidis, N.V. (2002). *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Tomasgard, A., Rømo, F., Fodstad, M., and Midthun, K. (2007). Optimization models for the natural gas value chain. In G. Hasle, K. Lie and E. Quak (Eds.), *Geometric Modelling, Numerical Simulation, and Optimization: Applied Mathematics at SINTEF* (pp. 521–558). Springer.

Visweswaran, V., and Floudas, C.A. (1990). A global optimization algorithm (GOP) for certain classes of nonconvex NLPs–II. Application of theory and test problems. *Computers & chemical engineering*, **14** (12), 1419–1434.

Visweswaran, V., and Floudas, C.A. (1993). New properties and computational improvement of the GOP algorithm for problems with quadratic objective functions and constraints. *Journal of Global Optimization*, **3** (4), 439–462.

Wardzinski, J., Foss, M.M., and Delano, F. (2004). Interstate natural gas–quality specifications and interchangeability. *Center for Energy Economics, Bureau of Economic Geology, University of Texas at Austin, December*.

White, D.L., and Trierwiler, L.D. (1980). Distributive recursion at SoCal. *ACM SIGMAP Bulletin*, **28**, 22–38.

Wicaksono, D.S., and Karimi, I.A. (2008). Piecewise MILP under- and over-estimators for global optimization of bilinear programs. *AIChE Journal*, **54** (4), 991–1008.

Wolsey, L.A. (1998). *Integer Programming*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons Ltd.

Zhang, J.H., Kim, N.H., and Lasdon, L. (1985). An improved successive linear programming algorithm. *Management Science*, **31** (10), 1312–1331.

Part II

Scientific contributions

Paper A

Strong formulations for the pooling problem[2]

Mohammed Alfaki[1] and Dag Haugland[1]

[1]Department of Informatics, University of Bergen,
P.O. Box 7803, N-5020 Bergen, Norway.
mohammeda@ii.uib.no and dag@ii.uib.no
[2]In: *Journal of Global Optimization*, doi: 10.1007/s10898-012-9875-6, 2012.

Strong formulations for the pooling problem

Mohammed Alfaki[1] and Dag Haugland[1]

1 Department of Informatics, University of Bergen,
P.O. Box 7803, N-5020 Bergen, Norway.
mohammeda@ii.uib.no and dag@ii.uib.no

Abstract

The pooling problem is a well-studied global optimization problem with applications in oil refining and petrochemical industry. Despite the strong $\mathcal{NP}$-hardness of the problem, which is proved formally in this paper, most instances from the literature have recently been solved efficiently by use of strong formulations. The main contribution from this paper is a new formulation that proves to be stronger than other formulations based on proportion variables. Moreover, we propose a promising branching strategy for the new formulation and provide computational experiments confirming the strength of the new formulation and the effectiveness of the branching strategy.

Keywords Pooling problem · Bilinear programming · Global optimization · Linear relaxation · Computational complexity

1 Introduction

The pooling problem can be considered as an extension of the minimum cost flow problem on networks with three layers of nodes. Raw materials of unequal quality are supplied at the sources, and are first mixed in intermediate nodes (pools). At the terminals, end products are formed by blending the output from the pools, possibly also with direct supply from the sources. The resulting qualities of the end products thus depend on what sources they originate from, and in what proportions. Restrictions, which may vary between the terminals, apply

to these qualities. This problem is typically modeled as a bilinear, non-convex optimization problem. When the intermediate nodes or pools are not needed, this problem is called the blending problem, which can be formulated as a linear program (LP). A typical pooling problem network is depicted in Fig. 1.

There are two main categories of formulations for the pooling problem: The P-formulation (Haverly, 1978) consists of flow and quality variables, whereas the Q-formulation (Ben-Tal et al., 1994) uses flow proportions instead of quality variables. The Q-formulation has later been shown to perform better when fed into generic branch-and-cut algorithms. Adding new nonlinear constraints to the Q-formulation by means of the reformulation linearization technique (RLT), which was first derived by Quesada and Grossmann (1995) and later refined by Sherali et al. (1998) and Sherali and Adams (1999), gives a stronger formulation called the PQ-formulation (Sahinidis and Tawarmalani, 2005).

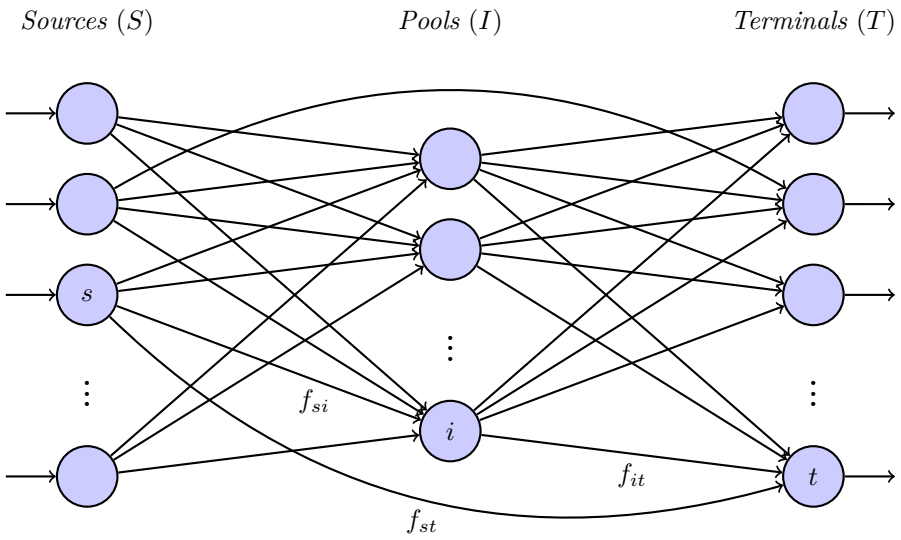


Figure 1: A typical pooling problem network.

Many solution techniques for the pooling problem have been proposed in the literature. Generally, these techniques can be classified into local and global optimization techniques. Haverly (1978) studied the method of fixing all quality variables to anticipated values, and then optimize the flow by solving the resulting

LP. Next, the flow is fixed to an optimal solution to this LP, and the procedure is repeated until a fix point is reached. It was demonstrated that, even for very small instances, the final solution depends on the initial guess. Audet et al. (2004) also used this method to compute good feasible solutions quickly.

One of the most frequently used local optimization techniques for the pooling problem is successive linear programming (SLP), which can be regarded as an extension of the method above. It performs in the same manner, except that the bilinear terms are approximated using the Taylor's first order expansion in both variables rather than a function in only one of the variables (Baker and Lasdon, 1985; Griffith and Stewart, 1961; Haverly, 1979; Palacios-Gomez et al., 1982; Sarker and Gunn, 1997; Zhang et al., 1985). A more advanced local method based on the generalized Benders decomposition was proposed by Floudas and Aggarwal (1990).

The first approach for the pooling problem that guarantees convergence to a global solution was the Global Optimization Algorithm (GOP) (Floudas and Visweswaran, 1990; Visweswaran and Floudas, 1990), which was based on duality theory and Lagrangian relaxation. Other Lagrangian-based techniques have been developed by Ben-Tal et al. (1994), Audet et al. (2000), and Almutairi and Elhedhli (2009). Global techniques based on relaxations by convex and concave envelopes (McCormick, 1976) providing lower bounds on the global solution, have been inspired by the work of Al-Khayyal and Falk (1983). Based on this idea, a pooling problem algorithm using a standard branching step of branch-and-bound global optimization was first given by Foulds et al. (1992). Sahinidis and Tawarmalani (2005) applied their branch-and-reduce algorithm, which is implemented in the generic global optimization code (BARON) (Sahinidis, 1996), to the PQ-formulation. When applying their code to standard instances from the literature, they were able to reduce the running time and the size of the search tree significantly.

In the last decade, most of the progress in the pooling problem has been achieved by integrating different relaxation techniques into global optimization algorithm such as branch-and-bound. These relaxation techniques include the reduced reformulation linearization technique (Liberti and Pantelides, 2006) and the piecewise linear relaxation (Gounaris et al., 2009). To tighten the linear re-

laxation, the technique is incorporated in a special branch-and-bound algorithm, and may automatically replace some of the bilinear constraints by linear ones.

The literature also shows some formulations of extensions of the pooling problem. A hybrid formulation with both quality and proportion variables for a generalization where arcs between pools exist was proposed by Audet et al. (2004). Mixed integer nonlinear formulations for another generalization involving decisions regarding the existence of pools and interconnectivity of the network were proposed in (Meyer and Floudas, 2006) and (Karuppiah and Grossmann, 2006). The models have applications in the design of waste-water treatment networks. It is demonstrated that through careful partitioning of the variable domain, the suggested branch-and-bound algorithms for the models converge to the global optimum. Recently, Misener et al. (2010) proposed a formulation for an extended version of the pooling problem that involves complex emission constraints.

This article is organized as follows: Section 2 defines some notation used throughout the text and defines the pooling problem in rigorous terms. In Section 3, we prove formally that the pooling problem is strongly $\mathcal{NP}$-hard, and thereby that no compact LP formulation exists unless $\mathcal{P} = \mathcal{NP}$. Section 4 presents the PQ-formulation, which is the most successful formulation known to date. In Section 5, we extend the idea in (Ben-Tal et al., 1994) and (Sahinidis and Tawarmalani, 2005), and give two new formulations. The first new formulation contrasts the PQ-formulation in that it uses terminal proportions as opposed to source proportions, and is comparable to the PQ-formulation in terms of strength. The second combines source and terminal proportions, and proves to be stronger than both its competitors. In Section 6, we describe a branch-and-bound algorithm, including a branching strategy tailored for the strongest new formulation. Section 7 reports results from experiments with BARON and our own algorithm.

2 Notation and preliminaries

To say that x is a (non-negative) real vector defined over a set S, we write $x \in \mathbb{R}^S$ ($x \in \mathbb{R}^S_+$), and the component corresponding to $i \in S$ is written x_i. An analogous notation with multiple subscripts, or a combination of subscripts and superscripts, is used for components of vectors defined over products of sets.

Consider a directed acyclic graph $D = (N, A)$ with node set N and arc set A, and with three disjoint sets of nodes S, I, and T where $N = S \cup I \cup T$. For any node $i \in N$, let $N_i^+ = \{j \in N : (i,j) \in A\}$ and $N_i^- = \{j \in N : (j,i) \in A\}$ denote the sets of out- and in-neighbors of i, respectively. We assume that S and T are non-empty, and refer to them as the sets of *sources* and *terminals*, respectively. We also assume that $A \subseteq (S \times I) \cup (S \times T) \cup (I \times T)$, implying $N_s^- = \emptyset \ \forall s \in S$ and $N_t^+ = \emptyset \ \forall t \in T$. We refer to all nodes in I as *pools*. Let $P \subseteq S \times I \times T$ be the set of paths with two arcs in D.

We define a finite set of *quality attributes* K. For each attribute $k \in K$, we associate a real constant with each source and each terminal: Define q_s^k as the *quality parameter* of attribute k at source $s \in S$, and define q_t^k as the *quality bound* of attribute k at terminal $t \in T$. For each $i \in N$, we define the constant *flow capacity* b_i, and for each arc $(i,j) \in A$, we define the constant *unit cost* c_{ij}. Reflecting the cost of purchasing raw material at s and the revenues of selling end products at t, we typically have $c_{si} \geq 0 \geq c_{it}$ for all $(s,i,t) \in P$, but the results of the current work do not rely on this assumption.

Define the *flow polytope* $\mathcal{F}(D,b)$ as the set of $f \in \mathbb{R}_+^A$ satisfying

$$\begin{aligned} \sum_{j \in N_s^+} f_{sj} &\leq b_s, && s \in S, \\ \sum_{j \in N_i^-} f_{ji} &\leq b_i, && i \in N \setminus S, \\ \sum_{j \in N_i^+} f_{ij} - \sum_{j \in N_i^-} f_{ji} &= 0, && i \in I. \end{aligned}$$

For any $f \in \mathcal{F}(D,b)$, we say that $w \in \mathbb{R}^{N \times K}$ is a *quality matrix* associated with f if the elements of w for all $k \in K$ are defined as $w_i^k = q_i^k$ if $i \in S$ and

$$w_i^k \sum_{j \in N_i^-} f_{ji} = \sum_{j \in N_i^-} w_j^k f_{ji}, \quad i \in N \setminus S. \tag{1}$$

Hence, for all attributes $k \in K$, the quality w_i^k associated with a node $i \in N \setminus S$ is assumed to be a weighted average of the quality of entering flow, where the flow values constitute the weights. This interpretation assumes that the total

entering flow is non-zero, and otherwise, w_i^k can take an arbitrary value. Any flow on an arc leaving node i gets quality w_i^k.

The problem can now be defined formally:

Problem 1 (The pooling problem)**.** *Find $f \in \mathcal{F}(D, b)$ with associated quality matrix $w \in \mathbb{R}^{N \times K}$ satisfying $w_t^k \leq q_t^k \ \forall t \in T, k \in K$, such that $\sum_{(i,j)\in A} c_{ij} f_{ij}$ is minimized.*

Some authors give the pooling problem a definition that differs slightly from Problem 1. For instance, both lower and upper bounds on the terminal qualities are introduced by Adhya et al. (1999). This is not an extension of Problem 1, since it can be modeled by introducing a new attribute k' for each $k \in K$, and letting $q_i^{k'} = -q_i^k$ for all $i \in S$. In the same paper, bounds on the pool qualities are defined, which leads to an extension of Problem 1. The same can be said about the arc flow bounds and the lower bounds on the node flows introduced by e.g. Audet et al. (2000). However, the extensions mentioned here are rather simple, and are unlikely to affect the conclusions of the present work.

3 Computational complexity

Audet et al. (2004) state that the general bilinear programming problem is strongly $\mathcal{NP}$-hard, but do not address the question whether this carries over to the pooling problem. A review of the comprehensive literature on the pooling problem shows a consensus that the problem, in some sense, is hard to solve. In this section, we express this folklore result in more rigorous terms.

The *maximum independent vertex set problem* (MIVS), which is known to be $\mathcal{NP}$-hard (Garey and Johnson, 1979), amounts to finding a largest possible subset of the vertices in a simple graph, such that all pairs of vertices in the subset are non-neighbors.

Proposition 2. *The pooling problem is strongly $\mathcal{NP}$-hard.*

Proof. The proof is by a polynomial reduction from MIVS. Let $G = (V, E)$ be a simple graph with vertex set V and edge set E. Define the corresponding pooling problem instance $\mathcal{P}_G$ to consist of $n = |V|$ sources and terminals, one

pool, and $2n$ quality attributes, and write $S = \{s_v : v \in V\}$, $T = \{t_v : v \in V\}$, $K = \{k_v^+ : v \in V\} \cup \{k_v^- : v \in V\}$, and $I = \{p\}$. We let $A = (S \times I) \cup (I \times T)$.

For notational simplicity, we introduce, for all $i \in N$, w_i^{v+} and w_i^{v-} as short hand notations for $w_i^{k_v^+}$ and $w_i^{k_v^-}$, respectively, and do analogously for components of q. The quality parameters at source s_v are defined as

$$q_{s_v}^{u+} = \begin{cases} 1, & u = v, \\ 0, & u \neq v, \end{cases} \quad \text{and} \quad q_{s_v}^{u-} = \begin{cases} -1, & u = v, \\ 0, & u \neq v. \end{cases}$$

The quality bounds at terminal t_v are defined as

$$q_{t_v}^{u+} = \begin{cases} 0, & \{u, v\} \in E, \\ 1, & \{u, v\} \notin E, \end{cases} \quad \text{and} \quad q_{t_v}^{u-} = \begin{cases} \frac{-1}{n}, & u = v, \\ 0, & u \neq v. \end{cases}$$

Finally, we let $b_p = n$, and for all $s \in S$ and $t \in T$, we let $b_s = b_t = 1$, $c_{sp} = 0$, and $c_{pt} = -1$. It follows that all non-zero input parameters are bounded by n, and thereby also by the size of the input.

Clearly, there exist non-zero flow vectors that are feasible in $\mathcal{P}_G$. Let f be one such vector, and let w be the associated quality matrix. Assume $f_{pt_u} > 0$ and $f_{pt_v} > 0$ for some $u, v \in V$. Since each terminal has p as its unique in-neighbor, we have by (1) that $w_{t_u}^k = w_{t_v}^k = w_p^k \; \forall k \in K$. Because $w_{t_u}^{u-} \leq q_{t_u}^{u-} < 0$, we hence get $w_p^{u-} < 0$, implying $f_{s_u p} > 0$ since

$$w_p^{u-} = \frac{\sum_{s \in S} q_s^{u-} f_{sp}}{\sum_{s \in S} f_{sp}} = \frac{-f_{s_u p}}{\sum_{s \in S} f_{sp}}.$$

Analogously, $f_{pt_v} > 0$ implies $f_{s_v p} > 0$, which in its turn yields

$$w_p^{v+} = \frac{\sum_{s \in S} q_s^{v+} f_{sp}}{\sum_{s \in S} f_{sp}} = \frac{f_{s_v p}}{\sum_{s \in S} f_{sp}} > 0.$$

Since $0 < w_p^{v+} = w_{t_u}^{v+} \leq q_{t_u}^{v+}$, we have by the definition of $q_{t_u}^{v+}$ that $\{u, v\} \notin E$. Hence, $\{v \in V : f_{pt_v} > 0\}$ is an independent vertex set, and since $f_{pt} \leq 1$ for all $t \in T$, we also have $\sum_{(i,j) \in A} c_{ij} f_{ij} = -\sum_{t \in T} f_{pt} \geq -|\{v \in V : f_{pt_v} > 0\}|$.

Conversely, assume $V' \subseteq V$ is an independent vertex set in G. Letting

$$f_{s_v p} = f_{p t_v} = \begin{cases} 1, & v \in V' \\ 0, & v \notin V', \end{cases}$$

yields, for all $v \in V'$, that

$$w^{u+}_{t_v} = \begin{cases} \frac{1}{|V'|}, & u \in V' \\ 0, & u \notin V' \end{cases} \quad \text{and} \quad w^{u-}_{t_v} = \begin{cases} \frac{-1}{|V'|}, & u \in V' \\ 0, & u \notin V' \end{cases}$$

Consequently, $w^k_{t_v} \leq q^k_{t_v}$ for all $k \in K$, and f is feasible in $\mathcal{P}_G$ with objective function value $\sum_{(i,j)\in A} c_{ij} f_{ij} = -|V'|$. It follows that any optimal solution to $\mathcal{P}_G$ identifies a maximum independent vertex set in G. Thus, there exists a polynomial reduction from MIVS to the pooling problem with all parameters bounded by (a polynomial in) the input size, which completes the proof. □

The proof of Proposition 2 shows that the strong $\mathcal{NP}$-hardness of the pooling problem persists in two interesting special cases.

Corollary 3. *The pooling problem for networks with only one pool is strongly $\mathcal{NP}$-hard.*

An LP-formulation of any problem is said to be *compact* if its numbers of constraints and variables are bounded by some polynomial in the input size. Corollary 3 shows that, unless $\mathcal{P} = \mathcal{NP}$, there exists no compact LP-formulation of the pooling problem for networks with $|I| = 1$. This contrasts the facts that in each of the cases $|S| = 1$ and $|T| = 1$, compact LP-formulations are easy to establish.

Consider the *maximum flow pooling problem* defined by replacing the minimization of $\sum_{(i,j)\in A} c_{ij} f_{ij}$ by the maximization of $\sum_{t\in T} \sum_{i\in N^-_t} f_{it}$ in Problem 1.

Corollary 4. *The maximum flow pooling problem is strongly $\mathcal{NP}$-hard, even for the set of instances with only one pool.*

The polynomial reduction defined in the proof of Proposition 2 results in problem instances where $|K|$ grows linearly with the network size. If we restrict the set of instances such that $|K|$ is not allowed to grow beyond some upper bound, the proof no longer applies. If the instance class also is restricted to networks with only one pool and no bypass arcs, the problem can be solved in polynomial time.

Proposition 5. *Let κ be any natural number. For the instance class where $|I| = 1$, $A = (S \times I) \cup (I \times T)$ and $|K| \leq \kappa$, Problem 1 can be solved in polynomial time.*

Proof. Consider any flow vector $f \in \mathcal{F}(D, b)$, and let $w \in \mathbb{R}^K$ be the corresponding quality vector of the pool. Define the associated vector $\overline{w} \in \mathbb{R}^K$ such that

$$\overline{w}^k = \inf \left\{ q_t^k : q_t^k \geq w^k, t \in T \right\}$$

is the strictest (if any) quality bound corresponding to attribute $k \in K$ that w satisfies. It follows that $\overline{w}^k$ equals either $-\infty$ or q_t^k for some $t \in T$. Let $W \subseteq \mathbb{R}^K$ be the set of possible values $\overline{w}$ thus can take, and observe that $|W| \leq (|T| + 1)^\kappa$.

Denote $I = \{p\}$. We have that (f, w) is a feasible solution if and only if $\overline{w}^k \leq q_t^k$ for all $k \in K$ and all $t \in T$ for which $f_{pt} > 0$. An optimal solution is hence found by enumerating all $\overline{w} \in W$, and identifying the best optimal solution to all LPs

$$\begin{aligned}
&\min_{f,w} \sum_{(i,j)\in A} c_{ij} f_{ij} \\
\text{s.t.} \quad & f \in \mathcal{F}(D, b), \\
& \sum_{s\in S} q_s^k f_{sp} \leq \overline{w}^k \sum_{s\in S} f_{sp}, && \forall k \in K, \\
& f_{pt} = 0, && \forall t \in T : q_t^k < \overline{w}^k \text{ for some } k \in K.
\end{aligned}$$

The proof is complete by observing that there are only a polynomial number of LPs to be solved, and each can be solved in polynomial time. □

4 The PQ-formulation

In the P-formulation of the pooling problem, the elements of the quality matrices of the pools are introduced as decision variables. This is contrasted by the Q- and PQ-formulations, which do not make explicit use of quality variables.

4.1 A model with flow and proportion variables

Let $f_{ij} = 0$ if $i, j \in N$ and $(i, j) \notin A$. Define y_i^s as the proportion of the flow through pool $i \in I$ originating from source $s \in S$ (let $y_i^s = 0$ if $(s, i) \notin A$). That is, if the flow through i is non-zero, we have $y_i^s = f_{si} / \sum_{t \in T} f_{it}$. For all paths $(s, i, t) \in P$, define x_{sit} as the flow along the path, and let $x_{sit} = 0$ for all $(s, i, t) \in (S \times I \times T) \setminus P$. Combining the path flow variables and the proportion variables with the flow variables introduced in the previous section, we arrive at the PQ-formulation (Sahinidis and Tawarmalani, 2005) written as:

$$
\begin{aligned}
[\text{PQ}] \quad \min_{f,y,x} \quad & \sum_{s \in S} \sum_{t \in T} \left(c_{st} f_{st} + \sum_{i \in I} (c_{si} + c_{it})\, x_{sit} \right) && && (2)\\
\text{s.t.} \quad & \sum_{t \in T} \left(f_{st} + \sum_{i \in I} x_{sit} \right) \leq b_s, && s \in S, && (3)\\
& \sum_{s \in S} \sum_{t \in T} x_{sit} \leq b_i, && i \in I, && (4)\\
& \sum_{s \in S} \left(f_{st} + \sum_{i \in I} x_{sit} \right) \leq b_t, && t \in T, && (5)\\
& \sum_{s \in S} \left(q_s^k - q_t^k \right) \left(f_{st} + \sum_{i \in I} x_{sit} \right) \leq 0, && t \in T,\ k \in K, && (6)\\
& \sum_{s \in S} y_i^s = 1, && i \in I, && (7)\\
& \sum_{s \in S} x_{sit} = f_{it}, && i \in I,\ t \in T, && (8)\\
& \sum_{t \in T} x_{sit} \leq b_i y_i^s, && s \in S,\ i \in I, && (9)\\
& x_{sit} = y_i^s f_{it}, && (s, i, t) \in P, && (10)
\end{aligned}
$$

$$f_{it} \geq 0, \qquad i \in S \cup I,\ t \in T, \quad (11)$$

$$0 \leq y_i^s \leq 1, \qquad i \in I,\ s \in S. \quad (12)$$

We observe that, due to the introduction of the new variables, the formulation needs flow variables only for arcs pointing at a terminal. That is, the flow along arc (s, i), where $s \in S$ and $i \in I$, is represented by $\sum_{t \in T} x_{sit}$. This is exploited in the source and pool capacity constraints (3)–(4). Constraints (5) and (6) express the capacity and the quality bound at the terminals. Correctness of (6) follows by observing that according to (1), we have in case terminal t receives flow, that

$$w_t^k \sum_{s \in S} \left(f_{st} + \sum_{i \in I} x_{sit} \right) = \sum_{s \in S} q_s^k \left(f_{st} + \sum_{i \in I} x_{sit} \right).$$

Then $w_t^k \leq q_t^k$ yields (6).

Constraint (10) ensures that the flow along path (s, i, t) equals the flow on arc (i, t) times the proportion of the flow at pool i coming from source s, and by (7), the sum of these proportions is 1.

Constraints (8)–(9) are redundant, but as shown in (Sahinidis and Tawarmalani, 2005), these cuts strengthen the formulation significantly. Their interpretation is respectively that the flow on arc (i, t) equals the total flow on paths intersecting the arc, and that path (s, i, t) can consume no more than the proportion y_i^s of the flow capacity of pool i.

Problem (2)-(12) may appear to be different from the PQ-formulation as suggested by Sahinidis and Tawarmalani (2005). Unlike theirs, our version of the PQ-formulation contains one variable (x_{sit}) for each occurring bilinear term ($y_i^s f_{it}$). By virtue of constraint (10), it is justified to replace each occurrence by the new variable. Apart from this substitution, the two versions are identical. Further, as long as the traditional relaxation technique explained in Section 4.2 is applied, the relaxations of the two versions have identical strength.

4.2 Linear relaxation

A linear relaxation of (2)–(12) is constructed by bounding x_{sit} between the convex and concave envelopes of $y_i^s f_{it}$ for all $(s, i, t) \in P$. For all $(i, t) \in I \times T$, let $\underline{f}_{it}$

and $\overline{f}_{it}$ be some lower an upper bounds on the variable f_{it}. A corresponding notation is applied for all other variables. It is trivial to find finite bounds between which any feasible value of f_{it} lies. For example, it is valid to put $\underline{f}_{it} = 0$ and $\overline{f}_{it} = \min\{b_i, b_t\}$.

It can be shown (Al-Khayyal and Falk, 1983; McCormick, 1976) that the convex and concave envelopes of yf (for notational convenience, we drop sub- and superscripts here) on the rectangle $C = \left[\underline{y}, \overline{y}\right] \times \left[\underline{f}, \overline{f}\right]$, are

$$\begin{aligned} \mathrm{Vex}_C(yf) &= \max\left\{\underline{y}f + \underline{f}y - \underline{f}\underline{y}, \overline{y}f + \overline{f}y - \overline{f}\overline{y}\right\}, \quad \text{and} \\ \mathrm{Cav}_C(yf) &= \min\left\{\underline{y}f + \overline{f}y - \overline{f}\underline{y}, \overline{y}f + \underline{f}y - \underline{f}\overline{y}\right\}, \end{aligned}$$

respectively. For any rectangle $C = \left[\underline{y}, \overline{y}\right] \times \left[\underline{f}, \overline{f}\right] \subset \mathbb{R}^2$, define the polyhedron

$$\mathcal{H}\left[C\right] = \left\{(y, f, x) \in \mathbb{R}^3 : \mathrm{Vex}_C(yf) \leq x \leq \mathrm{Cav}_C(yf), (y, f) \in C\right\}.$$

Note that it is straightforward to find a rectangle C^s_{it} enclosing all feasible (y^s_i, f_{it}). The linear relaxation of the PQ-formulation is thus obtained by replacing the bilinear constraint (10) by the linear inequalities that impose

$$(y^s_i, f_{it}, x_{sit}) \in \mathcal{H}\left[C^s_{it}\right], \quad i \in I, s \in S, t \in T. \tag{13}$$

5 Strong formulations with terminal proportions

The formulation introduced in Section 4.1 has proportion variables corresponding to sources, and flow variables on arcs between pools and terminals. Symmetric to the PQ-formulation, we suggest in this section a formulation with proportion variables corresponding to terminals, and flow variables on arcs between sources and pools.

Define for all pools $i \in I$, y^t_i as the proportion of the flow at i destined for terminal $t \in T$. That is, we let $y^t_i = f_{it} / \sum_{s \in S} f_{si}$ when the latter sum is positive. The path flow variables and the flow variables on arcs going directly from a source

to a terminal are kept in the new formulation. Henceforth, we refer to it as the TP-formulation, which is given as follows:

$$\text{[TP]} \quad \min_{f,y,x} \quad \sum_{s\in S}\sum_{t\in T}\left(c_{st}f_{st} + \sum_{i\in I}(c_{si}+c_{it})\,x_{sit}\right)$$

$$\text{s.t.} \quad (3)\text{–}(6),$$

$$\sum_{t\in T} y_i^t = 1, \qquad i \in I, \tag{14}$$

$$\sum_{t\in T} x_{sit} = f_{si}, \qquad i \in I,\ s \in S, \tag{15}$$

$$\sum_{s\in S} x_{sit} \le b_i y_i^t, \qquad i \in I,\ t \in T, \tag{16}$$

$$x_{sit} = y_i^t f_{si}, \qquad (s,i,t) \in P, \tag{17}$$

$$f_{si} \ge 0, \qquad s \in S,\ i \in I \cup T, \tag{18}$$

$$0 \le y_i^t \le 1, \qquad i \in I,\ s \in S. \tag{19}$$

The interpretation of the constraints is analogous to the PQ-formulation. Since the path flow variables are shared between the two formulations, the objective functions become identical. The same applies to the node capacity constraints (3)–(5) and the quality constraints (6). Constraints (14)–(19) can be considered as the TP-variant of constraints (7)–(12) from the PQ-formulations.

Constraints (15) and (16) are redundant. Following the pattern of the PQ-formulation, they are included to strengthen the relaxation.

A numerical comparison to the PQ-formulation shows that the formulations do not in general have equal strength, but none dominates the other (see Section 7.2 for a comparison on instances from the literature).

The full benefit of the new proportion variables is achieved when they are combined with source proportions in the same model. It follows from the definition of y_i^s and y_i^t that $y_i^s f_{it}$ and $y_i^t f_{si}$ both can be interpreted as the flow along the path (s,i,t). Given this observation, a formulation based on source and ter-

minal proportions (denoted the STP-formulation) can be derived by combining the variables and the constraints from both models:

$$
\begin{aligned}
[\text{STP}] \quad \min_{f,y,x} \quad & \sum_{s\in S}\sum_{t\in T}\left(c_{st}f_{st} + \sum_{i\in I}\left(c_{si}+c_{it}\right)x_{sit}\right) \\
\text{s.t.} \quad & (3)\text{–}(10), \\
& (14)\text{–}(17), \\
& f_{ij} \geq 0, && (i,j)\in A, \\
& 0 \leq y_i^s, y_i^t \leq 1, && i\in I,\ s\in S,\ t\in T.
\end{aligned}
$$

We refer to the linear relaxations as defined in Section 4.2 of the PQ-, TP- and STP-formulations, respectively, as the PQ-, TP- and STP-relaxations. Assume that the same variable bounds are applied in all three relaxations.

Proposition 6. *The lower bound on the optimal objective function value provided by the STP-relaxation is at least as tight as those provided by the PQ- and TP-relaxations.*

Proof. That the STP-bound is no weaker than the PQ-bound follows by observing that the STP-formulation inherits all linear constraints of the PQ-formulation, and that the relaxation inherits the constraints (13). Superiority over the TP-bound is proved analogously. □

Even in small instances, the STP-relaxation may give tighter bounds than its two competitors. Some standard test instances from the literature where this occurs, can be found in Section 7.2 below.

6 Branch-and-bound implementation

Besides having a stronger relaxation, the STP-formulation opens for a special branching technique (see Section 6.2) that does not apply to the other formulations studied in this work. We have implemented a simple branch-and-bound algorithm in C++ for the purpose of testing and comparing the branching technique to a traditional one.

6.1 The search tree

At each node in the branch-and-bound tree, the LP relaxation is solved using CPLEX 10.2. In the root node, before solving the initial LP relaxation, we tighten the variable bounds by applying the optimality based bounds tightening (OBBT) procedure (Shectman and Sahinidis, 1998). That is, each variable is minimized and maximized over the constraints of the LP relaxation, and the extreme values hence found constitute the (new) variable bounds. Whenever the OBBT procedure generates solutions that are feasible in the original problem, that is when (10) or (17) is satisfied, we also have an upper bound on the global optimum.

The search tree can be pruned either by local optimality (the relaxed bilinear constraints are satisfied), or by bound (the LP relaxation is infeasible or its minimum cost is larger than the best upper bound). When expanding an open node, we choose a variable pair by use of one of the strategies discussed in Section 6.2. The rectangle enclosing these variables is split into four new axis-parallel rectangles by two straight lines intersecting the relaxed optimum, and the corresponding four LP relaxations are solved next. The node to be expanded is chosen according to the best bound rule. Upper bounds on the minimum cost are found only when the solution to some relaxation is feasible in the original problem, and we make no attempt to find feasible solutions by application of a local optimizer.

6.2 Branching strategies

In this section, let (f, y, x) refer to an optimal solution to an LP relaxation of either of the three pooling problem formulations studied in this work. Let C^s_{it} be defined as in Section 4.2, and with reference to the TP- and STP-formulations, let C^t_{si} denote an axis-parallel rectangle enclosing a subset of feasible values of (f_{si}, y^t_i).

Consider a node in the search tree, where the LP relaxation, defined by the set(s) of rectangles $\{C^s_{it} : (s,i,t) \in P\}$ and/or $\{C^t_{si} : (s,i,t) \in P\}$ that apply, has optimal solution (f, y, x). Define for all $(s,i,t) \in P$ the infeasibility measures

$$g^s_{it} = \max\left\{f_{it}y^s_i - \mathrm{Vex}_{C^s_{it}}\left(f_{it}y^s_i\right), \mathrm{Cav}_{C^s_{it}}\left(f_{it}y^s_i\right) - f_{it}y^s_i\right\}, \quad (s,i,t) \in P,$$

$$g_{si}^t = \max\left\{f_{si}y_i^t - \mathrm{Vex}_{C_{si}^t}\left(f_{si}y_i^t\right), \mathrm{Cav}_{C_{si}^t}\left(f_{si}y_i^t\right) - f_{si}y_i^t\right\}, \quad (s,i,t) \in P.$$

We consider two strategies for choosing the variables to branch on:

The `max-infeas` strategy applies to all three formulations. In the PQ-formulation (the TP-formulation), we choose some $(s,i,t) \in P$ maximizing g_{it}^s (g_{si}^t), and branch on the corresponding pair of variables. In the STP-formulation, both g_{it}^s and g_{si}^t are defined, and the selection criterion of the `max-infeas` strategy is to maximize $\max\{g_{it}^s, g_{si}^t\}$. We then branch on either (f_{it}, y_i^s) or (f_{si}, y_i^t), depending on which of g_{it}^s and g_{si}^t is the larger.

The redundancy in the STP-formulation offers an alternative strategy. If either $g^S = \max\{g_{it}^s : (s,i,t) \in P\} = 0$ or $g^T = \max\{g_{si}^t : (s,i,t) \in P\} = 0$ then the relaxed solution is feasible in the original problem. That both g^S and g^T are zero is however not a necessary feasibility condition. This suggests the `min-max-infeas` strategy, defined exclusively for the STP-formulation, where the selection of branching variables is based on the smaller of g^S and g^T. If $g^S \leq g^T$, the variables are chosen as suggested in the `max-infeas` strategy for the PQ-formulation. Otherwise, the same strategy for the TP-formulation is followed.

7 Computational results

The purpose of the experiments reported in this section is to compare the strengths of the PQ-, TP- and STP-formulations, to evaluate the ability of a generic global optimizer and our own optimizer to locate (or bound from below) the global optimum, and to compare the two branching strategies of Section 6.2.

All experiments reported in this work were conducted on a computer equipped with quad-core 3.00 GHz processors, where each group of four cores share 8 GB of memory.

7.1 Test instances from the literature

In the first part of the experiments, we study 14 problem instances from the open literature that are widely used to assess the performance of the algorithms

and the strength of the formulations for the pooling problem. We refer to them by letting Adhya1–4 denote Examples 1–4 in (Adhya et al., 1999), Bental4–5 denote Problems 4–5 in (Ben-Tal et al., 1994), Foulds2–5 denote Examples 2–5 in (Foulds et al., 1992), Haverly1–3 denote the instances in (Haverly, 1978), and finally by letting RT2 be the instance with the same notation in (Audet et al., 2000). The latter instance is an extension of Problem 1 since positive lower bounds on the flow entering the terminals are introduced. We have included it in our tests since the extension is trivial, and RT2 is included in most published experimental studies of the pooling problem. In (Audet et al., 2000), the number of quality attributes in RT2 is reported to be 4, whereas we define it to be 8. This is because the model in the cited reference also admits lower quality bounds at the terminals, whereas our formulations handle this by introducing new attributes as explained in Section 2.

Table 1: Size of standard test instances in each formulation.

Instance	$\|S\|$	$\|I\|$	$\|T\|$	$\|K\|$	PQ-/TP-formulation			STP-formulation		
					vars	nlts	lcs	vars	nlts	lcs
Adhya1	5	2	4	4	33	20	42	46	40	57
Adhya2	5	2	4	6	33	20	50	46	40	65
Adhya3	8	3	4	6	52	32	62	72	64	85
Adhya4	8	2	5	4	58	40	55	76	80	75
Bental4	4	1	2	1	13	6	15	18	12	21
Bental5	5	3	5	2	92	60	53	119	120	83
Foulds2	6	2	4	1	36	16	30	48	32	44
Foulds3	11	8	16	1	672	512	219	832	1024	387
Foulds4	11	8	16	1	672	512	219	832	1024	387
Foulds5	11	4	16	1	608	512	147	704	1024	247
Haverly1	3	1	2	1	10	4	13	14	8	18
Haverly2	3	1	2	1	10	4	13	14	8	18
Haverly3	3	1	2	1	10	4	13	14	8	18
RT2	3	2	3	8	34	18	46	46	36	60

The first 5 columns of Table 1 give the instance identifier and size in terms of $|S|$, $|I|$, $|T|$ and $|K|$. Succeeding columns show the number of variables (vars), the number of distinct bilinear terms (nlts) and the number of linear constraints (lcs) for each formulation applied to each problem instance. It is easily verified that the

PQ- and TP-formulations are equal in this respect, whereas, as the table confirms, the number of bilinear terms is doubled when moving to the STP-formulation. Also the numbers of variables and constraints grow, and it is therefore likely that the STP-relaxation takes more computation time than the other two relaxations. We have not reported the number of nonlinear constraints since in all formulations presented in this paper, the number of nonlinear constraints equals the number of distinct bilinear terms.

7.2 Strength of the LP relaxations

In Table 2, we have reported the optimal objective function value of the initial PQ-, TP- and STP-relaxation in the problem instances collected from the literature. Unless all relaxations give the same value, the best values are given in bold. In the last column, the minimum objective function value of the original problem is given.

Table 2: Comparing the strengths of the relaxations in instances collected from the literature.

Instance	Objective function values of the relaxations			Global solution
	PQ-relaxation	TP-relaxation	STP-relaxation	
Adhya1	**-840.27**	-856.25	**-840.27**	-549.80
Adhya2	-574.78	-574.78	-574.78	-549.80
Adhya3	-574.78	-574.78	-574.78	-561.05
Adhya4	**-961.93**	-967.44	**-961.93**	-877.65
Bental4	-550.00	**-541.67**	**-541.67**	-450.00
Bental5	-3500.00	-3500.00	-3500.00	-3500.27
Foulds2	-1100.00	-1100.00	-1100.00	-1100.00
Foulds3	-8.00	-8.00	-8.00	-8.00
Foulds4	-8.00	-8.00	-8.00	-8.00
Foulds5	-8.00	-8.00	-8.00	-8.00
Haverly1	-500.00	-500.00	-500.00	-400.00
Haverly2	-1000.00	-1000.00	-1000.00	-600.00
Haverly3	**-800.00**	-875.00	**-800.00**	-750.00
RT2	-6034.87	**-5528.25**	**-5528.25**	-4391.83

The table shows that the STP-relaxation improves the lower bound of the PQ-relaxation in two instances, namely Bental4 and RT2. Likewise, it performs

better than the TP-relaxation in instances Adhya1, Adhya4, and Haverly3. In all instances reported in Table 2, the best bound produced by either the PQ- or the TP-relaxation is however even with the STP-bound.

7.3 Computing the global optimum in standard test instances

To investigate whether improved strength leads to more efficient computation of the global optimum, we applied BARON 1.8.5 (Sahinidis, 1996) as global solver to each formulation instantiated by each instance in Table 1. The absolute and relative optimality tolerances of BARON are set to 10^{-6} and 10^{-9}, respectively, while all other options are set to the default values. The linear relaxations are solved by CPLEX 11.1.1.

In all experiments with the PQ-formulation, we applied the variant (2)–(12), where each bilinear term occurs exactly once in a dedicated constraint (10). We checked that the performance of BARON is comparable to what is obtained by the original version of the PQ-formulation (Sahinidis and Tawarmalani, 2005), and found that standard instances from the literature gave only small performance variation. Version (2)–(12) gave generally smaller subtrees, while the original formulation gave slightly shorter running time.

The `RELAXATION_ONLY_EQUATIONS` construct of BARON was used to introduce the RLT constraints (8) and (9). This is an option to force the optimizer to ignore the constraints in the local search (Sahinidis and Tawarmalani, 2005). In the experiments with the TP-formulation, the constraints (15) and (16) were introduced by use of the same option. In the STP-formulation, we applied the option to all constraints involving the variables y_i^t and f_{si}, that is (14)–(18).

The results are reported in Table 3, which shows the size of the search tree in terms of number of nodes (#nodes) and the running time in CPU-seconds for each instance. When the number of nodes is reported to be zero, optimality was found and proved during the preprocessing procedure of BARON.

As seen from Table 3, the STP-formulation requires fewer nodes than both the PQ- and the TP-formulation, and it solves 12 out of 14 instances in the root node. The corresponding number for the other formulations is 9 and 11, respectively. The largest search tree had only 21 nodes (in instance Adhya3), whereas the PQ-

Table 3: Computational results from BARON applied to the PQ-, TP- and STP-formulations in instances collected from the literature.

Instance	PQ-formulation		TP-formulation		STP-formulation	
	#nodes	time (sec)	#nodes	time (sec)	#nodes	time (sec)
Adhya1	18	0.17	9	0.20	17	0.44
Adhya2	13	0.15	13	0.18	1	0.36
Adhya3	29	0.23	39	0.25	21	0.80
Adhya4	1	0.09	1	0.12	1	0.10
Bental4	1	0.00	1	0.01	1	0.01
Bental5	1	0.32	0	0.21	0	0.02
Foulds2	1	0.02	0	0.02	1	0.05
Foulds3	0	0.31	0	2.90	0	0.80
Foulds4	0	3.77	0	1.82	0	0.48
Foulds5	0	0.46	0	2.55	0	4.88
Haverly1	1	0.00	1	0.00	1	0.01
Haverly2	5	0.01	1	0.01	1	0.01
Haverly3	1	0.00	1	0.01	1	0.01
RT2	13	0.14	1	0.08	1	0.10

and TP-formulations implied trees with respectively 29 and 39 nodes in the worst case (instance Adhya3 for both). Due to the larger LP-relaxations, however, the smaller search tree does not in all instances lead to savings in running time.

BARON is very fast for all formulations in nearly all instances. By applying the STP-formulation, all instances were solved in less than one CPU-second, except for instance Foulds5 where it took 5 seconds. The PQ- and TP-formulations needed more than one second in one and three instances, respectively, but needed less than respectively 4 and 3 CPU-seconds in all instances.

7.4 Comparing branching techniques

To compare the two branching strategies for the STP-formulation, implemented in the algorithm of Section 6, we have conducted computational experiments with the instances listed in Table 1. We have set the absolute and the relative optimality gap to 10^{-6} and 10^{-9}, respectively, and regarded the solution of the LP relaxation as feasible in the original problem if $\min\left\{g^S, g^T\right\} < 10^{-6}$. The results of the computational experiments for each branching strategy are reported in

Table 4. For each of the strategies `max-infeas` and `min-max-infeas`, we report the same types of results as reported in Table 3.

Table 4 shows that in instances Adhya1–2 and Adhya4, strategy `min-max-infeas` is considerably more efficient than its competitor. This is reflected by both the size of the search tree and the CPU-time. For the other instances, no significant differences in the running time can be observed.

We observe that for both strategies, the search trees become larger than those produced by BARON (Table 3), regardless of the formulation. This can be explained by the fact that we have not implemented any upper-bounding technique. With good upper bounds available early in the process, like those provided by the heuristics implemented in BARON, nodes with weak lower bounds can be discarded. Without such upper bounds, pruning is postponed until the solution to some linear relaxation turns out to be feasible, and the corresponding upper bound is sufficiently sharp. Consequently, our code will in general require larger search trees.

Table 4: Comparison of the branching techniques designed for the STP-formulation.

Instance	`max-infeas`		`min-max-infeas`	
	#nodes	time (sec)	#nodes	time (sec)
Adhya1	2157	3.83	441	0.72
Adhya2	1604	2.53	480	0.73
Adhya3	441	0.87	410	0.80
Adhya4	1026	3.01	34	0.10
Bental4	5	0.00	5	0.01
Bental5	8	0.08	2	0.06
Foulds2	4	0.01	2	0.01
Foulds3	0	0.32	0	0.33
Foulds4	0	2.13	0	2.13
Foulds5	0	1.76	0	1.77
Haverly1	5	0.00	5	0.00
Haverly2	5	0.00	5	0.00
Haverly3	5	0.00	5	0.00
RT2	63	0.11	73	0.12

7.5 Randomly generated large instances

Experiments reported so far confirm that standard pooling problem instances from the literature can be solved very fast. To make more interesting evaluations of the formulations suggested in this paper, experiments with larger instances are needed. We have generated randomly a set of 20 instances divided into 3 groups (denoted A, B and C), where all instances in the same group have equal numbers of sources, pools, terminals and quality attributes. The number of arcs is not constant within a group. We introduce arc (i, j), where $(i, j) \in (S \times I) \cup (S \times T) \cup (I \times T)$, with a probability, referred to as the *expected network density*, given as input to the instance generation procedure. Details on network sizes and range of expected densities for each group are reported in Table 5. All instances are generated such that for each $k \in K$, there is some $k' \in K$, $k' \neq k$, satisfying $q_s^{k'} = -q_s^k$ for all $s \in S$ (see Section 2). In other words, there are both an upper and a lower quality bound associated with each attribute at the terminals. For each instance in the large-scale set, we report in Table 6 the number of variables (vars), bilinear terms (nlts), and linear constraints (lcs) for each formulation. The first character in the instance identifier reported in this table refers to the group to which the instance belongs.

All instances and formulations considered in this work can be downloaded in GAMS-format from http://www.ii.uib.no/~mohammeda/spooling.

Table 5: Large-scale instance characteristics.

Group	#inst.	Size of instances in group				Range of expected density
		$\|S\|$	$\|I\|$	$\|T\|$	$\|K\|$	
A	10	20	10	15	24	0.30 – 0.80
B	6	35	17	21	34	0.30 – 0.80
C	4	60	15	50	40	0.20 – 0.35

7.6 Computational experiments with large-scale instances

To assess the performance of the PQ-, TP-, and STP-formulations, we submitted each instance in Table 6 to BARON. In order to study the effect of the two branching strategies applicable to the STP-formulation (see Section 6.2), we also

Table 6: The size of the large-scale instances.

Instance	PQ-/TP-formulation			STP-formulation		
	vars	nlts	lcs	vars	nlts	lcs
A0	500	329	531	616	658	657
A1	540	361	541	666	722	677
A2	677	485	555	817	970	705
A3	756	538	562	903	1076	719
A4	979	731	592	1156	1462	779
A5	1245	968	611	1441	1936	817
A6	1364	1083	624	1573	2166	843
A7	1825	1500	661	2071	3000	917
A8	1895	1530	667	2147	3060	929
A9	2399	1992	701	2685	3984	997
B0	1537	1153	1093	1826	2306	1399
B1	2354	1839	1180	2730	3678	1573
B2	3739	3093	1273	4208	6186	1759
B3	5327	4537	1385	5908	9074	1983
B4	7534	6591	1494	8224	13182	2201
B5	8991	7947	1566	9753	15894	2345
C0	3637	2826	2356	4233	5652	2982
C1	5840	4770	2533	6613	9540	3336
C2	7998	6720	2674	8913	13440	3619
C3	10567	9116	2828	11635	18232	3926

submitted the instances to our own code. Unfortunately, we found no way to dictate BARON to apply branching strategy `min-max-infeas`. For all runs of both optimizers, we set the time limit to one CPU-hour, and set the relative optimality tolerance to 10^{-2}.

By default, BARON uses the *probing* technique (Ryoo and Sahinidis, 1995) to reduce the variable bounds. In preliminary experiments with the randomly generated instances, BARON spent in average 92% of its time succeeding the pre-processing phase on probing. By deactivating the probing technique, BARON consistently performed better in all instances in Table 6, regardless of the formulation. In all these instances, we therefore report uniquely the results obtained without application of the said bound reduction technique. Note however that

results for the instances of smaller scale (Table 3) are obtained with the probing activated, which turned out to perform better in these instances.

The results of the computational experiments are reported in Table 7. The first column gives the instance identifier, columns 2–4 give the optimal objective function value of all three relaxations, and columns 5–7 give, for each formulation, the lower bound on global minimum cost provided by BARON after one CPU-hour of computations. If the optimizer managed to solve the problem within this time limit, the CPU-time spent is given in parentheses. Columns 8–9 provide the lower bound on the global minimum cost from the runs with our own code applied to the STP-formulation, where both `max-infeas` (column 8, labeled STP1) and `min-max-infeas` (column 9, labeled STP2) are used as branching strategies. Within each column group, the best results are given in bold unless all are equal.

We observe that only two instances, A2 and A9, could be solved to optimality. In instance A2, this was accomplished by application of BARON to any of the formulations, and by our code combined with the STP-formulation using branching strategy `min-max-infeas`. Instance A9 could exclusively be solved by BARON in combination with the STP-formulation.

Comparing the optimal objective function values of the initial relaxations, we observe that the STP-formulation strictly dominates *both* the other relaxations in 10 instances. This extends the observation made from the runs on standard instances (Table 2), where the STP-relaxation in all instances gave a bound identical to the best of those given by the PQ- and the TP-relaxations. Hence, integrating source and terminal proportions in the same model provides a strength that cannot be achieved by however clever selection of only one of the sets of proportion variables.

Table 7: Comparison between the PQ-, TP- and STP-formulations using BARON and the algorithm given in Section 6.

Inst.	Relaxation obj. value			BARON lb (time)			Our code lb (time)	
	PQ	TP	STP	PQ	TP	STP	STP1	STP2
A0	-37772.75	-37443.95	**-37412.23**	-37129.03	-36472.50	**-36411.10**	-36850.86	**-36628.30**
A1	-31516.93	-31579.06	**-31438.51**	-30096.46	**-29880.90**	-29896.91	-30080.32	**-29775.79**
A2	-23898.81	**-23743.36**	**-23743.36**	(400.52)	(199.24)	**(83.67)**	-23056.99	**(552.56)**
A3	-42066.64	-42252.01	**-42032.79**	-40524.48	-40482.61	**-40324.91**	-40392.23	**-40155.50**
A4	-43460.83	-43435.15	**-43396.84**	-42788.51	-42694.84	**-42659.52**	-42558.13	**-42374.92**
A5	-28257.75	-28257.75	-28257.75	-28257.75	-28257.75	-28257.75	-28254.53	**-28251.33**
A6	-42463.05	-42463.05	-42463.05	-42463.05	-42463.05	-42463.05	-42462.12	**-42460.51**
A7	-44682.25	-44682.25	-44682.25	-44682.25	-44682.25	-44682.25	-44682.25	-44682.25
A8	-30666.87	-30666.87	-30666.87	-30666.87	-30666.87	-30666.87	-30666.87	-30666.87
A9	-21933.99	-21933.99	-21933.99	-21933.99	-21933.99	**(484.22)**	-21933.99	-21933.99
B0	-45466.54	-45466.54	**-45465.92**	-45327.98	-45230.10	**-45179.93**	-45317.07	**-45260.35**
B1	-65528.17	-65527.57	**-65523.34**	-65242.22	-65104.57	**-65048.03**	-65151.39	**-65110.17**
B2	-56530.06	-56490.77	**-56438.06**	-56125.88	-56232.02	**-56083.32**	**-56145.64**	-56152.94
B3	-74050.47	-74050.47	-74050.47	-74050.47	-74050.47	-74050.47	-74050.47	-74050.47
B4	-59469.66	-59469.66	-59469.66	-59469.66	-59469.66	-59469.66	-59469.66	-59469.66
B5	-60696.36	-60696.36	-60696.36	-60696.36	-60696.36	-60696.36	-60696.36	-60696.36
C0	-99129.09	-99155.38	**-98218.60**	-97459.09	-97543.25	**-96256.99**	-97273.39	**-96745.43**
C1	-119997.45	-119031.69	**-118673.48**	-118252.95	-118495.83	**-117856.16**	-117438.62	**-117185.23**
C2	-136102.85	-136043.63	**-135740.45**	-135831.02	-135913.45	**-135546.50**	-135238.89	**-135228.17**
C3	-130315.02	-130315.02	-130315.02	-130315.02	-130315.02	-130315.02	-130315.02	-130315.02

The strength of the STP-formulation is reflected by better lower bounds after one CPU-hour of computation with BARON. We observe that the STP-formulation gives a better lower bound than both its competitors in 10 instances (A0–A1, A3–A4, B0–B2, and C0–C2), while the TP-formulation is the unique winner in instance A1. In the remaining instances that were not solved to optimality, the three formulations yield identical lower bounds, and the PQ-formulation never performs better than more than one of the other formulations. The relative differences in the lower bounds are however small, with a 1.2% improvement from the PQ- to the STP-formulation in instance C0 as the largest. We also observe that the STP-formulation can be applied to solve instance A2 in 21% and 42%, respectively, of the time required by the PQ- and TP-formulations.

In 8 instances (A5–A8, B3–B5 and C3), BARON was, regardless of the formulation used, unable to improve the lower bound given by the initial LP relaxation. Our code could however increase the lower bound slightly in instances A5–A6.

By comparing the last column (results from our code with the STP-formulation and branching strategy `min-max-infeas`) to the result from BARON applied to the formulation giving the best bound, we see that our code produces better bounds in 7 unsolved instances (A1, A3–A6, C1–C2), BARON wins in 5 unsolved instances (A0, B0–B2, C0), while equal bounds are obtained in the remaining 6 unsolved instances (A7–A8, B3–B5, C3). The differences are however small, and replacement of one optimizer by its alternative yields a relative improvement below 1% in all instances. A comparison between the two last columns, shows that the `min-max-infeas` strategy performs better than the `max-infeas` strategy in 12 instances (A0–A6, B0–B1, C0–C2), and shows weaker performance only in instance B2. Although the margins between the two strategies are small, the `min-max-infeas` seems to add to the efficiency of the algorithm.

The upper bounds, and thereby the optimality gap, produced by the default upper-bounding techniques in BARON also depend on what formulation we apply. Columns 2, 4 and 6 in Table 8 contain the upper bounds ub on the optimal cost upon completion or interruption of BARON. In the columns labeled gap in the same table, we show for each formulation the remaining relative optimality gaps, that is $\left|\frac{ub-lb}{lb}\right|$, where lb refers to the lower bounds computed by BARON and reported in Table 7. Since our code does not include any heuristic for com-

puting feasible solutions, the optimality gaps and upper bounds are consistently weaker than those provided by BARON, and are not included in the table. We observe that in many instances, the qualities of the best feasible solutions found can be quite different for different formulations. The largest differences between the best and second best formulations in this respect are observed in instances C2 (the TP-bound improves the STP-bound by 66%) and B2 (the TP-bound improves the STP-bound by almost 50%). The optimality gaps vary correspondingly, and in 11 unsolved instances, the gap is minimized by applying the STP-formulation. In 7 unsolved instances, the TP-formulation gives the smallest gap, whereas the PQ-formulation wins only in instance B5.

Table 8: Relative optimality gap after one CPU-hour for each of the formulations.

Inst.	PQ		TP		STP	
	ub	gap	ub	gap	ub	gap
A0	-24340.44	0.344	-34715.67	0.048	**-35812.33**	**0.016**
A1	-29207.42	0.030	-29239.98	**0.021**	**-29276.56**	**0.021**
A2	**-23044.16**	0.010	-23042.04	**0.008**	-23042.04	0.010
A3	-30576.87	0.245	-39365.81	0.028	**-39446.54**	**0.022**
A4	-36517.07	0.147	**-38377.23**	**0.101**	-33687.13	0.210
A5	-5168.78	0.817	-18165.06	0.357	**-24015.54**	**0.150**
A6	-4446.61	0.895	-31344.65	0.262	**-37074.67**	**0.127**
A7	-12498.86	0.720	**-40554.40**	**0.092**	-38074.67	0.148
A8	-6506.23	0.788	**-29104.03**	**0.051**	-28795.26	0.061
A9	-21553.97	0.017	-17157.62	0.218	**-21912.35**	**0.001**
B0	-15516.33	0.658	-18646.01	0.588	**-20802.12**	**0.540**
B1	-26518.82	0.594	-36904.00	0.433	**-50055.21**	**0.230**
B2	-4299.77	0.923	-12399.21	0.779	**-18567.64**	**0.669**
B3	-11842.86	0.840	-15708.79	0.788	**-18327.40**	**0.753**
B4	-8955.51	0.849	**-13786.45**	**0.768**	-3711.84	0.938
B5	**-14672.44**	**0.758**	-3858.44	0.936	-10653.72	0.824
C0	-9106.04	0.907	-15158.92	0.845	**-15197.45**	**0.842**
C1	-17492.55	0.852	-16252.68	0.863	**-25196.93**	**0.786**
C2	-7411.96	0.945	**-12476.00**	**0.908**	-7497.52	0.945
C3	-16351.75	0.875	**-20675.44**	**0.841**	-7163.54	0.945

In the hardest instances, the optimality gap remains wide even after one CPU-hour, regardless of what formulation is applied (in instance C1, it is above 90% for

all formulations). We ran BARON for 10 CPU-hours on a selection of instances, but observed only marginal improvement of the lower bounds.

8 Concluding remarks

In this paper, we have proved formally that the pooling problem is strongly $\mathcal{NP}$-hard by constructing a polynomial reduction from the maximum independent vertex set problem. We have also showed that the problem remains strongly $\mathcal{NP}$-hard when there is only one pool and no direct arcs from sources to terminals. Moreover, we have derived new formulations for the problem based on terminal proportion variables. In the strongest model, denoted the STP-formulation, we have combined source and terminal proportion variables. This formulation is proved to be stronger than the popular PQ-formulation based uniquely on source proportions. We have also suggested a new branching strategy that performs well in combination with the STP-formulation.

Computational experiments with 14 instances from the open literature and 20 large-scale randomly generated instances have confirmed the strength of the STP-formulation. To evaluate the effectiveness of the branching strategy for the STP-formulation, we have implemented a branch-and-bound algorithm in C++. Computational experiments with large-scale instances, most of which are too large to be solved to optimality, indicate that the new branching strategy helps to improve the lower bound on the optimal objective function value.

Acknowledgements This research was sponsored by the Norwegian Research Council, Gassco, and Statoil under contract 175967/S30

References

Adhya, N., Tawarmalani, M., and Sahinidis, N.V. (1999). A Lagrangian approach to the pooling problem. *Industrial & Engineering Chemistry Research*, **38** (5), 1956–1972.

Al-Khayyal, F.A., and Falk, J.E. (1983). Jointly constrained biconvex programming. *Mathematics of Operations Research*, **8** (2), 273–286.

Almutairi, H., and Elhedhli, S. (2009). A new Lagrangian approach to the pooling problem. *Journal of Global Optimization*, **45** (2), 237–257.

Audet, C., Hansen, P., Jaumard, B., and Savard, G. (2000). A branch and cut algorithm for nonconvex quadratically constrained quadratic programming. *Mathematical Programming*, **87** (1), 131–152.

Audet, C., Brimberg, J., Hansen, P., Le Digabel, S., and Mladenović, N. (2004). Pooling problem: Alternate formulations and solution methods. *Management science*, **50** (6), 761–776.

Baker, T.E., and Lasdon, L.S. (1985). Successive linear programming at Exxon. *Management Science*, **31** (3), 264–274.

Ben-Tal, A., Eiger, G., and Gershovitz, V. (1994). Global minimization by reducing the duality gap. *Mathematical Programming*, **63** (2), 193–212.

Floudas, C.A., and Aggarwal, A. (1990). A decomposition strategy for global optimization search in the pooling problem. *Operations Research Journal On Computing*, **2** (3), 225–235.

Floudas, C.A., and Visweswaran, V. (1990). A global optimization algorithm (GOP) for certain classes of nonconvex NLPs–I. Theory. *Computers & chemical engineering*, **14** (12), 1397–1417.

Foulds, L.R., Haugland, D., and Jörnsten, K. (1992). A bilinear approach to the pooling problem. *Optimization*, **24** (1), 165–180.

Garey, M.R., and Johnson, D.S. (1979). *Computers and Intractability: A Guide to the Theory of NP-completeness*. New York, USA: W.H. Freeman & Co.

Gounaris, C.E., Misener, R., and Floudas, C.A. (2009). Computational comparison of piecewise–linear relaxation for pooling problems. *Industrial & Engineering Chemistry Research*, **48** (12), 5742–5766.

Griffith, R.E., and Stewart, R.A. (1961). A nonlinear programming technique for the optimization of continuous processing systems. *Management Science*, **7**, 379–392.

Haverly, C.A. (1978). Studies of the behavior of recursion for the pooling problem. *ACM SIGMAP Bulletin*, **25**, 19–28.

Haverly, C.A. (1979). Behavior of recursion model-more studies. *ACM SIGMAP Bulletin*, **26**, 22–28.

Karuppiah, R., and Grossmann, I.E. (2006). Global optimization for the synthesis of integrated water systems in chemical processes. *Journal of Computers & Chemical Engineering*, **30** (4), 650–673.

Liberti, L., and Pantelides, C.C. (2006). An exact reformulation algorithm for large nonconvex NLPs involving bilinear terms. *Journal of Global Optimization*, **36** (2), 161–189.

McCormick, G.P. (1976). Computability of global solutions to factorable nonconvex programs: part I - convex underestimating problems. *Mathematical Programming*, **10** (1), 147–175.

Meyer, C.A., and Floudas, C.A. (2006). Global optimization of a combinatorially complex generalized pooling problem. *AIChE Journal*, **52** (3), 1027–1037.

Misener, R., Gounaris, C.E., and Floudas, C.A. (2010). Mathematical modeling and global optimization of large-scale extended pooling problems with the (EPA) complex emissions constraints. *Journal of Computers & Chemical Engineering*, **34**, 1432–1456.

Palacios-Gomez, F., Lasdon, L.S., and Engquist, M. (1982). Nonlinear optimization by successive linear programming. *Management Science*, **28** (10), 1106–1120.

Quesada, I., and Grossmann, I.E. (1995). Global optimization of bilinear process networks with multi-component flows. *Computers & Chemical Engineering*, **19** (12), 1219–1242.

Ryoo, H.S., and Sahinidis, N.V. (1995). Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computers & Chemical Engineering*, **19** (5), 551–566.

Sahinidis, N.V. (1996). BARON: A general purpose global optimization software package. *Journal of Global Optimization*, **8** (2), 201–205.

Sahinidis, N.V., and Tawarmalani, M. (2005). Accelerating branch-and-bound through a modeling language construct for relaxation-specific constraints. *Journal of Global Optimization*, **32** (2), 259–280.

Sarker, R.A., and Gunn, E.A. (1997). A simple SLP algorithm for solving a class of nonlinear programs. *European Journal of Operational Research*, **101** (1), 140–154.

Shectman, J.P., and Sahinidis, N.V. (1998). A finite algorithm for global minimization of separable concave programs. *Journal of Global Optimization*, **12** (1), 1–36.

Sherali, H.D., and Adams, W.P. (1999). *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Sherali, H.D., Adams, W.P., and Driscoll, P.J. (1998). Exploiting special structures in constructing a hierarchy of relaxations for 0-1 mixed integer problems. *Operations Research*, **46** (3), 396–405.

Visweswaran, V., and Floudas, C.A. (1990). A global optimization algorithm (GOP) for certain classes of nonconvex NLPs–II. Application of theory and test problems. *Computers & chemical engineering*, **14** (12), 1419–1434.

Zhang, J.H., Kim, N.H., and Lasdon, L. (1985). An improved successive linear programming algorithm. *Management Science*, **31** (10), 1312–1331.

Paper B

B

A multi-commodity flow formulation for the generalized pooling problem[2]

Mohammed Alfaki[1] and Dag Haugland[1]

[1]Department of Informatics, University of Bergen,
P.O. Box 7803, N-5020 Bergen, Norway.
mohammeda@ii.uib.no and dag@ii.uib.no

[2]In: *Journal of Global Optimization*, doi: 10.1007/s10898-012-9890-7, 2012.

A multi-commodity flow formulation for the generalized pooling problem

Mohammed Alfaki[1] and Dag Haugland[1]

1 Department of Informatics, University of Bergen,
P.O. Box 7803, N-5020 Bergen, Norway.
mohammeda@ii.uib.no and dag@ii.uib.no

Abstract

The pooling problem is an extension of the minimum cost network flow problem where the composition of the flow depends on the sources from which it originates. At each source, the composition is known. In all other nodes, the proportion of any component is given as a weighted average of its proportions in entering flow streams. The weights in this average are simply the arc flow. At the terminals of the network, there are bounds on the relative content of the various components. Such problems have strong relevance in e.g. planning models for oil refining, and in gas transportation models with quality constraints at the reception side.

Although the pooling problem has bilinear constraints, much progress in solving a class of instances to global optimality has recently been made. Most of the approaches are however restricted to networks where all directed paths have length at most three, which means that there is no connection between pools. In this work, we generalize one of the most successful formulations of the pooling problem, and propose a multi-commodity flow formulation that makes no assumptions on the network topology. We prove that our formulation has stronger linear relaxation than previously suggested formulations, and demonstrate experimentally that it enables faster computation of the global optimum.

Keywords Pooling problem · Multi-commodity flow · Linear relaxation · Bilinear constraint · Convex and concave envelopes

1 Motivation

The classic blending problem, which appears in many industrial settings, involves determination of the optimal blend of raw materials to produce a certain quantity of end products. The composition of the end products is subject to certain specifications, and the decision maker must determine in what proportions the raw materials should be used in the various end products. The optimal blend means the minimum cost blend of inputs satisfying the given specifications. Such problems can be formulated as linear programs (LP), and are therefore very easy to solve.

A considerably more difficult problem occurs if the blended flow of raw materials is mixed in intermediate tanks (pools) and then blended again to form end products. The nodes are partitioned into sources, pools and terminals, and each arc goes either from a source to a pool, from a source to a terminal, or from a pool to a terminal. The corresponding flow problem in networks with this particular tripartite structure is referred to as the *pooling problem*. Tracking the quality from the sources to the terminals leads to bilinear constraints, and consequently the pooling problem can possibly have many local optima. Recently, it was shown (Alfaki and Haugland, 2012) that the pooling problem is strongly $\mathcal{NP}$-hard.

Most mathematical programming formulations for the pooling problem belong to either of two main classes. Belonging to the first class are formulations that, in addition to flow variables, make use of variables representing the quality of the flow. Noteworthy among these is the P-formulation, which was first introduced by Haverly (1978, 1979). In contrast, Ben-Tal et al. (1994) suggested the Q-formulation, where the quality variables are replaced by variables representing flow proportions. This model has later been shown to perform better when fed into generic branch-and-cut algorithms. By applying the reformulation-linearization technique (Sherali, 2002; Sherali and Adams, 1999; Sherali et al., 1998) to the Q-formulation, Tawarmalani and Sahinidis (2002) managed to derive a stronger model referred to as the PQ-formulation. The superiority in strength over the P-formulation was proved theoretically, and experiments with the global optimizer BARON (Sahinidis, 1996) showed that standard test instances could be

solved very quickly. Building on this work, Alfaki and Haugland (2012) managed to improve the PQ-formulation even further.

Practical applications, particularly in integrated water systems in chemical processes (Karuppiah and Grossmann, 2006) and pipeline transportation of natural gas (Li et al., 2011), frequently reveal that flow streams leaving one pool may in their turn be blended in pools further downstream in the network. Single-period planning models based on networks with a single layer of pools may also have this property when extended to multi-period models with inventories. In such models, each pair of inventory and time period becomes a pool, and constraints that define the pool qualities are necessary if the input quality cannot be assumed constant over time. Contradicting the assumptions made in the works cited in the paragraph above, the applications in question imply a flow network with paths intersecting more than one pool.

The optimization problem arising when no longer assuming the tripartite network structure is referred to (Audet et al., 2004; Misener and Floudas, 2009) as the *generalized pooling problem* (GPP). For the purpose of a clear distinction, the pooling problem is henceforth in the current work also referred to as the *standard* pooling problem.

According to Main (1993), paths of many pools tend to reduce the chance for local optimization methods to end up in the global optimum. This can be overcome by extending the P-formulation to GPP, and apply a global solver handling bilinear constraints to the resulting model. Such an extension is straightforward. Given the theoretical and computational superiority of the PQ-formulation for the standard pooling problem, it is however reasonable to expect that an extension of the PQ-formulation will be competitive with an extended P-formulation. This motivates the goal of the current work: Derive a computationally efficient model for GPP by extending the idea of the PQ-formulation to networks with arbitrary structure.

Some attempts to generalize pooling problem formulations based on proportion variables can be found in the literature. Audet et al. (2004) indicate a hybrid model, with both quality and proportion variables. Following the idea of Ben-Tal et al. (1994), they introduce proportion variables at pools that receive flow only

from sources, and apply quality variables in the spirit of the P-formulation for other pools.

Recently, generalizations of the pooling problem that make no assumptions on the network structure have been developed. Meyer and Floudas (2006) and Misener et al. (2010) proposed a mixed integer nonlinear (MINLP) formulation for the design of waste-water treatment. The networks may contain arcs between pools, since the flow of water may go through several stages of refinement (reduction of contamination). Their model is based on quality variables with constraints generalizing those of the P-formulation. This is a very natural choice of variables, since it does not seem appropriate to model quality updates in terms of proportion variables. Because the authors also consider a fixed charge for opening the arcs, binary variables are introduced, and the model becomes an MINLP. To tighten the relaxation, Misener et al. (2010) also apply a novel piecewise underestimation for the nonconvex bilinear terms (Gounaris et al., 2009; Wicaksono and Karimi, 2008). Another extension of the standard pooling problem has been proposed by Misener and Floudas (2010) to maximize the profit of blending reformulated gasoline subject to environmental standards, involving complex emission constraints. Other interesting applications of global optimization related to network flow problems are given in Chapter 6 in the textbook of Horst et al. (2000) and by Guisewite (1995).

The contribution from this paper is a new multi-commodity flow formulation for the GPP. Complying with the PQ-formulation, it is based uniquely on proportion variables in addition to the flow variables. In network instances where all paths intersect at most one pool, our formulation coincides with the PQ-formulation, and we demonstrate that the favorable theoretical and computational properties carry over to GPP.

The paper is organized as follows: In Section 2, we give some notations and definitions that will be used throughout the paper. Section 3 presents the P-formulation applied to GPP, and in Section 4, we propose and analyze the multi-commodity flow formulation. The analysis includes a comparison with the P-formulation and the hybrid model of Audet et al. (2004), both of which turn out to have weaker relaxations than our model. Section 5 presents computational

experiments where the said formulations are compared, and Section 6 concludes the paper.

2 Notations and definitions

We consider a directed graph $G = (N, A)$ with node set N and arc set A. For any node $i \in N$, let $N_i^+ = \{j \in N : (i,j) \in A\}$ and $N_i^- = \{j \in N : (j,i) \in A\}$ denote the set of out- and in-neighbors of i, respectively. We assume that G has non-empty sets $S, T \subseteq N$ of *sources* and *terminals*, respectively, where $N_s^- = \emptyset$ $\forall s \in S$ and $N_t^+ = \emptyset$ $\forall t \in T$. We refer to all nodes in $I = N \setminus (S \cup T)$ as *pools*. We define a finite set of *quality attributes* K. With each $i \in S \cup T$, we associate a real constant q_i^k for each $k \in K$. If $s \in S$, q_s^k is referred to as the *quality parameter* of attribute k at that source, and if $t \in T$, q_t^k is referred to as the *quality bound* of attribute k at terminal t. For each $i \in N$, we define the constant *flow capacity* b_i, and for each arc $(i,j) \in A$, we define the constant *unit cost* c_{ij}. This is slightly more general than defining costs and revenues only at the sources and the terminals, respectively, which is common practice in the pooling problem literature. For each $i \in N$, let S_i be the set of sources from which there exists a path to i in G (for all $s \in S$, we have $S_s = \{s\}$).

Define the *flow polytope* $\mathcal{F}(G, b)$ as the set of flow vectors $f \in \mathbb{R}_+^A$ satisfying

$$\sum_{j \in N_i^+} f_{ij} \leq b_i, \qquad i \in N \setminus T, \tag{1}$$

$$\sum_{j \in N_t^-} f_{jt} \leq b_t, \qquad t \in T, \tag{2}$$

$$\sum_{j \in N_i^+} f_{ij} - \sum_{j \in N_i^-} f_{ji} = 0, \qquad i \in I. \tag{3}$$

Consider any $f \in \mathcal{F}(G, b)$. For all $i \in N$ and $k \in K$, define w_i^k as a product quality of attribute k at node i corresponding to flow f. That is, $w_i^k = q_i^k$ if $i \in S$, and for all $i \in N \setminus S$, we have

$$w_i^k \sum_{j \in N_i^-} f_{ji} = \sum_{j \in N_i^-} w_j^k f_{ji},$$

for all $k \in K$.

This means that the product qualities are arbitrary at pools and terminals with zero entering flow, and uniquely defined at all other nodes. The definition implies that we assume linear blending at pools and terminals, and that the flow on all arcs leaving node i has the unique quality w_i^k.

In this work, we consider the following extension of the minimum cost flow problem:

Problem 1 (The generalized pooling problem)**.** *Find $f \in \mathcal{F}(G, b)$ and a corresponding matrix of product qualities $w \in \mathbb{R}^{N \times K}$ satisfying $w_t^k \leq q_t^k \ \forall t \in T,\ k \in K$, such that $\sum_{(i,j)\in A} c_{ij} f_{ij}$ is minimized.*

Without loss of generality, we have assumed that only upper quality bounds are imposed. Should a lower bound $w_t^k \geq \ell_t^k$ apply, we introduce a new quality attribute k^-, and define $q_s^{k^-} = -q_s^k$ for all $s \in S$, and let $q_t^{k^-} = -\ell_t^k$.

It follows directly from the strong $\mathcal{NP}$-hardness of the standard pooling problem (Alfaki and Haugland, 2012) that also Problem 1 is strongly $\mathcal{NP}$-hard. The proof of this fact relies on a problem definition without lower bounds on the delivery to the terminals, or at least includes zero bounds as a possibility. For the purpose of a simple model, we have assumed the lower delivery bounds to be zero, but the main results in this work do not depend on this assumption.

3 A formulation based on quality variables

3.1 The P-formulation for the generalized pooling problem

The P-formulation of the problem is found by extending the minimum cost flow problem (with node capacities) in directed graphs. Its decision variables are exactly those indicated in the problem definition, namely the flow f_{ij} along arc (i, j), and w_i^k representing the quality in terms of attribute k of the flow leaving node $i \in N$. In addition, we introduce the decision variable v_{ij}^k (for all $(i, j) \in A$, $k \in K$), which is best understood by thinking of the attribute indexed by k as a contaminant. While the quality w_i^k is the relative content of the contaminant in the flow along arc (i, j), v_{ij}^k is the total amount of the contaminant in the same flow. Thus, $v_{ij}^k = w_i^k f_{ij}$ is a necessary relation.

For notational convenience, define the constants $w_s^k = q_s^k$ for all $s \in S$, $k \in K$. Problem 1 is then formulated as (recall that $N_s^- = \emptyset \; \forall s \in S$ and $N_t^+ = \emptyset \; \forall t \in T$):

$$[\text{P}] \qquad \min_{f,w,v} \sum_{(i,j)\in A} c_{ij} f_{ij} \tag{4}$$

$$\text{s.t.} \qquad f \in \mathcal{F}(G, b), \tag{5}$$

$$\sum_{j\in N_i^-} v_{ji}^k - \sum_{j\in N_i^+} v_{ij}^k = 0, \qquad i \in I,\ k \in K, \tag{6}$$

$$\sum_{j\in N_t^-} v_{jt}^k - q_t^k \sum_{j\in N_t^-} f_{jt} \le 0, \qquad t \in T,\ k \in K, \tag{7}$$

$$v_{ij}^k - w_i^k f_{ij} = 0, \qquad (i,j) \in A,\ k \in K. \tag{8}$$

Constraints (6) state that the amounts of contaminant k entering and leaving pool i are equal, and constraints (7) say that if any flow enters terminal t, then the relative content of k resulting from blending, $\frac{\sum_{j\in N_t^-} v_{jt}^k}{\sum_{j\in N_t^-} f_{jt}}$, must be below q_t^k.

The bilinear program (4)–(8) is referred to as the P-formulation for Problem 1. Alternative variants of the P-formulation can be derived, e.g. by substituting some or all occurrences of v_{ij}^k by $w_i^k f_{ij}$. We have introduced the v-variables for the sole purpose of collecting all bilinear terms in one set of constraints (8).

3.2 Linear relaxations of bilinear terms

For all formulations considered in this work, we apply a well established technique for relaxing bilinear constraints. It is illustrated here in the case of the P-formulation.

For all $k \in K$ and $(i,j) \in A$, let $\left[\underline{w}_i^k, \overline{w}_i^k\right]$ and $\left[\underline{f}_{ij}, \overline{f}_{ij}\right]$ be some intervals enclosing the feasible values of variables w_i^k and f_{ij}, respectively. Conservative, but quickly computed bounds are given as e.g. $\min_{s\in S_i} q_s^k \le w_i^k \le \max_{s\in S_i} q_s^k$ and $0 \le f_{ij} \le \min\{b_i, b_j\}$.

It is well known (Al-Khayyal and Falk, 1983; McCormick, 1976) that since (8) is bilinear, v_{ij}^k can be bounded between the convex and concave envelopes

of $w_i^k f_{ij}$ on $\left[\underline{w}_i^k, \overline{w}_i^k\right] \times \left[\underline{f}_{ij}, \overline{f}_{ij}\right]$ by imposing four linear inequalities, henceforth referred to as the McCormick inequalities:

$$v_{ij}^k \geq \underline{w}_i^k f_{ij} + \underline{f}_{ij} w_i^k - \underline{w}_i^k \underline{f}_{ij}, \tag{9}$$
$$v_{ij}^k \geq \overline{w}_i^k f_{ij} + \overline{f}_{ij} w_i^k - \overline{w}_i^k \overline{f}_{ij}, \tag{10}$$
$$v_{ij}^k \leq \underline{w}_i^k f_{ij} + \overline{f}_{ij} w_i^k - \underline{w}_i^k \overline{f}_{ij}, \tag{11}$$
$$v_{ij}^k \leq \overline{w}_i^k f_{ij} + \underline{f}_{ij} w_i^k - \overline{w}_i^k \underline{f}_{ij}. \tag{12}$$

A linear relaxation of the P-formulation is thus obtained by replacing (8) by (9)–(12). The tightness of the relaxation obviously depends on the tightness of the variable bounds.

More generally, let F be any bilinear formulation for Problem 1 and C a rectangular subset defined in the space of the bilinear variables such that it contains all their feasible values. We denote by $LP^F[C]$ the linear program obtained by replacing each bilinear term by a new variable constrained by the McCormick inequalities corresponding to C. Let $z^F[C]$ denote the optimal objective function value of $LP^F[C]$.

4 A multi-commodity flow formulation based on proportion variables

Tawarmalani and Sahinidis (2002) suggested the so-called PQ-formulation for the standard pooling problem, and proved that it is stronger than the P-formulation. In this section, we generalize these results to GPP.

4.1 The PQ-formulation for the standard pooling problem

Assume now that $A \subseteq (S \times I) \cup (I \times T) \cup (S \times T)$. Define the proportion variables y_i^s ($s \in S$, $i \in I$) as the fraction of the flow through pool i that originates from

source s. Combining the new variables with flow variables, the PQ-formulation for the standard pooling problem can in our notation be written as

$$\text{[PQ]} \quad \min_{f,y} \sum_{s\in S}\sum_{i\in I\cap N_s^+} c_{si}y_i^s \sum_{t\in N_i^+} f_{it} + \sum_{t\in T}\sum_{j\in N_t^-} c_{jt}f_{jt} \tag{13}$$

$$\text{s.t.} \quad \sum_{i\in I\cap N_s^+} y_i^s \sum_{t\in N_i^+} f_{it} + \sum_{t\in T\cap N_s^+} f_{st} \le b_s, \qquad s\in S, \tag{14}$$

$$\sum_{t\in N_i^+} f_{it} \le b_i, \qquad i\in I, \tag{15}$$

$$\sum_{j\in N_t^-} f_{jt} \le b_t, \qquad t\in T, \tag{16}$$

$$\sum_{i\in I\cap N_t^-} f_{it}\sum_{s\in S_i} q_s^k y_i^s + \sum_{s\in S\cap N_t^-} q_s^k f_{st} - q_t^k \sum_{j\in N_t^-} f_{jt} \le 0, \qquad t\in T, k\in K, \tag{17}$$

$$\sum_{s\in S_i} y_i^s = 1, \qquad i\in I, \tag{18}$$

$$f_{it} - \sum_{s\in S_i} y_i^s f_{it} = 0, \qquad i\in I,\ t\in N_i^+, \tag{19}$$

$$\sum_{t\in N_i^+} y_i^s f_{it} - b_i y_i^s \le 0, \qquad i\in I,\ s\in S_i, \tag{20}$$

$$f_{jt} \ge 0, \qquad t\in T,\ j\in N_t^-,$$

$$0 \le y_i^s \le 1, \qquad i\in I,\ s\in S_i.$$

The model makes use of flow variables f_{it} only along arcs entering terminals. In contrast, the flow along arc $(s,i)\in A\cap(S\times I)$ is represented by $y_i^s\sum_{t\in N_i^+} f_{it}$. Multiplying y_i^s by the total flow through i gives the flow along (s,i), justifying the suggested flow representation.

Consequently, the first sum in the objective function covers cost of flow from sources to pools. Constraints (14)–(16) represent flow capacities at sources, pools and terminals, respectively, and (17) gives the quality constraint at the terminals. Constraint (18) follows directly from the definition of the proportion variables. Multiplying (18) by f_{it} yields (19), which hence is redundant. Similarly, (20) is obtained by multiplying (15) by y_i^s. Sahinidis and Tawarmalani (2005) have

shown that adding the redundant inequalities largely improves the linear relaxation and accelerates search algorithms.

4.2 The MCF-formulation for general network instances

In order to generalize the PQ-formulation to arbitrary networks, we apply the set of variables defined for the multi-commodity network flow problem. We associate a flow commodity with each source $s \in S$, where at most b_s units of the commodity can enter the network. The commodity can leave the network at any $t \in T$, whereas at all other nodes, the commodity neither enters nor leaves the network. Since there is a bijection between the sets of commodities and sources, we will whenever appropriate refer to the commodity that enters at source s as commodity s. Hence, for all $(i,j) \in A$ and all $s \in S$, we let variable x_{ij}^s denote the flow of commodity s along arc (i,j).

We keep the variable f_{ij} denoting total flow of all commodities along arc $(i,j) \in A$, and let the variable y_i^s denote the proportion of the total flow leaving node $i \in S \cup I$ constituted by commodity s (define the constants $y_i^s = 0$ for $s \notin S_i$ and $y_s^s = 1$). To make the f-, y-, and x-variables consistent, we impose $x_{ij}^s = y_i^s f_{ij}$.

This results in the following formulation of the generalized pooling problem.

$$\text{[MCF]} \qquad \min_{f,y,x} \sum_{(i,j)\in A} c_{ij} f_{ij} \tag{21}$$

$$\text{s.t.} \qquad (1)\text{–}(2),$$

$$\sum_{j\in N_i^-} x_{ji}^s - \sum_{j\in N_i^+} x_{ij}^s = 0, \qquad i \in I,\ s \in S_i, \tag{22}$$

$$\sum_{j\in N_t^-} \sum_{s\in S_j} \left(q_s^k - q_t^k\right) x_{jt}^s \le 0, \qquad t \in T,\ k \in K, \tag{23}$$

$$\sum_{s\in S_i} y_i^s = 1, \qquad i \in I, \tag{24}$$

$$\sum_{s\in S_i} x_{ij}^s - f_{ij} = 0, \qquad (i,j) \in A,\ i \in I, \tag{25}$$

$$\sum_{j\in N_i^+} x_{ij}^s - y_i^s b_i \le 0, \qquad i \in I,\ s \in S_i, \tag{26}$$

$$x_{ij}^s - y_i^s f_{ij} = 0, \qquad (i,j) \in A,\ s \in S_i, \tag{27}$$

$$f_{ij} \geq 0, \qquad (i,j) \in A, \tag{28}$$

$$0 \leq y_i^s \leq 1, \qquad i \in I,\ s \in S_i. \tag{29}$$

This is recognized as the multi-commodity minimum cost flow problem with the additional constraints (23) on the quality at the terminals, and the constraints (27) imposing the flow proportions y_i^s on all arcs with start node i. Bilinear terms occur exclusively in (27). Analogous to (19)–(20), constraints (25)–(26) are redundant, but are added for the same reason as (19)–(20) were added to the PQ-formulation.

We define two bilinear formulations to be *equivalent* if they have identical sets of bilinear terms, and, for all hyper-rectangles C, $z^{F_1}[C] = z^{F_2}[C]$. In the sense of this definition, the following result shows that the MCF-formulation generalizes the PQ-formulation.

Proposition 2. *If $A \subseteq (S \times I) \cup (I \times T) \cup (S \times T)$, then the PQ- and MCF-formulations are equivalent.*

Proof. Because of the assumed network structure, (27) introduces a bilinear term $y_i^s f_{it}$ if and only if $(s,i,t) \in S \times I \times T$ such that $(s,i),(i,t) \in A$. It is easily verified that these are exactly the bilinear terms occurring in (13), (14), (17), (19) and (20). Hence, both formulations associate a bilinear term $y_i^s f_{it}$ with each path (s,i,t) in G.

The relaxations $LP^{PQ}[C]$ and $LP^{MCF}[C]$ are now obtained by replacing all occurrences of $y_i^s f_{it}$ by some new variable $\hat{x}_{it}^s$, and by adding the corresponding McCormick inequalities. Hence, (27) becomes $\hat{x}_{it}^s = x_{it}^s$, and because of (25), we can substitute f_{it} by $\sum_{s \in S_i} \hat{x}_{it}^s$ everywhere in $LP^{MCF}[C]$ except from the McCormick inequalities. Likewise, (19) is transformed to $f_{it} = \sum_{s \in S_i} \hat{x}_{it}^s$, which allows the substitution of f_{it} everywhere but the McCormick inequalities in $LP^{PQ}[C]$. We observe that the two relaxations this way become identical, and $z^{PQ}[C] = z^{MCF}[C]$ follows. □

4.3 Strength of the MCF-formulation

In the remainder of the paper, we turn the attention to the general version of Problem 1, i.e. the assumption made in Proposition 2 will no longer be considered.

When comparing the strength of the P- and MCF-formulations, we face the challenge that the two formulations share only the f-variables. The assumptions about the bounds on respectively the w-variables (in [P]) and the y-variables (in [MCF]) must be consistent. To this end, we apply the bounds $\underline{w}_i^k = \min_{s \in S_i} q_s^k \leq w_i^k \leq \overline{w}_i^k = \max_{s \in S_i} q_s^k$ $(i \in I, k \in K)$, and $0 \leq y_i^s \leq 1$ $(i \in I, s \in S_i)$, respectively. The bounds on the flow variables are identical in the formulations: $\underline{f}_{ij} \leq f_{ij} \leq \overline{f}_{ij}$ $((i,j) \in A)$. We denote the corresponding hyper-rectangles C_P and C_{MCF}, respectively.

Proposition 3 below shows that the MCF-formulation is stronger than the P-formulation also in GPP. The result is a corollary of Proposition 9.1 in the book by Tawarmalani and Sahinidis (2002).

Proposition 3. $z^P[C_P] \leq z^{MCF}[C_{MCF}]$.

Proof. Assume (f, y, x) is a feasible solution to $LP^{MCF}[C_{MCF}]$. Following the idea of the proof of Proposition 9.1 by Tawarmalani and Sahinidis (2002), we show that letting $w_i^k = \sum_{s \in S_i} q_s^k y_i^s$ for all $i \in N$, $k \in K$, and $v_{ij}^k = \sum_{s \in S_i} q_s^k x_{ij}^s$ for all $(i,j) \in A$, $k \in K$, implies that (f, w, v) is feasible in $LP^P[C_P]$.

Obviously, (1)–(2) and $f \geq 0$ are satisfied. Further, (3) follows from (22) and (25), (6) follows by summing (22) multiplied by q_s^k over all $s \in S_i$, and (7) follows from (23) and (25). To show that the McCormick inequalities (9)–(12) hold, we first observe that $x_{ij}^s \geq \underline{f}_{ij} y_i^s + f_{ij} \underline{y}_i^s - \underline{f}_{ij} \underline{y}_i^s \geq \underline{f}_{ij} y_i^s$. Since $\underline{w}_i^k \leq q_s^k$ for all $s \in S_i$ and $k \in K$, and $\sum_{s \in S_i} y_i^s = 1$, we have

$$\begin{aligned}
\underline{w}_i^k f_{ij} + w_i^k \underline{f}_{ij} - \underline{w}_i^k \underline{f}_{ij} &= \underline{w}_i^k \left(f_{ij} - \underline{f}_{ij} \right) + w_i^k \underline{f}_{ij} \\
&= \underline{w}_i^k \sum_{s \in S_i} \left(x_{ij}^s - y_i^s \underline{f}_{ij} \right) + \underline{f}_{ij} \sum_{s \in S_i} q_s^k y_i^s \leq \\
&\sum_{s \in S_i} q_s^k \left(x_{ij}^s - y_i^s \underline{f}_{ij} \right) + \underline{f}_{ij} \sum_{s \in S_i} q_s^k y_i^s = v_{ij}^k.
\end{aligned}$$

In an analogous manner, (10) follows from $x_{ij}^s \leq \overline{f}_{ij} y_i^s$, $\overline{w}_i^k \geq q_s^k$, and (24),

(11) follows from $x^s_{ij} \leq \overline{f}_{ij} y^s_i$, $\underline{w}^k_i \leq q^k_s$, and (24), and finally, (12) follows from $x^s_{ij} \geq \underline{f}_{ij} y^s_i$, $\overline{w}^k_i \geq q^k_s$, and (24).

The proof is complete by observing that the objective functions (4) and (21) are identical. □

4.4 A hybrid model

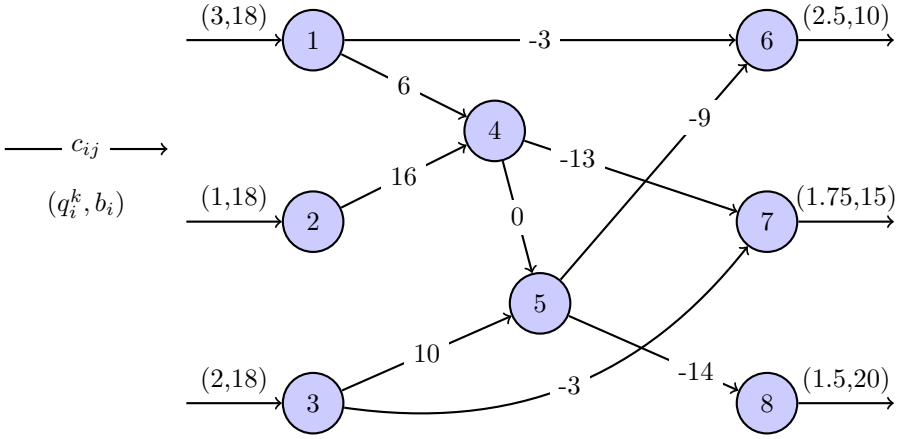
Audet et al. (2004) suggested a model combining quality and proportion variables. In order to avoid terms where proportion variables are squared, which will be the result of a straightforward generalization of the PQ-formulation, they introduce proportion variables for pools that only have sources as in-neighbors, and quality variables for the remaining pools. Denote the former subset of pools $I_1 = \{i \in I : N^-_i = S_i\}$.

The authors formulate their model in terms of the instance depicted in Figure 1, where both pools (nodes 4 and 5) have capacity 20. In agreement with the PQ-formulation, there is no explicit flow variable for arcs linking S to I_1, which in the given instance amounts to arcs (1,4) and (2,4), since respectively $y^1_4(f_{45}+f_{47})$ and $y^2_4(f_{45}+f_{47})$ represent flow along these arcs. For the purpose of keeping the number of bilinear terms low, Audet et al. (2004) also exploit the constraints (18) at node 4 in order to substitute one proportion variable. In our notation, without the suggested variable substitution, their formulation reads:

$$
\begin{aligned}
\min_{f,w,y} \quad & -3f_{16} + 6y^1_4(f_{45}+f_{47}) \\
& +16y^2_4(f_{45}+f_{47}) - 3f_{37} \\
& +10f_{35} - 9f_{56} - 13f_{47} - 14f_{58} + 0f_{45}, \\
\text{supply at 1:} \quad & f_{16} + y^1_4(f_{45}+f_{47}) \leq 18, \\
\text{supply at 2:} \quad & y^2_4(f_{45}+f_{47}) \leq 18, \\
\text{supply at 3:} \quad & f_{35} + f_{37} \leq 18, \\
\text{capacity at 4:} \quad & f_{45} + f_{47} \leq 20, \\
\text{capacity at 5:} \quad & f_{56} + f_{58} \leq 20, \\
\text{demand at 6:} \quad & f_{16} + f_{56} \leq 10, \\
\text{demand at 7:} \quad & f_{37} + f_{47} \leq 15,
\end{aligned}
$$

$$
\begin{aligned}
\text{demand at 8:} \quad & f_{58} \leq 20, \\
\text{quality balance at 5:} \quad & 2f_{35} + (3y_4^1 + y_4^2)f_{45} = w_5^1(f_{56} + f_{58}) \\
\text{quality bound at 6:} \quad & 3f_{16} + w_5^1 f_{56} \leq 2.5(f_{16} + f_{56}), \\
\text{quality bound at 7:} \quad & 2f_{37} + (3y_4^1 + y_4^2)f_{47} \leq 1.75(f_{37} + f_{47}), \\
\text{quality bound at 8:} \quad & w_5^1 \leq 1.5, \\
\text{sum of proportions at 4:} \quad & y_4^1 + y_4^2 = 1, \\
& y_4^1, y_4^2, f_{16}, f_{35}, f_{37}, f_{45}, f_{47}, f_{56}, f_{58} \geq 0.
\end{aligned}
$$

Figure 1: Instance studied by Audet et al. (2004).



Unfortunately, the constraint representing the quality bound at terminal 8 is too strict, and renders the formulation slightly incorrect. Although terminal 8 has pool 5 as its unique in-neighbor, it is not correct to transfer the quality bound to the pool. By doing so, all feasible solutions with zero flow along arc (5,8) and $w_5^1 > 1.5$ are incorrectly excluded. The error is corrected by replacing $w_5^1 \leq 1.5$ by $w_5^1 f_{58} \leq 1.5 f_{58}$. Such complementarity constraints illustrate the binary nature of arcs linking terminals with only one neighbor: Either the arc must be closed or the quality at the pool must satisfy the bound at the terminal.

With the given data, it is however optimal to assign positive flow to arc (5,8), and the suggested formulation finds the optimal flow pattern $f_{16} = 20/3$, $f_{35} = 84/11$, $f_{37} = 114/11$, $f_{45} = 136/11$, $f_{47} = 51/11$, $f_{56} = 10/3$, $f_{58} = 50/3$,

$y_4^1 = 13/136$, $y_4^2 = 123/136$, and $w_5^1 = 3/2$ (all other variables are zero at optimum). The corresponding cost is $-5621/132 \approx -42.58$, whereas Audet et al. (2004) incorrectly report the minimum cost to be -60.5.

The flaw (quality bound at 8) in the formulation of Audet et al. (2004) becomes active after a slight modification of the input data. If the supplies at sources 1 and 3 are increased to 30, the demands at terminals 6 and 7 are increased to 50 and 45, respectively, and the costs at arcs (1,6) and (5,6) are decreased to -6 and -12, respectively, we have $w_5^1 = 7/4$ and $f_{58} = 0$ in the optimal solution. The minimum cost is -220, whereas the formulation suggested by Audet et al. (2004) concludes with a solution of cost -210.

Below, we formulate the hybrid model in more general terms. For each pool which in the sense defined above is close to the sources, the hybrid model makes use of a proportion variable y_i^s for each neighboring source s. For other pools, a quality variable w_i^k for each attribute $k \in K$ is used. We repeat the principle of isolating the bilinear terms in dedicated constraints, which requires the introduction of variables $v_{ij}^k = w_i^k f_{ij}$ for all $k \in K$ and arcs (i,j) where $i \in I \setminus I_1$, and $x_{ij}^s = y_i^s f_{ij}$ for all $s \in S_i$ and arcs (i,j) where $i \in I_1$.

$$[\text{HYB}] \quad \min_{f,y,w,x,v} \sum_{s\in S}\left(\sum_{j\in N_s^+\setminus I_1} c_{sj}f_{sj} + \sum_{i\in N_s^+\cap I_1}\sum_{j\in N_i^+} c_{si}x_{ij}^s\right) + \sum_{i\in I}\sum_{j\in N_i^+} c_{ij}f_{ij} \tag{30}$$

$$\text{s.t.} \quad \sum_{j\in N_s^+\setminus I_1} f_{sj} + \sum_{i\in N_s^+\cap I_1}\sum_{j\in N_i^+} x_{ij}^s \le b_s, \quad s \in S, \tag{31}$$

$$\sum_{j\in N_i^+} f_{ij} \le b_i, \quad i \in I, \tag{32}$$

$$\sum_{i\in N_t^-} f_{it} \le b_t, \quad t \in T, \tag{33}$$

$$\sum_{j\in N_i^+} f_{ij} - \sum_{j\in N_i^-} f_{ji} = 0, \quad i \in I \setminus I_1, \tag{34}$$

$$\sum_{s\in N_i^-\cap S_i} q_s^k f_{si} + \sum_{j\in N_i^-\cap I_1}\sum_{s\in S_j} q_s^k x_{ji}^s + \sum_{j\in N_i^-\cap(I\setminus I_1)} v_{ji}^k - \sum_{j\in N_i^+} v_{ij}^k = 0, \quad i \in I \setminus I_1,\ k \in K, \tag{35}$$

$$\sum_{s\in N_t^-\cap S_t} q_s^k f_{st} + \sum_{j\in N_t^-\cap I_1}\sum_{s\in S_j} q_s^k x_{jt}^s + \sum_{j\in N_t^-\cap(I\setminus I_1)} v_{jt}^k - q_t^k \sum_{j\in N_t^-} f_{jt} \le 0, \quad t\in T,\ k\in K, \tag{36}$$

$$\sum_{s\in S_i} y_i^s = 1, \quad i\in I_1, \tag{37}$$

$$\sum_{s\in S_i} x_{ij}^s - f_{ij} = 0, \quad (i,j)\in A,\ i\in I_1, \tag{38}$$

$$\sum_{j\in N_i^+} x_{ij}^s - y_i^s b_i \le 0, \quad i\in I_1,\ s\in S_i, \tag{39}$$

$$x_{ij}^s - y_i^s f_{ij} = 0, \quad (i,j)\in A,\ i\in I_1,\ s\in S_i, \tag{40}$$

$$v_{ij}^k - w_i^k f_{ij} = 0, \quad (i,j)\in A,\ i\in I\setminus I_1,\ k\in K, \tag{41}$$

$$f_{ij}\ge 0, \quad (i,j)\in A,\ j\in N\setminus I_1, \tag{42}$$

$$0\le y_i^s\le 1, \quad i\in I_1,\ s\in S_i. \tag{43}$$

Let C_{HYB} be the hyper-rectangle defined by the bounds on all bilinear variables in (30)–(43) as defined in Proposition 3.

Proposition 4. $z^P\left[C_P\right] \le z^{HYB}\left[C_{HYB}\right] \le z^{MCF}\left[C_{MCF}\right]$.

Proof. The proof is analogous to the proof of Proposition 3. □

5 Computational comparisons

The theoretical part of this work demonstrates that the MCF-formulation is stronger than the HYB-formulation, which in its turn is stronger than the P-formulation. Computational experiments are needed to quantify the differences between the strengths. Experiments are also useful when evaluating whether, and to what extent, stronger formulations lead to faster computation of a narrow interval enclosing the global optimum. To this end, we report in this section the lower bounds obtained by solving the linear relaxations of all three formulations applied to a number of instances. For a comparison of the global optimization capabilities, we have applied BARON 9.3.1 (Sahinidis, 1996) as a global solver to the same set of formulations and instances. All experiments reported in this work

were conducted on a computer equipped with quad-core 3.00GHz processors, where each group of four cores share 8GB of memory.

5.1 Instances

We have tested the formulations in question on 40 instances divided into two sets. The first set consists of standard pooling problem instances from the literature, which have been extended such that all networks contain directed paths intersecting more than one pool. The instances in the second set are generated randomly.

5.1.1 Extended instances from the literature

The instance depicted in Figure 1, henceforth denoted L1, already has an arc between pools, and is therefore not extended. In addition to L1, the first instance set consists of extensions of Examples 1–4 introduced by Adhya et al. (1999) (denoted L2–5), extensions of Problems 4–5 introduced by Ben-Tal et al. (1994) (denoted L6–7), extensions of Examples 2–5 introduced by Foulds et al. (1992) (denoted L8–11), extensions of the instances introduced by Haverly (1978) (denoted L12–14), and an extension of RT2 introduced by Audet et al. (2004) (denoted L15).

The extensions are made as follows. For each pair of pools $\{i, j\} \subseteq I$, where $i \neq j$, we add the arcs (i, j) and (j, i). This does however not produce any extension of instances with only one pool, that is, the instances of Haverly (1978) and Problem 4 (Ben-Tal et al., 1994). In such instances, we therefore first add a new pool i_s for each source s from which there is an arc to some terminal. We then add the arc (s, i_s), and each arc $(s, t) \in A$ where $t \in T$ is replaced by (i_s, t). Finally, a directed clique in the new set of pools is constructed as explained above. Table 1 reports the resulting node set cardinalities, the number of arcs, and the number of quality attributes in each new instance.

Table 1: Characteristics of extended instances from the literature.

Instance	Number of nodes, arcs and attributes				
	$\|S\|$	$\|I\|$	$\|T\|$	$\|A\|$	$\|K\|$
L1	3	2	3	9	1
L2	5	2	4	15	4
L3	5	2	4	15	6
L4	8	3	4	26	6
L5	8	2	5	20	4
L6	4	2	2	10	1
L7	5	3	5	38	2
L8	6	2	4	22	1
L9	11	8	16	216	1
L10	11	8	16	216	1
L11	11	4	16	108	1
L12	3	2	2	9	1
L13	3	2	2	9	1
L14	3	2	2	9	1
L15	3	2	3	18	8

5.1.2 Random instances

The set of randomly generated instances is divided into 5 groups (denoted A, B, C, D and E, respectively), consisting of 5 instances each. All instances within each group have identical number of sources, pools, terminals, and quality attributes.

Arcs are introduced randomly in all instances in groups A–D in such a way that the network becomes acyclic. To this end, the pools are ordered $i_1, \ldots, i_{|I|}$, and arcs from i_k $(k = 2, \ldots, |I|)$ to any of $i_1, \ldots, i_{k-1}$ are avoided. For all other pairs of nodes (i, j), an arc from node i to node j may be introduced if $i \in S \cup I$ and $j \in I \cup T$. The probability of doing so is given by an instance specific parameter referred to as the *expected network density*. In the instances in group E, arcs are introduced in an analogous way, with the only difference that directed cycles of pools are allowed.

The arc costs are defined as $c_{ij} = d_i - d_j$ for all $(i, j) \in A$, where $d_i = 0$ for all pools $i \in I$. For sources and terminals, we let d_i be a randomly generated integer in the domains $\{0, \ldots, 5\}$ and $\{5, \ldots, 14\}$, respectively. All outcomes in a domain have equal probabilities. Similarly, the flow capacities, source quali-

ties and quality bounds, are generated randomly from the domains $\{20, \ldots, 59\}$, $\{0, \ldots, 9\}$, and $\{2, \ldots, 6\}$, respectively. Table 2 reports the node set cardinalities, the number of quality attributes and the range of the expected network densities for each group of instances.

Table 2: Characteristics of randomly generated instances.

Group	#inst.	# of nodes and attributes				Range of expected density
		$\|S\|$	$\|I\|$	$\|T\|$	$\|K\|$	
A	5	3	2	3	2	0.70 – 0.90
B	5	5	4	3	3	0.70 – 0.90
C	5	8	6	6	4	0.50 – 0.70
D	5	12	10	8	5	0.40 – 0.60
E	5	10	10	15	12	0.40 – 0.60

5.2 Comparing the strength of the relaxations

Since the extension procedure suggested in Section 5.1.1 implies that every pool gets another pool as its in-neighbor, we get $I_1 = \emptyset$ in all extended instances. It then follows from (30)–(43) that no proportion variables are defined in the HYB-formulation, and that the HYB-formulation thus degenerates to the P-formulation. Consequently, we report results for these formulations jointly for instances L2–15 and separately for all other instances.

In Tables 3 and 4, where the first column contains instance identifiers, we report the size of each test instance with the P-, HYB-, and MCF-formulations. The size is measured in terms of the number of variables (vars), the number of distinct bilinear terms (nlts), and the number of linear constraints (lcs). We have not provided the number of nonlinear constraints, because in all formulations in question, we have replaced each occurrence of a bilinear term by a new variable, and added a constraint equating the variable with the bilinear term. Hence, the number of nonlinear constraints equals the number of distinct bilinear terms.

For each formulation F, Tables 3 and 4 also report the optimal objective function value $z^F\,[C_F]$ of the relaxation referred to in Proposition 4. This number is given in bold if it is superior to the corresponding values obtained by the other formulations.

Table 3: Instance size and relaxed optimal objective function value of the P-, HYB-, and MCF-formulations for extended instances from the literature.

Instance	P- and HYB-formulations				MCF-formulation			
	vars	nlts	lcs	z^P/z^{HYB}	vars	nlts	lcs	z^{MCF}
L2	63	40	37	-999.32	75	50	59	**-853.47**
L3	87	60	49	-854.10	75	50	67	**-574.78**
L4	152	108	60	-882.84	194	144	108	**-574.78**
L5	76	48	45	-1032.50	132	96	81	**-972.44**
L6	18	6	14	-650.00	42	24	34	**-550.00**
L7	86	42	32	-3500.00	134	84	71	-3500.00
L8	34	10	20	-1200.00	70	40	44	**-1100.00**
L9	408	184	67	-8.00	2328	2024	419	-8.00
L10	408	184	67	-8.00	2328	2024	419	-8.00
L11	188	76	55	-8.00	988	836	215	-8.00
L12	17	6	13	-600.00	33	18	29	**-500.00**
L13	17	6	13	-1200.00	33	18	29	**-1000.00**
L14	17	6	13	-875.00	33	18	29	-875.00
L15	98	64	53	-6331.73	48	24	57	**-6034.87**

It is easily verified that when $|I||K| < \sum_{i\in I}|S_i|$, which is the case in L1–2, L4–14 and all instances in groups A–D, the number of bilinear terms is larger in model [MCF] than in its two competitors. When $|I||K| > \sum_{i\in I}|S_i|$, which is true for L3, L15 and in all instances in group E, the MCF-formulation introduces fewer bilinear terms than the formulations involving quality variables.

All the test instances and the models studied in this paper are available in GAMS-format at http://www.ii.uib.no/~mohammeda/gpooling.

The columns of Tables 3 and 4 labeled z^F show that the MCF-relaxation dominates the P- and HYB-relaxations in 26 out of 40 instances. In all other instances, it gives the same lower bound as provided by the other two formulations. The results do not only confirm the theoretical results of Propositions 3 and 4, but also prove the existence of an instance (C1) in which both inequalities of Proposition 4 are strict. Furthermore, in some test instances (A5, D4, E1–3 and E5), $\left|z^{MCF}\right|$ is less than the half of both $\left|z^P\right|$ and $\left|z^{HYB}\right|$.

Table 4: Instance size and relaxed optimal objective function value of the P-, HYB-, and MCF-formulations for the randomly generated instances.

Instance	P-formulation				HYB-formulation				MCF-formulation			
	vars	nlts	lcs	z^P	vars	nlts	lcs	z^{HYB}	vars	nlts	lcs	z^{MCF}
L1	15	4	15	-43.00	16	6	19	-43.00	24	10	27	-43.00
A1	31	12	20	-1175.00	32	15	25	-1175.00	39	18	34	-1175.00
A2	25	8	17	-641.00	23	8	20	-641.00	28	10	27	-641.00
A3	24	8	19	-420.60	24	10	23	-420.60	30	12	31	-420.60
A4	35	14	20	-669.40	37	18	26	-669.40	44	21	35	**-599.00**
A5	36	14	20	-508.80	34	14	25	-508.80	40	17	33	**-198.00**
B1	69	30	37	-427.37	63	30	46	-427.37	77	35	65	-427.37
B2	88	39	37	-210.00	78	35	45	-210.00	99	47	68	-210.00
B3	98	48	37	-993.33	107	60	46	-993.33	138	80	81	**-932.00**
B4	97	45	37	-912.80	104	55	45	-912.80	135	75	80	-912.80
B5	104	48	37	-439.00	106	53	44	-439.00	138	75	79	-439.00
C1	161	84	74	-1549.56	157	84	83	-1538.41	197	105	141	**-1352.72**
C2	211	116	74	-1222.67	221	134	90	-1222.67	306	194	161	**-682.14**
C3	243	144	74	-2045.75	253	162	95	-2045.75	355	238	170	**-1716.62**
C4	246	140	74	-1535.24	238	140	94	-1535.24	330	208	165	**-1512.10**
C5	266	160	74	-1573.63	280	178	86	-1573.63	414	288	178	**-1071.81**
D1	429	265	130	-2503.40	414	265	151	-2503.40	672	467	315	**-1994.00**
D2	462	270	130	-1883.70	494	317	161	-1883.70	784	538	342	**-1356.51**
D3	478	290	130	-2090.67	479	301	145	-2090.67	795	559	334	**-2071.00**
D4	561	345	130	-1341.70	576	365	143	-1341.70	951	682	355	**-637.86**
D5	586	370	130	-1711.00	586	380	153	-1711.00	1006	736	362	**-1641.80**
E1	1573	1272	345	-1512.10	1573	1272	345	-1512.10	1341	1060	531	**-463.23**
E2	1500	1212	345	-1480.91	1380	1104	349	-1480.91	1187	926	512	**-556.00**
E3	1803	1464	345	-536.12	1803	1464	345	-536.12	1539	1220	547	**-78.68**
E4	2057	1716	345	-1583.42	2057	1716	345	-1583.42	1751	1430	568	**-891.25**
E5	2144	1776	345	-1892.80	2144	1776	345	-1892.80	1828	1480	573	**-221.35**

5.3 Global optimization performance

In order to compare the capabilities of the P-, HYB-, and MCF-formulations to solve the test instances to optimality, we have implemented them in the modeling language GAMS and used the optimizer BARON as global solver. We set the time limit for all formulations to one CPU-hour, and set both the relative and absolute optimality tolerances to 10^{-3}.

For each formulation, Tables 5 and 6 report in the first column (#nodes) the size of the search tree. The second column (lb (time)) gives the lower bound if BARON failed to solve the instance within the time limit, and the CPU-time in parentheses, otherwise. The third column (ub) within each formulation shows the best upper bound found by BARON.

The tables reveal that the MCF-formulation solves all but 2 (L4 and D1) out of 40 instances within the time limit, whereas both the P- and HYB-formulations are successful in this respect in only 22 instances (L1–2, L5–15, A1–5, B1–2 and B4–5). Among these, instances L1, L6–8, L12–14, A1–3, B1–2 and B4–5 could also be solved by all of the formulations in virtually no time (less than a second). This was accomplished without branching in instances L1, L7, A1–3, B1–2 and B4–5. In addition to the instances solved without branching by all formulations, the MCF-formulation solved A4–5, B3 and C1–2 without branching in less than one CPU-second. It also solved instances L9, E1, E3 and E5 without branching in less than one CPU-minute.

Three of the instances (L9–11), all of which are extensions of those provided by Foulds et al. (1992), are solved significantly faster if the P-formulation rather than the MCF-formulation is used. For all formulations, the initial LP-bound is tight in these instances, and therefore optimality can be proved by any heuristic method that happens to output the global optimum. It seems as if the heuristic search provided by BARON is more efficient in the case of the P-formulation, at least in instances L10–11. Here, the optimal solution was hit early in the search, whereas the MCF-formulation required branching and several CPU-minutes in order to conclude.

Among instances solved by all formulations, L2, L5, L15, and A4–5 are solved significantly faster by the MCF-formulation. Only in instances L11 and D5 did

it need more than 3 CPU-minutes, but the running time was close to 39 CPU-minutes in instance D5. A comparison between the P- and the HYB-formulations shows only small differences. They could solve exactly the same set of instances, and only in instances A4–5 a significantly faster convergence can be achieved by application of the HYB-formulation. Concerning the unsolved instances, the HYB-formulation provides better lower bounds in C1–2, and otherwise the bounds are identical.

Instances L4 and D1 could not be solved by any formulation. The superior lower bounding capabilities of the MCF-formulation (Proposition 4) are illustrated in both of these. In instance D1, we are left with an optimality gap less than 1.1%, while the gap in instance L4 is close to 2.0%. Owing to weaker lower bounds, the gaps are significantly larger for the other two formulations.

Table 5: Computational results from BARON applied to the P/HYB-, and MCF-formulations for extended instances from the literature.

Instance	P/HYB-formulation			MCF-formulation		
	#nodes	lb (time)	ub	#nodes	lb (time)	ub
L2	172163	(1489.88)	-549.80	15157	(176.30)	-549.80
L3	189998	-556.38	-549.80	15805	(165.71)	-549.80
L4	56409	-882.84	-561.04	31270	-572.19	-561.04
L5	4320	(61.83)	-877.65	195	(5.04)	-877.65
L6	45	(0.07)	-450.00	25	(0.13)	-450.00
L7	1	(0.05)	-3500.00	1	(0.58)	-3500.00
L8	71	(0.11)	-1100.00	9	(0.10)	-1100.00
L9	0	(1.94)	-8.00	0	(28.41)	-8.00
L10	0	(0.74)	-8.00	11	(157.96)	-8.00
L11	0	(0.60)	-8.00	316	(738.62)	-8.00
L12	73	(0.08)	-400.00	19	(0.06)	-400.00
L13	101	(0.10)	-600.00	21	(0.11)	-600.00
L14	79	(0.08)	-750.00	47	(0.19)	-750.00
L15	2693	(41.46)	-4391.83	11	(0.14)	-4391.83

Table 6: Computational results from BARON applied to the P-, HYB-, and MCF-formulations for randomly generated instances.

Instance	P-formulation			HYB-formulation			MCF-formulation		
	#nodes	lb (time)	ub	#nodes	lb (time)	ub	#nodes	lb (time)	ub
L1	1	(0.02)	-42.58	1	(0.02)	-42.58	1	(0.02)	-42.58
A1	0	(0.02)	-1175.00	0	(0.02)	-1175.00	0	(0.02)	-1175.00
A2	0	(0.02)	-641.00	0	(0.02)	-641.00	0	(0.02)	-641.00
A3	0	(0.02)	-420.60	0	(0.02)	-420.60	0	(0.02)	-420.60
A4	3039	(6.31)	-599.00	619	(1.09)	-599.00	0	(0.02)	-599.00
A5	10577	(25.23)	-198.00	247	(0.57)	-198.00	0	(0.02)	-198.00
B1	1	(0.05)	-427.37	1	(0.04)	-427.37	0	(0.02)	-427.37
B2	0	(0.02)	-210.00	0	(0.03)	-210.00	0	(0.02)	-210.00
B3	168072	-993.33	-932.00	129968	-993.33	-932.00	0	(0.05)	-932.00
B4	0	(0.05)	-912.80	0	(0.03)	-912.80	0	(0.52)	-912.80
B5	1	(0.06)	-439.00	1	(0.08)	-439.00	0	(0.03)	-439.00
C1	51515	-1521.91	-1352.72	49204	-1479.52	-1352.72	0	(0.92)	-1352.70
C2	49226	-1222.67	-673.86	41380	-1220.27	-673.86	1	(0.62)	-673.86
C3	32914	-2045.75	-1716.63	25093	-2045.75	-1716.63	37	(22.00)	-1716.63
C4	21907	-1535.24	-1512.10	15490	-1535.24	-1512.10	109	(35.79)	-1512.10
C5	26661	-1573.63	-1071.81	24931	-1573.63	-1071.81	19	(22.81)	-1071.81
D1	12940	-2503.40	-1932.97	12277	-2503.40	-1911.16	5824	-1994.00	-1973.06
D2	11883	-1883.70	-1356.51	11368	-1883.70	-1356.51	55	(54.74)	-1356.51
D3	10621	-2090.67	-2071.00	10333	-2090.67	-2068.98	118	(97.86)	-2070.33
D4	8686	-1341.70	-637.86	7501	-1341.70	-637.86	127	(94.92)	-637.86
D5	6139	-1711.00	-1641.80	5518	-1711.00	-1641.80	1954	(2324.40)	-1641.80
E1	1483	-1512.10	-461.74	610	-1512.10	-461.74	0	(2.32)	-463.23
E2	762	-1480.91	-550.51	707	-1480.91	-550.51	64	(111.94)	-556.00
E3	766	-536.12	-0.00	1188	-536.12	-0.00	0	(13.91)	-78.68
E4	845	-1583.42	-665.00	755	-1583.42	-219.36	28	(115.52)	-891.25
E5	703	-1892.80	-0.00	858	-1892.80	-0.00	0	(42.59)	-221.35

We have also compared the results in the last column of Table 5 to the results reported in e.g. Adhya et al., 1999. In all of L2–15, the best solution found turns out to have the same cost as the minimum cost in the instances from which they are extended. Note that for L4, the comparison is based upon the solution giving the sharpest upper bound, of which optimality is not proved. The observation shows that the extension described in Section 5.1.1 in none of the reported instances, possibly with the exception of L4, introduced arcs that enable cost reductions. The computational burden is nevertheless increased (L2–5, L9–11), as the PQ-formulation solves the original instances in virtually no time.

The experiments reported in this section indicate that in comparison with the two alternative formulations, the MCF-formulation represents a significantly stronger tool for locating the global optimal solution to instances of the pooling problem with multiple layers of pools. Tighter variable bounds provided in the branching process seem to have modest effect when weaker formulations are applied, whereas our formulation translates this into tighter lower bounds on the optimal objective function value, resulting in faster convergence to optimum.

6 Conclusions

In this paper, we have developed a multi-commodity flow formulation for the generalized pooling problem. The proposed model is an extension of the PQ-formulation for the standard pooling problem, which applies only to networks where all directed paths intersect at most one pool.

We have proved that our multi-commodity flow formulation has stronger relaxation than two other formulations from the literature on the generalized pooling problem. We have presented computational experiments with the proposed formulation and its two competitors applied to 15 extensions of instances from the literature and to 25 randomly generated instances with up to 35 nodes and 12 quality attributes. Experiments confirm that the suggested formulation performs better. By submitting our formulation to the global optimizer BARON, all but two instances could be solved to optimality within one CPU hour, whereas in

the case of the other formulations, BARON missed the global optimum in 18 instances.

Acknowledgements This research was sponsored by the Norwegian Research Council, Gassco, and Statoil under contract 175967/S30

References

Adhya, N., Tawarmalani, M., and Sahinidis, N.V. (1999). A Lagrangian approach to the pooling problem. *Industrial & Engineering Chemistry Research*, **38** (5), 1956–1972.

Al-Khayyal, F.A., and Falk, J.E. (1983). Jointly constrained biconvex programming. *Mathematics of Operations Research*, **8** (2), 273–286.

Alfaki, M., and Haugland, D. (2012). Strong formulations for the pooling problem. *Journal of Global Optimization*. doi:10.1007/s10898-012-9875-6

Audet, C., Brimberg, J., Hansen, P., Le Digabel, S., and Mladenović, N. (2004). Pooling problem: Alternate formulations and solution methods. *Management science*, **50** (6), 761–776.

Ben-Tal, A., Eiger, G., and Gershovitz, V. (1994). Global minimization by reducing the duality gap. *Mathematical Programming*, **63** (2), 193–212.

Foulds, L.R., Haugland, D., and Jörnsten, K. (1992). A bilinear approach to the pooling problem. *Optimization*, **24** (1), 165–180.

Gounaris, C.E., Misener, R., and Floudas, C.A. (2009). Computational comparison of piecewise–linear relaxation for pooling problems. *Industrial & Engineering Chemistry Research*, **48** (12), 5742–5766.

Guisewite, G.M. (1995). Network problems. In R. Horst and P. Pardalos (Eds.), *Handbook of Global Optimization* (pp. 609–648). Dordrecht, The Netherlands: Kluwer Academic Publishers.

Haverly, C.A. (1978). Studies of the behavior of recursion for the pooling problem. *ACM SIGMAP Bulletin*, **25**, 19–28.

Haverly, C.A. (1979). Behavior of recursion model-more studies. *ACM SIGMAP Bulletin*, **26**, 22–28.

Horst, R., Pardalos, P.M., and Thoai, N.V. (2000). *Introduction to Global Optimization* (Second). Dordrecht, The Netherlands: Kluwer Academic Publishers.

Karuppiah, R., and Grossmann, I.E. (2006). Global optimization for the synthesis of integrated water systems in chemical processes. *Journal of Computers & Chemical Engineering*, **30** (4), 650–673.

Li, X., Armagan, E., Tomasgard, A., and Barton, P.I. (2011). Stochastic pooling problem for natural gas production network design and operation under uncertainty. *AIChE Journal*, **57** (8), 2120–2135.

Main, R.A. (1993). Large recursion models: Practical aspects of recursion techniques. In T. Ciriani and R. Leachman (Eds.), *Optimization in Industry: Mathematical Programming and Modeling Techniques in Practice* (pp. 241–249). New York, USA: John Wiley & Sons Ltd.

McCormick, G.P. (1976). Computability of global solutions to factorable nonconvex programs: part I - convex underestimating problems. *Mathematical Programming*, **10** (1), 147–175.

Meyer, C.A., and Floudas, C.A. (2006). Global optimization of a combinatorially complex generalized pooling problem. *AIChE Journal*, **52** (3), 1027–1037.

Misener, R., and Floudas, C.A. (2009). Advances for the pooling problem: Modeling, global optimization, and computational studies. *Applied and Computational Mathematics*, **8** (1), 3–22.

Misener, R., and Floudas, C.A. (2010). Global optimization of large-scale generalized pooling problems: Quadratically constrained MINLP models. *Industrial & Engineering Chemistry Research*, **49** (11), 5424–5438.

Misener, R., Gounaris, C.E., and Floudas, C.A. (2010). Mathematical modeling and global optimization of large-scale extended pooling problems with the (EPA) complex emissions constraints. *Journal of Computers & Chemical Engineering*, **34**, 1432–1456.

Sahinidis, N.V. (1996). BARON: A general purpose global optimization software package. *Journal of Global Optimization*, **8** (2), 201–205.

Sahinidis, N.V., and Tawarmalani, M. (2005). Accelerating branch-and-bound through a modeling language construct for relaxation-specific constraints. *Journal of Global Optimization*, **32** (2), 259–280.

Sherali, H.D. (2002). Tight relaxations for nonconvex optimization problems using the Reformulation-Linearization/Convexification Technique (RLT). In P. Pardalos and E. Romeijn (Eds.), *Handbook of Global Optimization - Volume 2* (pp. 1–64). Dordrecht, The Netherlands: Kluwer Academic Publishers.

Sherali, H.D., and Adams, W.P. (1999). *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Sherali, H.D., Adams, W.P., and Driscoll, P.J. (1998). Exploiting special structures in constructing a hierarchy of relaxations for 0-1 mixed integer problems. *Operations Research*, **46** (3), 396–405.

Tawarmalani, M., and Sahinidis, N.V. (2002). *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Wicaksono, D.S., and Karimi, I.A. (2008). Piecewise MILP under- and overestimators for global optimization of bilinear programs. *AIChE Journal*, **54** (4), 991–1008.

Paper C

Solving the pooling problem with LMI relaxations[2]

Lennart Frimannslund[1], Mohamed El Ghami[1], Mohammed Alfaki[1] and Dag Haugland[1]

C

[1]Department of Informatics, University of Bergen,
P.O. Box 7803, N-5020 Bergen, Norway.
{lennart,mohamed,mohammeda,dag}@ii.uib.no

[2]A short version of this paper is published in: S. Cafieri, B. G.-Tóth, E. Hendrix, L. Liberti and F. Messine (Eds.), *Proceedings of the Toulouse Global Optimization Workshop* (pp. 51–54), 2010.

Solving the pooling problem with LMI relaxations

Lennart Frimannslund[1], Mohamed El Ghami[1],
Mohammed Alfaki[1] and Dag Haugland[1]

1 Department of Informatics, University of Bergen,
P.O. Box 7803, N-5020 Bergen, Norway.
{lennart,mohamed,mohammeda,dag}@ii.uib.no

Abstract

We consider the standard pooling problem with a single quality parameter, which is a polynomial global optimization problem occurring among other places in the oil industry. In this paper, we show that if the feasible set has a nonempty interior, the problem can be solved by a hierarchy of semidefinite relaxations in which the resulting sequences of their optimal values converge to the global optimum. For a fixed relaxation order, this technique provides tight lower bounds for the global objective function value. Based on the experiments, for low order relaxations, the lower bound provided by this method matches the true global optimum in several instances.

Keywords Pooling problem · Linear matrix inequality · Semidefinite programming · Polynomial optimization · Global optimization

1 Introduction

Consider the problem of transporting oil from producers to consumers, through a pipeline network. Suppose we have two sources of oil, for instance two offshore platforms. Suppose in addition that the oil from both sources contains a contaminant which cannot be above a certain level in order for the oil to be usable.

If there are no purification nodes in the network, the only way to control the level of contaminant (or *quality*) of the oil that reaches the terminals is to blend the oil from the different sources either at the terminals or within the network. One can imagine the oil being blended in a big vat, or pool, which gives rise to the name *pooling problem*. The pooling problem in itself is not inherently linked to oil, such problems can also occur with gas, chemicals, beverages or even food production – anywhere, when two or more source ingredients with a notion of quality can be blended in a network.

For oil, the contaminant can be e.g. sulfur, for natural gas, its contaminant can be CO_2, H_2S, or other components. In other words, there can be more than one quality attribute. In this work, however, we will focus on the situation where there is only one such contaminant.

The pooling problem has been studied for many years, see e.g. (Adhya et al., 1999; Haugland, 2010; Misener and Floudas, 2009) and the references therein for a comprehensive treatment. To the best of our knowledge, all existing global optimization methods for this problem (Adhya et al., 1999; Ben-Tal et al., 1994; Foulds et al., 1992; Quesada and Grossmann, 1995; Sahinidis and Tawarmalani, 2005; Visweswaran and Floudas, 1990) employ branch-and-bound based techniques for searching the feasible domain. In this work, we propose an alternative solution method that based on linear matrix inequality (LMI) relaxations proposed by Lasserre (2001a).

The paper is organized as follows. In Section 2, we introduce the problem under study through a popular instance, and give a general formulation for it. Section 3 describes the LMI relaxations for quadratic polynomial optimization problems. In Section 4, we show how we can apply this technique to the pooling problem. Finally, we present numerical experiments in Section 5, and conclude in Section 6.

2 The standard pooling problem with a single quality

2.1 Haverly's first instance

A frequently studied problem instance was constructed by Haverly (1978), and

is henceforth referred to as Haverly1. This instance can be visualized as in Figure 1. As the figure shows, there are three sources on the left, which can provide oil with various levels of sulfur ("S") contamination, at different prices. On the right, there are two terminals, each with an upper bound on the amount of oil needed, which quality is acceptable, and the price they will pay. This is a pooling problem instance because of the structure around node 4.

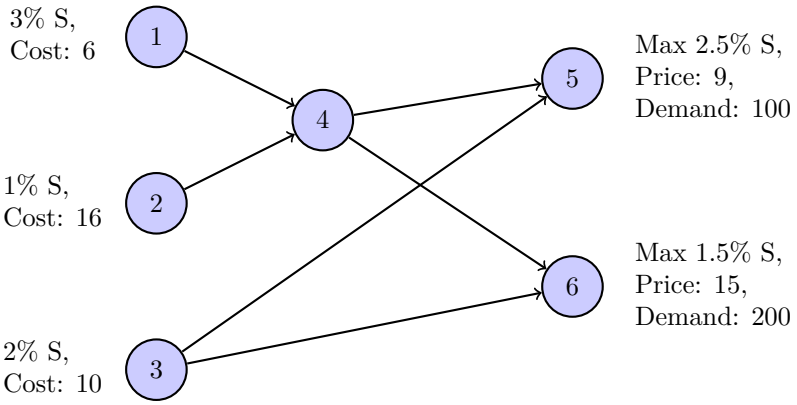


Figure 1: The Haverly1 pooling problem instance.

Here the oil coming in from nodes 1 and 2 is blended, and the sulfur content of the oil exiting node 4 is a weighted average of the sulfur content of the oil entering it. Letting w_4 be the relative sulfur content of the oil exiting node 4, and x_{ij} be the flow from node i to node j, we have:

$$w_4 = \frac{2x_{14} + x_{24}}{x_{14} + x_{24}},$$

which we can also write

$$w_4(x_{14} + x_{24}) = 2x_{14} + x_{24}.$$

This is a bilinear constraint, which in general makes the problem nonconvex and

consequently hard to solve. If we wish to minimize the cost, the entire Haverly1 instance can be formulated as:

$$\begin{aligned}
&\min_{x,w} \; 6x_{14} + 16x_{24} + 10(x_{35} + x_{36}) - 9(x_{35} + x_{45}) - 15(x_{36} + x_{46}), \\
\text{s.t.} \quad & x_{35} + x_{45} \leq 100, \\
& x_{36} + x_{46} \leq 200, \\
& x_{14} + x_{24} - x_{45} - x_{46} = 0, \\
& 3x_{14} + x_{24} - w_4(x_{45} + x_{46}) = 0, \\
& 2x_{35} + w_4 x_{45} - 2.5(x_{35} + x_{45}) \leq 0, \\
& 2x_{36} + w_4 x_{46} - 1.5(x_{36} + x_{46}) \leq 0, \\
& x_{14}, x_{24}, x_{35}, x_{36}, x_{45}, x_{46} \geq 0, \qquad (1) \\
& w_4 \geq 0. \qquad (2)
\end{aligned}$$

The globally optimal solution is:

$$\begin{bmatrix} x_{14} \\ x_{24} \\ x_{35} \\ x_{36} \\ x_{45} \\ x_{46} \\ w_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 100 \\ 0 \\ 100 \\ 0 \\ 100 \\ 1 \end{bmatrix},$$

which corresponds to the objective function value -400. Minimizing the cost is not the only possible objective function, for instance one might want to maximize the flow to the customers, which would have given the objective function

$$\max_{x} \; x_{35} + x_{36} + x_{45} + x_{46},$$

with the same constraints.

Representing the problem with arc flows as the variables and a linear objective function as we have done is called the P-formulation of the problem. The problem can also be written in other ways, for instance with the so-called Q-formulation

(Ben-Tal et al., 1994), which has a nonlinear objective function. We restrict ourselves to the P-formulation in this work.

2.2 General formulation

In general, we define our problem as follows. Consider a directed graph $G = (N, A)$ with the node set N consisting of sources S, pools I and terminals T. Let the arc set A be such that the graph is connected, and such that all arcs link either a source with a pool, a source with a terminal, or a pool with a terminal.

With each node $i \in N$, we define the node *capacity* b_i. At each source or terminal node $i \in S \cup T$ there is defined a quality q_i and a constant c_i. If $s \in S$, q_s is referred to as the *quality parameter* and c_s is the *cost* at that source. If $t \in T$, q_t is referred to as the *quality bound* and c_t is the *price* of selling at this terminal.

Choosing minimization of the total cost as the objective, we can write the problem as:

$$\min_{x,w} \sum_{s\in S} \sum_{j\in N:(s,j)\in A} c_s x_{sj} - \sum_{t\in T} \sum_{i\in N:(i,t)\in A} c_t x_{it},$$

$$\text{s.t.} \quad \sum_{j\in N:(s,j)\in A} x_{sj} \le b_s, \qquad s \in S, \tag{3}$$

$$\sum_{j\in N:(j,i)\in A} x_{ji} \le b_i, \qquad i \in I \cup T, \tag{4}$$

$$\sum_{s\in S:(s,i)\in A} x_{si} - \sum_{t\in T:(i,t)\in A} x_{it} = 0, \qquad i \in I, \tag{5}$$

$$\sum_{s\in S:(s,i)\in A} q_s x_{si} - w_i \sum_{t\in T:(i,t)\in A} x_{it} = 0, \qquad i \in I, \tag{6}$$

$$\sum_{s\in S:(s,t)\in A} q_s x_{sj} + \sum_{i\in I:(i,t)\in A} w_i x_{it} - q_t \sum_{j\in N:(j,t)\in A} x_{jt} \le 0, \quad t \in T, \tag{7}$$

$$x_{ij} \ge 0, \qquad (i,j) \in A.$$

Inequalities (3) and (4) are the flow capacity constraints at all nodes, and (7) is the quality constraints at each terminal. Equation (5) is the flow conservation constraint for each pool, and Equation (6) is the corresponding quality balance

constraint, stating that the amount of the contaminant entering a pool equals the amount leaving it.

As one can see the objective function is linear, and the constraints are either linear or bilinear. In other words, even though the feasible region is nonconvex, both the objective function and the constraints are polynomial, and in fact quadratic. We can therefore apply polynomial optimization techniques.

3 Polynomial optimization by LMI relaxations

We now give a brief outline of the technique of global optimization of (quadratic) polynomials by linear matrix inequality (LMI) relaxations, as presented by Lasserre (2001b). For a more thorough introduction of the case of polynomials of any degree, see (Lasserre, 2001a).

3.1 Moments and moment matrices

For a polynomial function f on $\mathbb{R}^n$, we wish to compute

$$f^* = \min_{x \in K} f(x), \tag{8}$$

where K is a subset of $\mathbb{R}^n$ defined by polynomial constraints

$$g_k(x) \geq 0, \;\; k = 1, 2, \ldots, m.$$

An equivalent problem is

$$\min_{\mu \in \mathcal{P}(K)} \int f(x) \mu(dx),$$

where we minimize over the space of finite Borel signed measures with their support in K. This is an infinite dimensional problem, so instead of determining the measure itself we try to determine its moments y defined as

$$y_\alpha = \int x^\alpha \mu(dx),$$

where α denotes a collection of n indices and x^α is a product of the components $x_1, x_2, \ldots, x_n$ to the power of the corresponding index in α. For example, if $n = 3$, then

$$y_{456} = \int x_1^4 x_2^5 x_3^6 \mu(dx).$$

For moments corresponding to a probability distribution, the moment matrix of order i is a positive semidefinite matrix $M_i(y)$ containing all moments up to order $2i$, which satisfies

$$p^T M_i(y) p = \int [p(x)]^2 \mu_y(dx),$$

where p is a vector of the coefficients of a polynomial, ordered so that they correspond to the entries of the moment matrix. Here, μ_y is the (not necessarily unique) probability distribution corresponding to the moments in $M_i(y)$. If $n = 2$ and $i = 2$ then

$$M_2(y) = \left[\begin{array}{c|cc|ccc} 1 & y_{10} & y_{01} & y_{20} & y_{11} & y_{02} \\ \hline y_{10} & y_{20} & y_{11} & y_{30} & y_{21} & y_{12} \\ y_{01} & y_{11} & y_{02} & y_{21} & y_{12} & y_{03} \\ \hline y_{20} & y_{30} & y_{21} & y_{40} & y_{31} & y_{22} \\ y_{11} & y_{21} & y_{12} & y_{31} & y_{22} & y_{13} \\ y_{02} & y_{12} & y_{03} & y_{22} & y_{13} & y_{04} \end{array}\right].$$

In this case, p contains the coefficients of the base polynomials

$$1, x_1, x_2, x_1^2, x_1 x_2, x_2^2,$$

in this order. Similarly, there exist moment matrices relating to the constraints. For each constraint $g_k(x) \geq 0,\ k = 1, 2, \ldots, m$, there exists a matrix $M_i(g_k y)$, so that

$$p^T M_i(g_k y) p = \int g_k(x) [p(x)]^2 \mu_y(dx).$$

3.2 LMI Relaxations

These two matrix types are the main ingredients in the following hierarchy of convex LMI relaxations for quadratic problems:

$$\mathbb{Q}_i = \begin{cases} \min\limits_{y} \ \sum\limits_{\alpha} (g_0)_\alpha y_\alpha, \\ \text{s.t.} \quad\quad M_i(y) \succeq 0, \\ \qquad\quad M_{i-1}(g_k y) \succeq 0, \quad k = 1, 2 \ldots, m, \end{cases} \tag{9}$$

where g_0 is a vector containing the coefficients of $f(x)$ ordered according to the order of all the moments $\{y_\alpha\}$ stacked in a vector. The moments y_α are the unknowns. For the different orders of relaxations and the original problem (8), we have (Lasserre, 2001b, Proposition 3.1),

$$\inf \mathbb{Q}_i \leq \inf \mathbb{Q}_{i+1} \leq f^*, \ i = 1, 2, \ldots$$

Under certain conditions we have that, for increasing i, the objective function value

$$\lim_{i \to \infty} \inf \mathbb{Q}_i \uparrow f^* \tag{10}$$

A sufficient condition for (10) to hold that can be applied to the pooling problem is that we can add a redundant constraint of the form

$$g_{m+1}(x) = M - \|x\|^2 \geq 0, \tag{11}$$

for some finite constant M. For the exact requirements, see (Lasserre, 2001a,b). Let $\mathbb{Q}_i^*$ denote the dual of (9). It can be written

$$\mathbb{Q}_i^* = \begin{cases} \max\limits_{X, Z_k} \ -X_{11} - \sum\limits_{k=1}^{m} (g_k)_0 (Z_k)_{11}, \\ \text{s.t.} \quad X \bullet B_\alpha + \sum\limits_{k=1}^{m} Z_k \bullet C_\alpha^k = (g_0)_\alpha, \quad \forall \alpha \neq 0, \\ \qquad\qquad\qquad\qquad X, Z_k \succeq 0, \quad k = 1, 2, \ldots, m. \end{cases} \tag{12}$$

Here, X_{11} and $(Z_k)_{11}$ denote the elements in position (1,1) of these matrices, and $(g_k)_0$ denotes the constant term in the expression $g_k(x) \geq 0$. The operator $\bullet$ denotes the inner product between symmetric matrices, that is, $A \bullet B = trace(AB)$. The matrices B_α and C_α are defined by

$$\sum_{\alpha} y_\alpha B_\alpha \quad = \quad M_i(y), \text{ and}$$

$$\sum_{\alpha} y_\alpha C_\alpha^k \quad = \quad M_{i-1}(g_k y),$$

where the sum is over each component in α, that is, corresponding to each unique member of $M_i(y)$. The y-component corresponding to the constant terms is chosen to be 1. Similarly, the expression $\forall \alpha \neq 0$ in the constraints of (12) means that there is one such constraint for each component of α not equal to zero.

3.3 Finite convergence

In many cases, there exists an integer i_0 such that

$$\max \mathbb{Q}_i^* = \inf \mathbb{Q}_i = f^* \quad \forall\ i \geq i_0, \tag{13}$$

and in some instances i_0 is relatively small, say 2 or 3. If K has a nonempty interior, then a necessary and sufficient condition for (13) to hold is that the polynomial $f(x) - f^*$ can be written as a sum of squares, particularly (see Equation 18.12 in Lasserre, 2001b):

$$f(x) - f^* = \sum_{j=1}^{r_0} [p_j(x)]^2 + \sum_{k=1}^{m} g_k(x) \left[\sum_{j=1}^{r_k} [p_{kj}(x)]^2 \right], \tag{14}$$

where the functions $p_j(x)$ and $p_{kj}(x)$ are polynomials, for all j and all (k, j)-pairs. The coefficients of these polynomials can be retrieved from the dual problem $\mathbb{Q}_i^*$. Specifically, if (14) holds and given the solution to $\mathbb{Q}_i^*$ (e.g. $X, Z_k,\ k = 1, \ldots, m$), then if we let p_j and p_{kj} be the vectors of coefficients for the corresponding polynomials, we have

$$\begin{aligned} \sum_{i=1}^{r_0} p_j p_j^T &= X, \text{ and} \\ \sum_{j=1}^{r_{kj}} p_{kj} p_{kj}^T &= Z_k,\ k = 1, 2, \ldots, m. \end{aligned}$$

If K does *not* have a nonempty interior then the theory of Lasserre (2001b) cannot give an if-and-only-if condition for finite convergence. Nevertheless, as

pointed out in the introduction of Lasserre (2001a), from a numerical point of view this is not important, and numerical results are promising for this case as well (Lasserre, 2001b). Recent results regarding the representation of polynomials (Kojima and Muramatsu, 2009) may close this theoretical gap in the future.

Given a solution $\max \mathbb{Q}_i^*$, then the corresponding x-variables for the original problem can be extracted from $M_i(y)$ using the procedure by Henrion and Lasserre (2005) or be read from the first order moments directly. To verify global optimality, the obtained x must be feasible for the original problem, and attain the same objective function value.

If the original polynomial optimization problem has n variables and m constraints, then the i-th order LMI relaxation (9) has $O(n^{2i})$ variables and $m+1$ positive semidefiniteness constraints. The LMI can be cast as an SDP problem (Lasserre, 2008). Such problems can be solved in polynomial time using interior point methods. In other words, any polynomial optimization problem for which there exists an i_0 such that (13) holds in all instances of the problem, is solvable in polynomial time. It is however important to note that i_0 must be independent of the problem instance.

4 LMI relaxations applied to the pooling problem

For the solution framework in Section 3 to work, the set of feasible solutions must have a nonempty interior. This means we will have to eliminate all equality constraints somehow. We show how to do this by example, on the Haverly1 instance.

4.1 Preprocessing the Haverly1 instance

First, we eliminate the flow conservation equation, that is,

$$x_{14} + x_{24} - x_{45} - x_{46} = 0.$$

Let us perform the substitution

$$x_{14} \leftarrow (-x_{24} + x_{45} + x_{46}).$$

This gives the following formulation:

$$\begin{aligned}
\min_{x,w}\ & 10x_{24} + x_{35} - 5x_{36} - 3x_{45} - 9x_{46}, \\
\text{s.t.}\ & x_{35} + x_{45} \leq 100, \\
& x_{36} + x_{46} \leq 200, \\
& -2x_{24} + 3x_{45} + 3x_{46} - w_4(x_{45} + x_{46}) = 0, \\
& 2x_{35} + w_4 x_{45} - 2.5(x_{35} + x_{45}) \leq 0, \\
& 2x_{36} + w_4 x_{46} - 1.5(x_{36} + x_{46}) \leq 0, \\
& -x_{24} + x_{45} + x_{46} \geq 0, \\
& x_{24}, x_{35}, x_{36}, x_{45}, x_{46} \geq 0, \\
& 1 \leq w_4 \leq 3.
\end{aligned}$$

Note here that we have replaced the constraint $w_4 \geq 0$ in (2) with the tighter constraints $w_4 \leq 3$, and $w_4 \geq 1$. These bounds are easily identified, since the quality of the flow that leaves a pool is bounded by the qualities of the flows entering it. Note also that constraint (1), which includes the nonnegativity constraint $x_{14} \geq 0$ subject to substitution. We continue in the same fashion and use the remaining equality constraint to remove the variable x_{24}, by performing the substitution

$$x_{24} \leftarrow 1.5x_{45} + 1.5x_{46} - 0.5w_4(x_{45} + x_{46}).$$

This gives the formulation:

$$\begin{aligned}
\min_{x,w}\ & x_{35} - 5x_{36} + 12x_{45} + 6x_{46} - 5w_4(x_{45} + x_{46}), \\
\text{s.t.}\ & x_{35} + x_{45} \leq 100, \\
& x_{36} + x_{46} \leq 200, \\
& 2x_{35} + w_4 x_{45} - 2.5(x_{35} + x_{45}) \leq 0, \\
& 2x_{36} + w_4 x_{46} - 1.5(x_{36} + x_{46}) \leq 0, \\
& x_{45} + x_{46} - (1.5x_{45} + 1.5x_{46} - 0.5w_4(x_{45} + x_{46})) \geq 0, \\
& 1.5x_{45} + 1.5x_{46} - 0.5w_4(x_{45} + x_{46}) \geq 0, \\
& x_{35}, x_{36}, x_{45}, x_{46} \geq 0,
\end{aligned}$$

$$1 \leq w_4 \leq 3.$$

Note that the objective function is now nonlinear, and that the two last constraints correspond to the nonnegativity constraints on the two eliminated variables, $x_{14} \geq 0$ and $x_{24} \geq 0$. We can visualize this reduced formulation as in Figure 2.

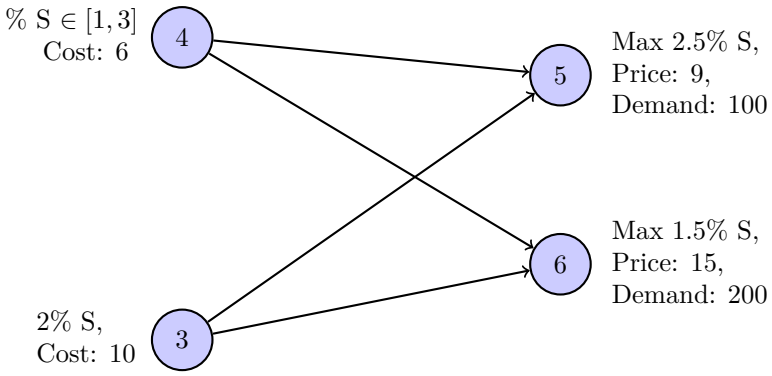


Figure 2: The Haverly1 pooling problem instance, with two variables substituted.

The feasible region has a nonempty interior, since all the inequalities hold strictly, e.g.

$$\begin{bmatrix} x_{35} \\ x_{36} \\ x_{45} \\ x_{46} \\ w_4 \end{bmatrix} = \begin{bmatrix} 6 \\ 6 \\ 1 \\ 1 \\ 7/6 \end{bmatrix}.$$

Having eliminated the equality constraints, we can add a constraint of the form (11). In the current instance, it can be

$$(10^5 + 9) - x_{35}^2 - x_{36}^2 - x_{45}^2 - x_{46}^2 - w_4^2 \geq 0.$$

4.2 Preprocessing general instances

All pooling problem instances have a flow conservation and quality balance equation for each pool, but these can be eliminated by the method outlined in the example.

There may be additional equalities present stemming from the bounds on the flow variables, as well as the bounds on the quality entering the pools. In most cases these equalities can be eliminated by repeated application of one or more of the following preprocessing steps:

1. No flow possible because of quality constraints ⇒ remove edge.
2. Disconnected node ⇒ remove node
3. Quality or flow restricted to one value ⇒ replace variable in question with a constant.

However, the feasible set for a formulation with no equality constraints is not always guaranteed to have a nonempty interior, but the converse is true. For example, if we have the network to the left in Figure 3, using the technique we have described, we can eliminate all equality constraints, but the feasible set still has empty interior.

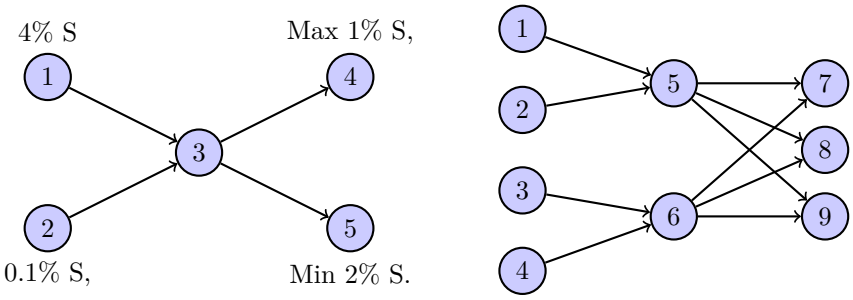


Figure 3: To the left, an instance where the feasible set has a nonempty interior. To the right the network used in the experiments.

5 Numerical experiments

We test the LMI relaxation technique using the software package Gloptipoly 3 (Henrion et al., 2009). First we try a few instances from the literature, listed in the first five rows of Table 2. All of these are defined in e.g. (Adhya et al., 1999).

As Table 2 reflects, the first order relaxation only finds lower bounds on global optimum f^* of the original problem. For order 2 we have $\max \mathbb{Q}_2^* = f^*$ and Gloptipoly is able to extract the globally optimal solution x^* for all but the Foulds2 instance. Foulds2 has infinitely many globally optimal solutions corresponding the same value of the qualities at its two pools. One such solution is identifiable from the first-order moments in the moment matrix $M_2(y)$, but the software is not able to detect this, presumably for numerical reasons. It should, namely, be noted that all of the instances tested are extremely sensitive to scaling, and most can only be solved successfully with Gloptipoly with the variables scaled to their expected magnitude at the optimal solution. Since the problem is invariant to scaling this is a numerical, not a theoretical issue. Unfortunately we are not able to solve Foulds2 with relaxation order 3 due to lack of memory on the computer used for testing.

We also generated some instances, using the graph to the right in Figure 3. The different values for the parameters and constraints are found in Table 1.

The first instance is defined to have a nonempty interior and consistency between quality and price. The rest were randomly generated. As it turns out, instances B and D have empty interiors. For B, no flow can reach the second of third terminals because of their strict constraints on the quality. For D, no flow can reach the first and second terminals, for the same reason. We therefore construct two versions of each of these instances. Instance B and D denote the original instances, and B2 and D2 denote the same instances with the unreachable terminals removed. The results in these instances are reported in the bottom half of Table 2.

Table 1: Parameter settings for test instances.

	Sources			Terminals		
	Cost	Quality parameter	Flow capacity	Price	Quality bound	Flow capacity
A	$\begin{bmatrix} 10 \\ 4 \\ 5 \\ 2 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 3 \\ 2 \\ 4 \end{bmatrix}$		$\begin{bmatrix} 20 \\ 7 \\ 5 \end{bmatrix}$	$\leq \begin{bmatrix} 1.5 \\ 2.5 \\ 3.5 \end{bmatrix}$	$\leq \begin{bmatrix} 100 \\ 100 \\ 100 \end{bmatrix}$
B	$\begin{bmatrix} 8.5577 \\ 6.7080 \\ 5.2359 \\ 2.9882 \end{bmatrix}$	$\begin{bmatrix} 7.0397 \\ 3.8161 \\ 5.6768 \\ 8.8786 \end{bmatrix}$		$\begin{bmatrix} 0.8747 \\ 2.6073 \\ 0.2280 \end{bmatrix}$	$\leq \begin{bmatrix} 8.1544 \\ 0.0136 \\ 0.0309 \end{bmatrix}$	$\leq \begin{bmatrix} 84.2949 \\ 89.8799 \\ 93.9003 \end{bmatrix}$
C	$\begin{bmatrix} 6.4832 \\ 6.1467 \\ 4.6965 \\ 5.7778 \end{bmatrix}$	$\begin{bmatrix} 9.1131 \\ 3.7622 \\ 2.2876 \\ 4.2352 \end{bmatrix}$	$\leq \begin{bmatrix} 6.1825 \\ 23.1367 \\ 11.8486 \\ 9.8780 \end{bmatrix}$	$\begin{bmatrix} 3.0035 \\ 4.0003 \\ 5.1772 \end{bmatrix}$	$\leq \begin{bmatrix} 5.3464 \\ 3.8544 \\ 8.7345 \end{bmatrix}$	$\leq \begin{bmatrix} 27.3596 \\ 44.4566 \\ 62.7515 \end{bmatrix}$
D	$\begin{bmatrix} 8.1472 \\ 9.0579 \\ 1.2699 \\ 9.1338 \end{bmatrix}$	$\begin{bmatrix} 6.3236 \\ 0.9754 \\ 2.7850 \\ 5.4688 \end{bmatrix}$		$\begin{bmatrix} 8.0028 \\ 1.4189 \\ 4.2176 \end{bmatrix}$	$\geq \begin{bmatrix} 9.7059 \\ 9.5717 \\ 4.8538 \end{bmatrix}$	$\leq \begin{bmatrix} 95.7507 \\ 96.4889 \\ 15.7613 \end{bmatrix}$

Table 2: Results from five instances from the literature and the instances defined in Table 1.

Name	max $\mathbb{Q}_i^*$ for relaxation order			Notes
	1	2	3	
Haverly1	-600	-400	–	x^* found at order 2
Haverly2	-1200	-600	–	x^* found at order 2
Haverly3	-875	-750	–	x^* found at order 2
BenTal4	-600	-450	–	x^* found at order 2
Foulds2	-1200	-1100	–	Sol. in $M_2(y)$
A	-1925	-1553	-1541	x^* found at order 3
B	0	–	–	x^* found at order 1
B2	0	–	–	x^* found at order 1
C	-5.69	-5.69	–	x^* found at order 2
D	0	–	–	x^* found at order 1
D2	0	–	–	x^* found at order 1

Two things happen here that is different from the results of the experiments with the instances from the literature. One is that for instance A, the solution is not found with relaxation order 2, but that order 3 is needed. Secondly, for instances with zero as their solution the global optimum is found and verified for order 1.

Table 3: Results from experiments with max flow instances.

Name	max $\mathbb{Q}_i^*$ for relaxation order			Notes
	1	2	3	
Haverly1	300	300	–	x^* found at order 2
Haverly2	800	800	–	x^* found at order 2
Haverly3	300	300	–	x^* found at order 2
BenTal4	300	300	–	x^* found at order 2
Foulds2	600	600	–	x^* found at order 2
A	300	300	–	x^* found at order 2
B	181	84.30	84.29	x^* found at order 3
B2	84.29	–	–	x^* found at order 1
C	51.04	51.04	–	x^* found at order 2
D	22.89	15.76	15.76	x^* found at order 3
D2	15.76	15.76	–	x^* found at order 2

We also test the same instances, but with a max flow objective function.

These results are reported in Table 3. For the max flow objective function, the instances with an empty interior requires a higher relaxation order than for min cost, namely order 3. In the remaining instances relaxation order 1 always provides the correct objective function value, and in the B2 instance, this applies to the variables as well. In the rest of the instances, relaxation order 2 is needed in order to obtain the optimal variable values.

6 Conclusion

We use LMI relaxations for solving the standard pooling problem with a single quality. Based on our experiments, small instances of this problem can be solved with low LMI relaxation orders, provided that they can be formulated in such a way that they have a nonempty interior. However, solving such relaxation implies a large computational effort, which for large instances makes the method impractical to use. Current research on solution methods for sparse semidefinite optimization has good promise to ease this limitation.

Acknowledgements This research was sponsored by the Norwegian Research Council, Gassco, and Statoil under contract 175967/S30

References

Adhya, N., Tawarmalani, M., and Sahinidis, N.V. (1999). A Lagrangian approach to the pooling problem. *Industrial & Engineering Chemistry Research*, **38** (5), 1956–1972.

Ben-Tal, A., Eiger, G., and Gershovitz, V. (1994). Global minimization by reducing the duality gap. *Mathematical Programming*, **63** (2), 193–212.

Foulds, L.R., Haugland, D., and Jörnsten, K. (1992). A bilinear approach to the pooling problem. *Optimization*, **24** (1), 165–180.

Haugland, D. (2010). An overview of models and solution methods for pooling problems. In E. Bjørndal, M. Bjørndal, P. Pardalos and M. Rönnqvist. (Eds.), *Energy, Natural Resources and Environmental Economics* (pp. 459–469). Springer.

Haverly, C.A. (1978). Studies of the behavior of recursion for the pooling problem. *ACM SIGMAP Bulletin*, **25**, 19–28.

Henrion, D., and Lasserre, J.B. (2005). Detecting global optimality and extracting solutions in GloptiPoly. *Positive polynomials in control*, 293–310.

Henrion, D., Lasserre, J.B., and Löfberg, J. (2009). GloptiPoly 3: moments, optimization and semidefinite programming. *Optimization Methods & Software*, **24** (4-5), 761–779.

Kojima, M., and Muramatsu, M. (2009). A note on sparse SOS and SDP relaxations for polynomial optimization problems over symmetric cones. *Computational Optimization and Applications*, **42** (1), 31–41.

Lasserre, J.B. (2001a). Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, **11** (3), 796–817.

Lasserre, J.B. (2001b). New positive semidefinite relaxations for nonconvex quadratic programs. In H. Hadjisavvas and P. Paradalos (Eds.), *Advances in Convex Analysis and Global Optimization: Honoring the Memory of C. Caratheodory (1873-1950)* (pp. 161–189). Kluwer Academic Publishers.

Lasserre, J.B. (2008). A semidefinite programming approach to the generalized problem of moments. *Mathematical Programming*, **112** (1), 65–92.

Misener, R., and Floudas, C.A. (2009). Advances for the pooling problem: Modeling, global optimization, and computational studies. *Applied and Computational Mathematics*, **8** (1), 3–22.

Quesada, I., and Grossmann, I.E. (1995). Global optimization of bilinear process networks with multi-component flows. *Computers & Chemical Engineering*, **19** (12), 1219–1242.

Sahinidis, N.V., and Tawarmalani, M. (2005). Accelerating branch-and-bound through a modeling language construct for relaxation-specific constraints. *Journal of Global Optimization*, **32** (2), 259–280.

Visweswaran, V., and Floudas, C.A. (1990). A global optimization algorithm (GOP) for certain classes of nonconvex NLPs–II. Application of theory and test problems. *Computers & chemical engineering*, **14** (12), 1419–1434.

Paper D

Comparison of discrete and continuous models for the pooling problem[2]

Mohammed Alfaki[1] and Dag Haugland[1]

D

[1]Department of Informatics, University of Bergen,
P.O. Box 7803, N-5020 Bergen, Norway.
mohammeda@ii.uib.no and dag@ii.uib.no

[2]In: A. Caprara and S. Kontogiannis (Eds.), *11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems* (Vol. 20, pp. 112–121). OpenAccess Series in Informatics (OASIcs). Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, doi: 10.4230/OASIcs.ATMOS.2011.112, 2011.

Comparison of discrete and continuous models for the pooling problem

Mohammed Alfaki[1] and Dag Haugland[1]

1 Department of Informatics, University of Bergen,
P.O. Box 7803, N-5020 Bergen, Norway.
mohammeda@ii.uib.no and dag@ii.uib.no

Abstract

The pooling problem is an important global optimization problem which is encountered in many industrial settings. It is traditionally modeled as a bilinear, nonconvex optimization problem, and solved by branch-and-bound algorithms where the subproblems are convex. In some industrial applications, for instance in pipeline transportation of natural gas, a different modeling approach is often made. Rather than defining it as a bilinear problem, the range of qualities is discretized, and the complicating constraints are replaced by linear ones involving integer variables. Consequently, the pooling problem is approximated by a mixed-integer programming problem. With a coarse discretization, this approach represents a saving in computational effort, but may also lead to less accurate modeling. Justified guidelines for choosing between a bilinear and a discrete model seem to be scarce in the pooling problem literature. In the present work, we study discretized versions of models that have been proved to work well when formulated as bilinear programs. Through extensive numerical experiments, we compare the discrete models to their continuous ancestors. In particular, we study how the level of discretization must be chosen if a discrete model is going to be competitive in both running time and accuracy.

Keywords Global optimization · Industrial optimization · Graphs and networks · Pooling problem

1 Introduction

The pooling problem is an important industrial optimization problem that originates from the petroleum refineries. It can be considered as an extension of the minimum cost flow problem on networks of three sets of nodes, referred to as sources, pools and terminals. From each source, a raw material is supplied to the network. The qualities of the raw materials depend on the source from which they are supplied. At the pools, raw materials of possibly unequal qualities are mixed to form intermediate products. In their turn, the intermediate products are blended again to form end products at the terminals. The resulting qualities of end products thus depend on what sources they originate from, and in what proportions. Restrictions, which may vary between the terminals, apply to these qualities.

Earlier work on the optimization of the pooling problem can be traced back to Haverly (1978), and since then there has been a continuous interest in the problem. Mainly, the literature focuses on formulations, solution methods, applications, and experimental evaluations.

The problem is often formulated as a non-convex, continuous optimization problem, and many solution methods have been proposed to solve it. The ambition of the earliest approaches was to find a good local optimum. This includes the popular method of Haverly (1978), which solves a sequence of linear programs approximating the problem. Based on Benders decomposition, Floudas and Aggarwal (1990) proposed an algorithm to search for the global solution. Building on this, Floudas and Visweswaran (1990) developed an algorithm based on Lagrangian relaxation techniques. Other Lagrangian-based algorithms were proposed by Adhya et al. (1999) and Almutairi and Elhedhli (2009). Foulds et al. (1992) developed a branch-and-bound algorithm based on linear relaxations of bilinear programs as suggested by McCormick (1976) and Al-Khayyal and Falk (1983).

Several continuous formulations have been proposed for the pooling problem. In addition to traditional network flow variables, the models also need some representation of product quality. The most straightforward approach (Haverly, 1978), is to introduce a decision variable for each pool, and to let the variable

be defined as the quality of the product at the given pool. As an alternative, Ben-Tal et al. (1994) proposed a formulation where the quality variables are replaced by variables representing the proportions in which the pool receives flow from various sources. Tawarmalani and Sahinidis (2002) strengthen this formulation by the application of reformulation-linearization technique (RLT) suggested by Sherali and Adams (1999). Following the idea of proportion variables, Alfaki and Haugland (2012b) proposed two formulations: In the first model, the source proportions introduced in (Ben-Tal et al., 1994) are replaced by terminal proportions. By combining source and terminal proportions, the second model becomes stronger than the first and also stronger than the model in (Tawarmalani and Sahinidis, 2002).

In its traditional form, the pooling problem is defined on tripartite networks where all arcs connect a source to a pool, a pool to a terminal or a source to a terminal. Contrary to formulations with quality variables, formulations based on proportion variables cannot easily be generalized to arbitrary networks. Audet et al. (2004) considered the case where there are connections between pools, and suggested a hybrid formulation involving both quality and proportion variables. Using only flow and proportion variables, Alfaki and Haugland (2012a) proposed a multi-commodity flow formulation for arbitrary networks, and proved that it dominates the hybrid formulation and a quality based formulation.

The above mentioned continuous formulations all have bilinear constraints. For the models with quality variables, this can be explained by the fact that the quality at a pool is defined as the weighted average of entering qualities, where the flow constitutes the weights. Reflecting the $\mathcal{NP}$-hardness of the problem (Alfaki and Haugland, 2012b), bilinear constraints seem inescapable in continuous formulations, and represent a serious challenge to solution algorithms. Consequently, there is a need for easy-to-use and well studied solution strategies, such as mixed integer programs. This can be seen from the work of Faria and Bagajewicz (2008), who discretized the quality variables of the wastewater treatment problem, which is closely related to the pooling problem, and replaced the bilinear constraints by "big M" constraints. Pushing in the same direction, Pham et al. (2009) and Pham (2007) eliminated the bilinear terms by discretizing the quality

variables. Consequently, the pooling problem is approximated by a mixed-integer programming problem.

In this paper, we generalize the discretization approach proposed in (Pham et al., 2009) to arbitrary networks. Through numerical experiments on large scale instances, we compare our discrete formulation with a continuous formulation. The purpose of this is to investigate whether discrete models are more suitable for finding good solutions when the global optimum is out of reach. By lower bounding techniques, we also aim to estimate the error introduced by discretizing the solution space.

The remainder of the paper is organized as follows: Section 2 introduces the pooling problem and one of its continuous formulations, and gives a brief description of the traditional solution methods. In Section 3, we present our discrete model and its extension to arbitrary networks. The numerical experiments are reported in Section 4, and major conclusions are summarized in Section 5.

2 Problem statement and formulation

We consider a directed graph (network) $G = (N, A)$ with node set N and arc set A. For each node $i \in N$, let $N_i^- = \{j \in N : (j,i) \in A\}$ and $N_i^+ = \{j \in N : (i,j) \in A\}$ denote the set of in- and out-neighbors of i, respectively. We assume that G has non-empty sets $S, T \subseteq N$ of *sources* and *terminals*, respectively, where $N_s^- = \emptyset \ \forall s \in S$ and $N_t^+ = \emptyset \ \forall t \in T$. We refer to all nodes in $I = N \setminus (S \cup T)$ as *pools*. We define a finite set of *quality attributes* K. With each $i \in S \cup T$, we associate a real constant q_i^k for each $k \in K$. If $s \in S$, q_s^k is referred to as the *quality parameter* of attribute k at that source, and if $t \in T$, q_t^k is referred to as the *quality bound* of attribute k at terminal t. For each $i \in N$, we define the constant *flow capacity* b_i, and for each arc $(i,j) \in A$, we define the constant *unit cost* c_{ij}. This is slightly more general than defining costs and revenues only at the sources and the terminals, respectively, which is common practice in the pooling problem literature. For each $i \in N$, let S_i be the set of sources from which there exists a path to i in G.

Define f_{ij} as the flow along the arc $(i,j) \in A$, and w_i^k $(k \in K)$ as the quality of flow leaving pool $i \in I$. The pooling problem is to assign flow values to all arcs

in the pooling network such that each flow capacity b_i is respected at all nodes $i \in I$, and such that the total flow cost is minimized. Besides that, the quality of the flow leaving any pool is given as the weighted average of the quality of entering flow, where the flow values constitute the weights. More precisely, the matrix of qualities satisfies

$$\sum_{s \in N_i^- \cap S} q_s^k f_{si} + \sum_{j \in N_i^- \setminus S} w_j^k f_{ji} = w_i^k \sum_{j \in N_i^-} f_{ji}, \qquad i \in N \setminus S,\ k \in K, \tag{1}$$

if $\sum_{j \in N_i^-} f_{ji} > 0$. Otherwise, w_i^k is given an arbitrary value. In addition, the flow arriving at terminal $t \in T$ must for all attributes $k \in K$ satisfy the quality bounds q_t^k. Assuming that the qualities also at the terminals are given as weighted averages of entering flow, we arrive at the constraints:

$$\sum_{s \in N_t^- \cap S} q_s^k f_{st} + \sum_{j \in N_t^- \setminus S} w_j^k f_{jt} \leq q_t^k \sum_{j \in N_t^-} f_{jt}, \qquad t \in T,\ k \in K. \tag{2}$$

Instead of defining quality variables, we associate a flow *commodity* with each source $s \in S$, where at most b_s units of the commodity can enter the network, and the commodity can leave the network at any $t \in T$. At all other nodes, the commodity neither enters nor leaves the network. Now, the variable f_{ij} defines the total flow of all commodities along arc $(i, j) \in A$. Relative to the total flow leaving node $i \in S \cup I$, let the variable y_i^s denote the proportion of commodity s (define $y_i^s = 0$ if $s \notin S_i$ and $y_s^s = 1$). Therefore, the quantity $y_i^s f_{ij}$ defines the flow of commodity s (meaning the commodity associated with source s, we simply refer to s as a commodity whenever convenient) along the arc (i, j). Based on the multi-commodity flow formulation, Alfaki and Haugland (2012a) proposed the following formulation to the pooling problem:

$$z^* = \min \quad \sum_{(i,j) \in A} c_{ij} f_{ij} \tag{3}$$

$$\text{s.t.} \quad \sum_{j \in N_i^+} f_{ij} \leq b_i, \qquad i \in S \cup I, \tag{4}$$

$$\sum_{j \in N_t^-} f_{jt} \leq b_t, \qquad t \in T, \tag{5}$$

$$\sum_{j \in N_i^-} y_j^s f_{ji} - \sum_{j \in N_i^+} y_i^s f_{ij} = 0, \qquad s \in S_i,\ i \in I, \tag{6}$$

$$\sum_{j \in N_t^-} \left(\sum_{s \in S_j} q_s^k y_j^s - q_t^k \right) f_{jt} \leq 0, \qquad t \in T,\ k \in K, \tag{7}$$

$$\sum_{s \in S_i} y_i^s = 1, \qquad i \in I, \tag{8}$$

$$\sum_{s \in S_i} y_i^s f_{ij} = f_{ij}, \qquad (i,j) \in A,\ i \in I, \tag{9}$$

$$\sum_{j \in N_i^+} y_i^s f_{ij} \leq y_i^s b_i, \qquad s \in S_i,\ i \in I, \tag{10}$$

$$f_{ij} \geq 0, \qquad (i,j) \in A, \tag{11}$$

$$0 \leq y_i^s \leq 1, \qquad s \in S_i,\ i \in I. \tag{12}$$

The formulation (3)–(12) generalizes the PQ-formulation (Tawarmalani and Sahinidis, 2002) for networks without directed paths connecting two pools. Constraints (4)–(5) impose the flow capacity constraints at all nodes, while (6) ensures that y_i^s is the proportion of the flow leaving pool i that originates from source s. The definition of y_i^s also implies (8). The desired quality at the terminals is achieved by (7). Constraints (9)–(10) are redundant RLT cuts (Sherali and Adams, 1999) that contribute to stronger relaxations. They are derived respectively by multiplying (8) by f_{ij}, and by multiplying (4) by y_i^s.

2.1 Traditional solution methods

Because of the bilinear constraints (6)–(7) and (9)–(10), the feasible region of (3)–(12) is generally non-convex. Traditional solution approaches to such problems are typically based upon linear relaxation, which is embedded into a branch-and-bound procedure. Linear relaxations for the pooling problem are constructed by replacing each occurrence of the bilinear terms with its convex and concave envelopes (Al-Khayyal and Falk, 1983; McCormick, 1976).

In the root node of a branch-and-bound algorithm, this relaxation is solved. In this way, the solution to the linear relaxation provides a lower bound on the

global minimum. Convergence can then be attained through partitioning of the domain within a branch and bound framework.

3 Discrete formulation

To linearize the bilinear term $y_i^s f_{ij}$, we discretize the proportion variable y_i^s into $n+1$ known points, i.e. we divide the interval $[0,1]$ to $n \geq 1$ intervals. For simplicity, we assume that the number of discretization points is equal for all i and s, and that the discretization points are uniformly distributed on $[0,1]$. However, the methodology suggested in this work does not rely on these assumptions.

3.1 Computing a set of discretized proportion vectors

Consider any pool $i \in I$ and the corresponding set of sources S_i that can feed the pool. By the suggested discretization of y_i^s for all $s \in S_i$, we get $(n+1)^{|S_i|}$ different combinations of discretized proportions. However, many of these violate (8). Let $\Omega_i = \left\{Y \in \mathbb{R}^{S_i} : nY^s \in \{0,1,\ldots,n\}, \sum_{s\in S_i} Y^s = 1\right\}$ be the set of discrete values of y_i that satisfy (8). For the purpose of simple notation, let the sources in S_i be identified by the integers $1,\ldots,|S_i|$.

For any $Y \in \Omega_i$, the components of nY define a unique composition of n into $|S_i|$ parts. As demonstrated by Knuth (2011, in Section 7.2.1.3), there is hence a bijection between Ω_i and the set of $(|S_i|-1)$-combinations of $\{1,\ldots,n+|S_i|-1\}$. Let any such combination be denoted $\left(a_1,\ldots,a_{|S_i|-1}\right)$, where $1 \leq a_1 < \cdots < a_{|S_i|-1} \leq n+|S_i|-1$. It follows from Section 7.2.1.3 in (Knuth, 2011) that the corresponding $Y \in \Omega_i$ can be written $Y^s = (a_s - a_{s-1} - 1)/n$ $(s = 1,\ldots,|S_i|)$, where $a_0 = 0$ and $a_{|S_i|} = n+|S_i|$. The above reference also suggests an algorithm for enumerating all $(|S_i|-1)$-combinations of $\{1,\ldots,n+|S_i|-1\}$, and thereby also the set Ω_i. This is outlined in Algorithm 1.

It is shown in (Knuth, 2011) that the while-loop of Algorithm 1 is executed $\frac{|S_i|-1}{n+1}|\Omega_i|$ times. The while-loop thus implies that enumerating $|\Omega_i|$ by use of Algorithm 1 does not run in $O(|\Omega_i|)$ time.

Algorithm 1: Discretization(i,n).

$\Omega_i \leftarrow \emptyset$
$a_s \leftarrow s, \forall\, s = 0, 1, \ldots, |S_i| - 1$
$a_{|S_i|} \leftarrow n + |S_i|$
repeat
  $Y^s \leftarrow (a_s - a_{s-1} - 1)/n, \forall\, s = 1, \ldots, |S_i|$
  $\Omega_i \leftarrow \Omega_i \cup \{Y\}$
  $s \leftarrow 1$
  while $a_s + 1 = a_{s+1}$ **do**
    $a_s \leftarrow s$
    $s \leftarrow s + 1$
  end
  if $s < |S_i|$ **then**
    $a_s \leftarrow a_s + 1$
  end
until $s = |S_i|$
return Ω_i

3.2 The discrete model defined in an extended graph

We introduce an extension of G where each pool i is replaced by a set I^i consisting of $|\Omega_i|$ duplications of i. Each new pool $j \in I^i$, corresponds to a unique $Y_j \in \Omega_i$ with components Y_j^s $s \in S_i$. We refer to these vectors as the discretized proportions. The set of pools in the extended network hence becomes $I_n = \cup_{i \in I} I^i$, and the extended network will be represented by the directed graph $G_n = (N_n, A_n)$, where $N_n = S \cup I_n \cup T$ and $A_n = A \cap (S \times T) \cup \{(j,l) : l \in I^i, (j,i) \in A\} \cup \{(l,j) : l \in I^i, (i,j) \in A\}$. For any $j \in I_n$, let $i(j)$ denote the parent pool in G. That is, $i(j)$ is the unique pool satisfying $j \in I^{i(j)}$. For completeness, let $i(j) = j$ for all $j \in S \cup T$.

For the selection of proportions at pool $i \in I$, define the binary variables p_j for each $j \in I^i$ such that,

$$p_j = \begin{cases} 1, & \text{if } y_i^s = Y_j^s \text{ for all } s \in S_i, \\ 0, & \text{otherwise,} \end{cases}$$

and impose the constraint $\sum_{j \in I^i} p_j = 1$ for each $i \in I$, to ensure compatibility

with the original problem. In the extended network, the flow can pass through at most one $j \in I^i$, leading to the constraints $\sum_{l \in N_j^+} f_{jl} \leq b_i p_j$ for all $j \in I^i$, where f now denotes flow in the extended network. The number of pools in the extended graph G_n increases exponentially with n. To reduce $|I_n|$, we identify pairs of pools $i, i' \in I$ such that $N_i^+ = N_{i'}^+$ and $N_i^- = N_{i'}^-$. For all such pairs, we do not introduce $I^{i'}$.

The MILP formulation approximating the continuous formulation (3)–(12), can hence be stated as follows

$$z(n) = \min \quad \sum_{(j,l) \in A_n} c_{i(j),i(l)} f_{jl} \tag{13}$$

$$\text{s.t.} \quad \sum_{l \in N_s^+} f_{sl} \leq b_s, \qquad s \in S, \tag{14}$$

$$\sum_{l \in N_j^+} f_{jl} \leq b_{i(j)} p_j, \qquad j \in I_n, \tag{15}$$

$$\sum_{l \in N_t^-} f_{lt} \leq b_t, \qquad t \in T, \tag{16}$$

$$\sum_{l \in N_j^-} Y_l^s f_{lj} - \sum_{l \in N_j^+} Y_j^s f_{jl} = 0, \qquad s \in S_{i(j)},\ j \in I_n, \tag{17}$$

$$\sum_{l \in N_t^-} \left(\sum_{s \in S_{i(l)}} q_s^k Y_l^s - q_t^k \right) f_{lt} \leq 0, \qquad t \in T,\ k \in K, \tag{18}$$

$$\sum_{j \in I^i} p_j = 1, \qquad i \in I, \tag{19}$$

$$p_j \in \{0,1\}, \qquad j \in I^i, i \in I, \tag{20}$$

$$f_{jl} \geq 0, \qquad (j,l) \in A_n. \tag{21}$$

Any feasible solution to (13)–(21) is a feasible solution to the original problem, and produces thereby an upper bound on z^*. The sequence $z(n)$ converges to z^* as $n \to \infty$, but even for instances of moderate size the computational burden represented by the MILP becomes prohibitively large for large values of n. However, with a coarse discretization, the optimal solution to (13)–(21) may be computable for instances where a global optimization algorithm based on a continuous formulation fails to converge within a reasonable time limit. In such instances, it is

relevant to compare the optimal MILP-solution to the best solution obtained by an interrupted global optimization procedure.

3.3 Example

To illustrate the network extension outlined above, consider the first instance in (Haverly, 1978), denoted Haverly1, depicted in Figure 1. Observe that node 4 is the unique pool in the network, and that $S_4 = \{1, 2\}$. Let $n = 2$, which implies that

$$(Y_j^s) = \begin{pmatrix} 0 & 1 \\ 1/2 & 1/2 \\ 1 & 0 \end{pmatrix}. \tag{22}$$

Each row of the matrix in (22) represents a possible combination of the flow proportions. Therefore, we replace pool 4 with 3 new pools ($I^4 = \{4, 5, 6\}$ and $I_2 = I^4$), and we change the numbering of the terminals accordingly.

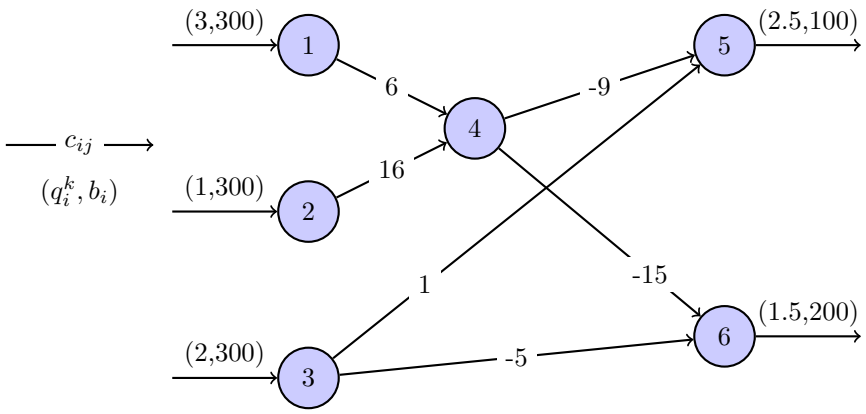


Figure 1: The Haverly1 pooling problem instance (Haverly, 1978).

The set I^4 has the same set of neighbors as the original pool in Figure 1. The new network structure for Haverly1 instance is shown in Figure 2.

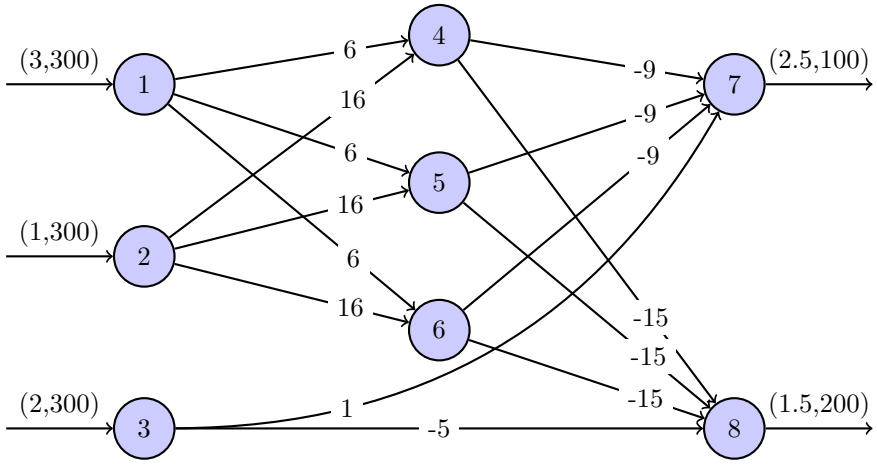


Figure 2: The discretized version of Haverly1 pooling problem instance with $n = 2$.

4 Computational experiments

For computational comparison of the discrete and the continuous formulations, we have used the 35 large-scale instances, with 15 arbitrary instances from (Alfaki and Haugland, 2012a) and 20 standard instances taken from (Alfaki and Haugland, 2012b). The instances are divided into six groups, three groups with arbitrary networks (arbC, arbD and arbE) and the other three groups (stdA, stdB and stdC) with standard instances. The instances in the former three groups can be downloaded from the web page `http://www.ii.uib.no/~mohammeda/gpooling/` and the other instances can be downloaded from `http://www.ii.uib.no/~mohammeda/spooling/`. Table 1 reports the network sizes and number of arcs range in the network for each group.

Computational experiments were conducted by submitting these instances using the formulation (3)–(12) to the global solver BARON (Sahinidis, 1996) version 1.8.5. The same instances were submitted to ILOG CPLEX version 10.2 using the discretized formulation (13)–(21) with $n = 1, 2, 4$. For both strategies, we set the time limit of each run to one CPU-hour, and set the relative optimality tolerance to 10^{-3}. Experiments reported here are conducted on a computer equipped

Table 1: Instance characteristics.

Group	#instances	Size of node and quality sets				#arcs range
		$\|S\|$	$\|I\|$	$\|T\|$	$\|K\|$	
arbC	5	8	6	6	4	57 – 82
arbD	5	12	10	8	5	114 – 166
arbE	5	10	10	15	12	181 – 248
stdA	10	20	10	15	24	171 – 407
stdB	6	35	17	21	34	384 – 1044
stdC	4	60	15	50	40	811 – 1451

with quad-core 3.00GHz processors where each group of four cores share 8GB of memory.

The results of the computational experiments are reported in Table 2. The first column gives the instance name, columns 2–3 report the lower and the upper bound provided by BARON with the continuous formulation. Column 4–5 give, for each value of n, the best feasible solution to the discrete model that CPLEX could find within the time limit. In instances where CPLEX could not prove optimality within the time limit, the best solution is written in parentheses. A stroke (—) in the table means that no feasible solution was found. For each instance, unless both of the formulations give the same solution, the best solution found is written in bold.

BARON computed the global optima for 14 instances. In the other hand, the feasible solutions for 9 instances with the discretized formulation are the true optimal solutions. Eight instances were solved to optimality by both of the formulations. Comparing the upper bounds (the feasible solutions) provided by both the continuous and the discretized formulations, we observe that the discrete formulation found the best upper bound in 21 instances out of 35. Even for $n = 1$, which means that all pools receive flow from at most one source, the best solution from the discrete model tends to outperform the best solution obtained by the continuous one. However, increasing the number of discretization points beyond 2 seems appropriate only in the smaller instances, and failed to produce feasible solutions in the remaining ones. For the more complicated instances, no better results are obtained by extending the search from solutions with no blending at

the pools ($n = 1$) to solutions allowing blending of at most two streams in equal proportions ($n = 2$).

Table 2: Comparison between continuous and discrete models for the pooling problem.

Inst.	Continuous model		Discrete model		
	lb	ub	$z(n = 1)$	$z(n = 2)$	$z(n = 4)$
arbC0	-1352.72	**-1352.72**	-1262.38	-1348.83	-1350.30
arbC1	-673.86	**-673.86**	-508.00	-615.50	-655.62
arbC2	-1716.62	**-1716.62**	-1688.69	-1705.81	(-1710.76)
arbC3	-1512.10	**-1512.10**	-1489.70	-1505.43	(-1508.92)
arbC4	-1071.81	-1071.81	-1071.81	-1071.81	-1071.81
arbD0	-1994.00	-1571.11	-1833.33	**-1911.35**	—
arbD1	-1356.51	-1356.51	-1346.54	-1356.51	—
arbD2	-2071.00	-2065.85	-2069.06	**-2070.16**	—
arbD3	-637.86	-637.86	-637.86	-637.86	—
arbD4	-1641.80	-1641.80	-1641.43	-1641.80	—
arbE0	-463.23	-463.23	-463.23	-462.23	—
arbE1	-556.00	-556.00	-556.00	-556.00	—
arbE2	-78.68	-78.68	-78.68	-78.68	—
arbE3	-891.25	-891.25	-891.25	-891.25	—
arbE4	-221.35	-221.35	-221.35	-221.35	—
stdA0	-37402.74	-5383.70	-31990.52	-34175.71	(**-34853.43**)
stdA1	-30362.74	-29276.56	-24590.16	-25179.84	(**-28389.31**)
stdA2	-23044.16	**-23044.16**	-19846.94	-20666.60	(-21795.71)
stdA3	-41113.10	-31258.05	-36233.75	-37116.64	(**-38624.98**)
stdA4	-42999.89	-8770.94	-38126.91	(**-39331.58**)	(-39345.90)
stdA5	-28257.75	-6369.59	-26447.07	(**-27008.30**)	(-26729.51)
stdA6	-42463.05	-9555.82	-41777.00	(**-42022.93**)	(-41829.91)
stdA7	-44682.25	-5762.08	-42582.29	(**-43309.48**)	(-42227.89)
stdA8	-30666.87	-6576.76	-30341.61	(**-30435.00**)	(-30265.99)
stdA9	-21933.99	-14059.98	-21887.77	(**-21891.96**)	(-21527.08)
stdB0	-45441.79	-9075.24	-40171.43	(**-41036.54**)	(-40600.32)
stdB1	-65468.81	-34069.43	-60720.54	(**-62445.97**)	(-61858.06)
stdB2	-56512.64	-11149.29	-53261.82	(**-53355.55**)	—
stdB3	-74050.47	-11469.84	(**-73572.52**)	(-73469.63)	—
stdB4	-59469.66	-13145.64	(**-59399.63**)	(-59233.59)	—
stdB5	-60696.36	-10313.90	(**-60080.85**)	(-59486.56)	—
stdC0	-98792.76	-2400.00	(-77517.74)	(**-79384.25**)	—
stdC1	-119006.17	-12114.75	(**-97290.27**)	(-91215.32)	—
stdC2	-135916.19	-6342.08	(**-117024.36**)	(-115594.77)	—
stdC3	-130315.02	-8770.86	(**-122570.51**)	(-114675.85)	—

5 Conclusion

In this paper, we have given a mixed integer programming model serving as an approximation to the pooling problem. The model makes no assumption about the network structure, and admits for example directed paths intersecting more than one pool. Computational experiments on a set of large-scale instances show that a discrete model is superior to its continuous ancestor, even when a very coarse discretization is applied. With a fine discretization, the model implies a large computational effort. To cope with this, a topic for future research is to develop an adaptive discretization rule. Computations can be saved if the number of discretization points can be kept small, while gradually focusing the search on solution sets of decreasing size.

Acknowledgements This research was sponsored by the Norwegian Research Council, Gassco, and Statoil under contract 175967/S30

References

Adhya, N., Tawarmalani, M., and Sahinidis, N.V. (1999). A Lagrangian approach to the pooling problem. *Industrial & Engineering Chemistry Research*, **38** (5), 1956–1972.

Al-Khayyal, F.A., and Falk, J.E. (1983). Jointly constrained biconvex programming. *Mathematics of Operations Research*, **8** (2), 273–286.

Alfaki, M., and Haugland, D. (2012a). A multi-commodity flow formulation for the generalized pooling problem. *Journal of Global Optimization*. doi:10.1007/s10898-012-9890-7

Alfaki, M., and Haugland, D. (2012b). Strong formulations for the pooling problem. *Journal of Global Optimization*. doi:10.1007/s10898-012-9875-6

Almutairi, H., and Elhedhli, S. (2009). A new Lagrangian approach to the pooling problem. *Journal of Global Optimization*, **45** (2), 237–257.

Audet, C., Brimberg, J., Hansen, P., Le Digabel, S., and Mladenović, N. (2004). Pooling problem: Alternate formulations and solution methods. *Management science*, **50** (6), 761–776.

Ben-Tal, A., Eiger, G., and Gershovitz, V. (1994). Global minimization by reducing the duality gap. *Mathematical Programming*, **63** (2), 193–212.

Faria, D.C., and Bagajewicz, M.J. (2008). A new approach for the design of multi-component water/wastewater networks. *Computer Aided Chemical Engineering*, **25**, 43–48.

Floudas, C.A., and Aggarwal, A. (1990). A decomposition strategy for global optimization search in the pooling problem. *Operations Research Journal On Computing*, **2** (3), 225–235.

Floudas, C.A., and Visweswaran, V. (1990). A global optimization algorithm (GOP) for certain classes of nonconvex NLPs–I. Theory. *Computers & chemical engineering*, **14** (12), 1397–1417.

Foulds, L.R., Haugland, D., and Jörnsten, K. (1992). A bilinear approach to the pooling problem. *Optimization*, **24** (1), 165–180.

Haverly, C.A. (1978). Studies of the behavior of recursion for the pooling problem. *ACM SIGMAP Bulletin*, **25**, 19–28.

Knuth, D.E. (2011). *The Art of Computer Programming*. Reading, Massachusetts: Addison-Wesley.

McCormick, G.P. (1976). Computability of global solutions to factorable nonconvex programs: part I - convex underestimating problems. *Mathematical Programming*, **10** (1), 147–175.

Pham, V. (2007). *A Global Optimization Approach to Pooling Problems in Refineries*. (Master's thesis, Department of Chemical Engineering, Texas A&M University, Texas, USA).

Pham, V., Laird, C., and El-Halwagi, M. (2009). Convex hull discretization approach to the global optimization of pooling problems. *Industrial & Engineering Chemistry Research*, **48** (4), 1973–1979.

Sahinidis, N.V. (1996). BARON: A general purpose global optimization software package. *Journal of Global Optimization*, **8** (2), 201–205.

Sherali, H.D., and Adams, W.P. (1999). *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Tawarmalani, M., and Sahinidis, N.V. (2002). *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Paper E

Computing feasible solutions to the pooling problem[2]

Mohammed Alfaki[1] and Dag Haugland[1]

[1]Department of Informatics, University of Bergen,
P.O. Box 7803, N-5020 Bergen, Norway.
mohammeda@ii.uib.no and dag@ii.uib.no
[2]Submitted to: *Annals of Operations Research*, 2011.

Computing feasible solutions to the pooling problem

Mohammed Alfaki[1] and Dag Haugland[1]

1 Department of Informatics, University of Bergen,
P.O. Box 7803, N-5020 Bergen, Norway.
mohammeda@ii.uib.no and dag@ii.uib.no

Abstract

Pooling and blending are important operations in petrochemical and agricultural industries with high potential economic value. For instance, transporting the natural gas from the production sources to the exit terminals is a complex process consisting of a blend of gases from different sources and many terminals. Constraints of particular importance, are restrictions regarding gas quality at terminals and the actual quality of the gas produced at sources. In many situations, intermediate pooling tanks is necessary, implying that an otherwise linear network flow model is transformed into a strongly $\mathcal{NP}$-hard problem recognized as the pooling problem. In this paper, we propose an algorithm for computing good feasible solutions to the pooling problem. In particular, we give a greedy construction method that in each iteration solves a pooling problem instance with only one terminal. Computational experiments demonstrate the merit of the method, which, in the hardest instances, give considerably better results than commercially available local optimizers.

Keywords Pooling problem · Heuristic methods · Network optimization

1 Introduction

Due to its industrial applications in petroleum refining, pipeline transportation of natural gas, wastewater treatment and general chemical engineering process

design (Bagajewicz, 2000; DeWitt et al., 1989; Quesada and Grossmann, 1995; Rømo et al., 2009), the pooling problem is an important global optimization problem that has been extensively studied over the last four decades. A problem instance is characterized by a network with, at the reception side, source streams with different qualities that can enter the network. Flow from the sources is fed into a limited number of available storage tanks (pools), where the entering flow is mixed to form intermediate blends with new qualities. The pool contents are subsequently used to form final blends at the terminals, where specific quality requirements to the blend are imposed by the market. The pooling problem is strongly $\mathcal{NP}$-hard (Alfaki and Haugland, 2012b), and is usually modeled as a bilinear programming model. Much of the research on the pooling problem is focused on mathematical formulations that exhibit favorable properties, and design of efficient computational methods for solving the problem.

1.1 Pooling problem formulations

In this section, we review different optimization models (or formulations) for the pooling problem and its extensions existing in the literature. The most straightforward formulation, commonly referred to as the P-formulation was proposed by Haverly (1978). This formulation depends solely on flow and quality variables. Ben-Tal et al. (1994) suggested the Q-formulation, involving variables that represent source proportions (fractions) of flow instead of explicit quality variables.

Tawarmalani and Sahinidis (2002) proposed the PQ-formulation, which is an extension of the Q-formulation, by adding two additional sets of constraints to strengthen the linear relaxation. These constraints were already derived by Quesada and Grossmann (1995) using the reformulation-linearization technique (RTL) to obtain global solutions to a bilinear problem with applications in process networks. Recently, Alfaki and Haugland (2012b) proposed a new formulation that incorporates, in addition to the source proportion variables present in the PQ-formulation, variables representing terminal proportions, and showed that the linear relaxation of their formulation is tighter than the corresponding relaxation of the PQ-formulation.

The formulations above apply to the *standard* pooling problem, where con-

nections from sources to pools, from sources to terminals, and from pools to terminals are the only connections allowed in the network. Audet et al. (2004) introduced the *generalized* pooling problem, where connections between pools are allowed as well. For this problem, Audet et al. (2004) suggested a hybrid formulation with variables representing source proportions, quality and flow. Alfaki and Haugland (2012a) proposed a multi-commodity flow formulation, which generalizes the PQ-formulation to the generalized pooling problem.

Meyer and Floudas (2006) and Misener and Floudas (2010) introduced an extension of the generalized pooling problem where the network topology is treated as decision variables, and the pools may function as treatment plants for reduction of contamination. There are also fixed charges for opening arcs and activation of pools. Consequently, binary variables are needed, and the model becomes a mixed integer nonlinear program (MINLP).

This extension has applications to the design of wastewater treatment networks. Another interesting extension was proposed by Misener and Floudas (2010), where they added, in addition to the standard pooling problem constraints, constraints involving environmental standards. The goal of this model is to maximize the profit of the reformulated gasoline. Li et al. (2011) presented a stochastic MINLP formulation for the generalized pooling problem with applications to natural gas network design and operations.

1.2 Local optimization methods

Local optimization techniques for the pooling problem are mainly based on improving heuristics, successive linear programming and Benders decomposition.

The first improving heuristic algorithm for the pooling problem is the recursive method proposed by Haverly (1978, 1979), and referred to as the alternating method (ALT) by Audet et al. (2004). The method starts by estimating and freezing the pool qualities, and then the resulting linear program (LP) is solved. The new quality values are calculated using the flow values from the solution of the LP. If all new quality values coincide with the old values, the procedure stops, otherwise a new LP using the new attribute qualities is constructed. Main (1993) observed that the recursive procedure is unstable in large instances. Audet et al.

(2004) suggested a variable neighborhood search heuristic based upon the ALT heuristic.

In the oil refining industry, the pooling problem has traditionally been approached by successive linear programming (SLP), or the method of approximate programming (Griffith and Stewart, 1961). This method starts with an initial guess of the variable values, approximates the bilinear terms using the Taylor's first order expansion at the initial guess, and then solves the resulting LP. The procedure is repeated with the LP solution as the new base of the Taylor expansion, until convergence to a fix point is obtained. Some improvements of the SLP are reported by Palacios-Gomez et al. (1982), Zhang et al. (1985), Baker and Lasdon (1985) and Sarker and Gunn (1997). Successful implementation of the SLP techniques in petrochemical industries are reported by Simon and Azma (1983), DeWitt et al. (1989), Rigby et al. (1995) and Amos et al. (1997).

1.3 Global optimization methods

During the last two decades, most of the solution techniques for the pooling problem have been focused on global optimization methods. These are either Lagrangian-based methods or variants of the branch-and-bound algorithm. The first Lagrangian-based method was proposed by Floudas and Aggarwal (1990) and Visweswaran and Floudas (1990), where the lower and upper bounds provided by solving relaxed dual and primal subproblems. Other techniques based on Lagrangian relaxation can be found in e.g. (Ben-Tal et al., 1994) and (Almutairi and Elhedhli, 2009). A branch-and-bound algorithm where the relaxation is constructed by the McCormick's convex and concave envelopes (McCormick, 1976) was first applied to the pooling problem in (Foulds et al., 1992). More advanced relaxation techniques, embedded in a branch-and-bound procedure, are employed by e.g. Liberti and Pantelides (2006) and Gounaris et al. (2009).

Global optimization algorithms are quite effective for instances of modest size. In larger instances, however, global optimizers fail to converge in reasonable time, while existing local optimizers depend largely on good initial guesses. The goal of the current research is to develop a fast method for identifying feasible solutions of low cost, without requiring that an initial solution is provided.

1.4 Contribution from the paper

In this paper we develop a greedy construction method for the pooling problem. The suggested method is designed to give good feasible solutions, particularly in large instances, and does not guarantee to find the optimal solution. In order to keep the work focused, we confine the study to the standard version of the problem, but it is straightforward to extend the proposed method to the generalized version.

The paper is organized as follows. In Section 2, we define the problem under study, and present the P- and PQ-formulations. Section 3 describes our proposed heuristic algorithm. In Section 4, we present computational results, and concluding remarks are made in the last section.

2 Problem definition and formulations

Consider a directed acyclic graph $D = (N, A)$ with node set N and arc set A, and with three disjoint sets of nodes S, I, and T where $N = S \cup I \cup T$. For any node $i \in N$, let $N_i^+ = \{j \in N : (i,j) \in A\}$ and $N_i^- = \{j \in N : (j,i) \in A\}$ denote the sets of out- and in-neighbors of i, respectively. We assume that S, and T are non-empty, and refer to them as the sets of *sources* and *terminals*, respectively. For simplicity, we assume in this work that $A \subseteq (S \times I) \cup (I \times T) \cup (S \times T)$, implying $N_s^- = \emptyset \; \forall s \in S$ and $N_t^+ = \emptyset \; \forall t \in T$. We refer to all nodes in I as *pools*.

For each node $i \in N$, define the node *capacity* b_i and for each arc $(i,j) \in A$ define the arc unit *cost* c_{ij}. Let K be the set of all *quality attributes*. With each $i \in S \cup T$, we define a real constant q_i^k for each $k \in K$. If $s \in S$, q_s^k is referred to as the *quality* of attribute k at that source, and if $t \in T$, q_t^k is referred to as the *quality bound* of attribute k at terminal $t \in T$.

Throughout this work, we make the *linear blending* assumption. That is, we define the quality at node $i \in I \cup T$ as a weighted average of the qualities at entering arcs, where the corresponding arc flows constitute the weights. The quality of any arc $(i,j) \in A$ is defined as the quality at node i.

Definition 1. The *pooling problem* is to assign flow values to all arcs $(i,j) \in A$ such that the flow capacity is respected at all nodes, the total flow entering a

pool equals the flow leaving the pool, the quality bounds at the terminals are respected, and such that the total flow cost is minimized.

2.1 The P-formulation

For notational simplicity, define $w_s^k = q_s^k$ for all $s \in S$ and all $k \in K$.

The P-formulation is derived by direct translation of Definition 1. By defining the flow variables $f \in \mathbb{R}_+^A$ and the quality variables $w \in \mathbb{R}^{I \times K}$, the P-formulation is given as:

$$[P] \qquad \min_{f,w} \sum_{(i,j)\in A} c_{ij} f_{ij} \tag{1}$$

$$\text{s.t.} \qquad \sum_{j \in N_s^+} f_{sj} \leq b_s, \qquad s \in S, \tag{2}$$

$$\sum_{j \in N_i^-} f_{ji} \leq b_i, \qquad i \in I \cup T, \tag{3}$$

$$\sum_{s \in N_i^-} f_{si} - \sum_{t \in N_i^+} f_{it} = 0, \qquad i \in I, \tag{4}$$

$$\sum_{s \in N_i^-} q_s^k f_{si} - \sum_{t \in N_i^+} w_i^k f_{it} = 0, \qquad i \in I,\ k \in K, \tag{5}$$

$$\sum_{j \in N_t^-} \left(w_j^k - q_t^k\right) f_{jt} \leq 0, \qquad t \in T,\ k \in K, \tag{6}$$

$$f_{ij} \geq 0, \qquad (i,j) \in A. \tag{7}$$

Constraints (2)–(3) impose the flow capacity constraints at all nodes. Constraints (4)–(5) enforce flow conservation and quality balance at pools. The quality requirements at the terminals is modeled by (6).

2.2 PQ-formulation

The PQ-formulation is obtained by replacing the quality variables by proportion variables. Define $y_i^s = \dfrac{f_{si}}{\sum_{t \in N_i^+} f_{it}}$, as the proportion of the flow at pool $i \in I$ coming from source $s \in N_i^-$. We then arrive at the formulation:

$$[PQ] \qquad \min_{f,y} \sum_{(i,j)\in A} c_{ij} f_{ij} \tag{8}$$

$$\text{s.t.} \qquad (2)\text{–}(3), \tag{9}$$

$$f_{si} - \sum_{t\in N_i^+} y_i^s f_{it} = 0, \qquad s \in N_i^-,\ i \in I, \tag{10}$$

$$\sum_{s\in N_t^-\cap S} q_s^k f_{st} + \sum_{i\in N_t^-\cap I} \sum_{s\in N_i^-} q_s^k y_i^s f_{it} \le q_t^k \sum_{j\in N_t^-} f_{jt}, \quad t \in T,\ k \in K, \tag{11}$$

$$\sum_{s\in N_i^-} y_i^s = 1, \qquad i \in I, \tag{12}$$

$$\sum_{s\in N_i^-} y_i^s f_{it} = f_{it}, \qquad i \in I,\ t \in N_i^+, \tag{13}$$

$$f_{ij} \ge 0, \qquad (i,j) \in A, \tag{14}$$

$$0 \le y_i^s \le 1, \qquad i \in I,\ s \in N_i^-. \tag{15}$$

Constraints (10) and (12) are implied by the definition of y_i^s, while constraint (11) gives the quality constraint at terminal $t \in T$. The latter is composed by direct flow from the sources in the first term, and flow from the pools in the second term. Constraint (13) is a redundant RLT cut (Sherali and Adams, 1999) that contributes to stronger relaxations. It is derived by multiplying (12) by f_{it}.

3 A method for constructing feasible solutions

Consider an arbitrary terminal $\tau \in T$, and define $D_\tau = (S_\tau, I_\tau, \{\tau\}, A_\tau)$ as the subgraph of D consisting of, in addition to τ, the sources S_τ, pools I_τ, and arcs A_τ, that are reachable by backtracking in D from τ. The pooling problem defined in graph D_τ can be expressed as a linear program (let $f_{ij} = 0$ if $(i,j) \in A \setminus A_\tau$):

$$\left[P_\tau^0\right] \qquad z_\tau^0 = \min_f \sum_{(i,j)\in A_\tau} c_{ij} f_{ij} \tag{16}$$

$$\text{s.t.} \qquad \sum_{j\in N_s^+} f_{sj} \le b_s, \qquad s \in S_\tau, \tag{17}$$

$$\sum_{j\in N_i^-} f_{ji} \le b_i, \qquad i \in I_\tau \cup \{\tau\}, \tag{18}$$

$$\sum_{s \in N_i^-} f_{si} - f_{i\tau} = 0, \qquad i \in I_\tau, \tag{19}$$

$$\sum_{s \in S_\tau} \sum_{j \in N_s^+} \left(q_s^k - q_\tau^k\right) f_{sj} \leq 0, \quad k \in K, \tag{20}$$

$$f_{ij} \geq 0, \qquad (i,j) \in A_\tau. \tag{21}$$

We observe that $\left[P_\tau^0\right]$ is a minimum cost flow problem with side constraints (20), and can be solved fast. Observe that if the constraint saying that flow can be sent to only one terminal had been added, the best would be to choose a terminal τ such that z_τ^0 is minimized.

Solving the above LPs constitutes the initial step of our construction method. We order the terminals $T = \left\{\tau^1, \ldots, \tau^{|T|}\right\}$ such that $z_{\tau^1} \leq \cdots \leq z_{\tau^{|T|}}$. In iteration $p = 1, \ldots, |T|$ of the heuristic, we augment a feasible flow vector (initially zero), by sending flow uniquely to terminal τ^p. The flow to τ^p is chosen to minimize the cost, subject to the constraints that the flows allocated in iterations $1, \ldots, p-1$ are preserved. Moreover, the flow from sources S_{τ^p} to τ^p, may affect the quality at $\tau^1, \ldots, \tau^p$, and therefore, we impose the constraints that the quality points at all terminals $\tau^1, \ldots, \tau^p$ are respected.

To define the optimization problem to be solved in iteration p in more precise terms, we introduce the shorthand notation $D^p = D_{\tau^p}$, $S^p = S_{\tau^p}$, $I^p = I_{\tau^p}$ and $A^p = A_{\tau^p}$. We also define $T^p = \left\{\tau^1, \ldots, \tau^p\right\}$, and let F^{p-1} be the vector of flow assignments accumulated in iterations $1, \ldots, p-1$. That is F^{p-1} is a feasible flow in the pooling problem, and considered as a fixed constant in iteration p.

In the current iteration, we aim to find an optimal augmentation of the feasible flow F^{p-1} by sending flow in the subgraph D^p. For pools $i \in I^p$, we define the decision variables w_i^k ($k \in K$) as the new quality of pool i. This implies the bilinear quality balance constraints $w_i^k \left(f_{i\tau^p} + \sum_{t \in T^{p-1}} F_{it}^{p-1}\right) = \sum_{s \in N_i^-} q_s^k \left(f_{si} + F_{si}^{p-1}\right)$ for all $i \in I^p$ and $k \in K$. The quality constraints at $t \in T^{p-1}$ are now given as the linear inequalities $\sum_{j \in N_t^-} \left(w_j^k - q_t^k\right) F_{jt}^{p-1} \leq 0$. In contrast, the corresponding constraint at the new terminal τ^p becomes bilinear since both the pool quality and the entering flow are variables: $\sum_{j \in N_{\tau^p}^-} \left(w_j^k - q_{\tau^p}^k\right) f_{j\tau} \leq 0$. However, only $\left|N_{\tau^p}^- \cap I\right| |K|$ bilinear terms are

involved in this subproblem ($[P^p]$), and fast solution algorithms are likely to be found.

$$[P^p] \qquad z^p = \min_{f,w} \sum_{(i,j)\in A^p} c_{ij} f_{ij} \tag{22}$$

$$\text{s.t.} \qquad \sum_{j\in N_s^+} \left(f_{sj} + F_{sj}^{p-1}\right) \le b_s, \qquad s \in S^p, \tag{23}$$

$$\sum_{j\in N_i^-} \left(f_{ji} + F_{ji}^{p-1}\right) \le b_i, \qquad i \in I^p \cup \{\tau^p\}, \tag{24}$$

$$\sum_{s\in N_i^-} f_{si} - f_{i\tau^p} = 0, \qquad i \in I^p, \tag{25}$$

$$w_i^k \left(f_{i\tau^p} + \sum_{t\in N_i^+ \cap T^{p-1}} F_{it}^{p-1}\right) = \sum_{s\in N_i^-} q_s^k \left(f_{si} + F_{si}^{p-1}\right), \quad i \in I^p,\ k \in K, \tag{26}$$

$$\sum_{j\in N_t^-} \left(w_j^k - q_t^k\right) F_{jt}^{p-1} \le 0, \qquad t \in T^{p-1},\ k \in K, \tag{27}$$

$$\sum_{j\in N_{\tau^p}^-} \left(w_j^k - q_{\tau^p}^k\right) f_{j\tau^p} \le 0, \qquad k \in K, \tag{28}$$

$$f_{ij} \ge 0, \qquad (i,j) \in A^p. \tag{29}$$

The construction method is shown in Algorithm 1. Whenever profitable flow can be sent to the new terminal, we make the corresponding flow augmentation. Otherwise, the terminal is ignored, and will receive zero flow.

The above construction algorithm is explained with reference to the P-formulation. Details for adopting it to the PQ-formulation are omitted, as this is accomplished in an analogous manner.

4 Computational experiments

For the purpose of evaluating the potential of the proposed heuristic, we have conducted computational experiments where we compare it to state-of-the-art local optimization techniques. These experiments were carried out on 20 large-scale standard pooling problem instances introduced in (Alfaki and Haugland, 2012b).

Algorithm 1: The construction method.

Solve $\left[P_\tau^0\right]$ for all $\tau \in T$
Define $\left\{\tau^1, \ldots, \tau^{|T|}\right\}$ such that $z^0_{\tau^1} \leq \cdots \leq z^0_{\tau^{|T|}}$
$p \leftarrow 0$, $T^0 \leftarrow \emptyset$, $F^0 \leftarrow 0$
repeat
    $p \leftarrow p + 1$
    Solve $[P^p]$ to obtain the optimal solution (z^p, f^p, w^p)
    if $z^p < 0$ **then**
        $F^p \leftarrow F^{p-1} + f^p$
        $T^p \leftarrow T^{p-1} \cup \{\tau^p\}$
    end
until $p = |T|$

All experiments were run on a computer with 3.00GHz Intel(R) processor and 8GB RAM. All models and instances were formulated in GAMS, and we have used version 11.1.1 of CPLEX, version 5.51 of MINOS and version 8.1.5 of BARON. For all experiments, we set the time limit of each run to one CPU-hour.

The test instances are shown in Table 1, where an instance identifier is given in the first column. Columns 2-6 give numbers of sources, pools, terminals, quality attributes and arcs for each instance. All instances can be downloaded in GAMS-format from http://www.ii.uib.no/~mohammeda/spooling.

In the first experiment, we have implemented our heuristic algorithm described in Section 3 with two different formulations (i) with the P-formulation and (ii) with the PQ-formulation. The algorithm is coded in GAMS modeling language. In the initial step, we solve the LP model (16)–(21) using CPLEX and in iterations $1, \ldots, |T|$, the bilinear subproblem model (16)–(21) (or the corresponding model in case of PQ-formulation) is solved by use of BARON as global solver. Since the algorithm runs for T iterations, we have divided the time limit among the iteration, i.e. we have set, in each iteration, a time limit of $3600/|T|$ seconds, and if the subproblem of the current iteration is solved before this time expires, the excess time is transferred to the next iteration. That is, iteration p is interrupted after $\frac{3600p}{|T|}$ CPU-seconds, and the best solution to $[P^p]$ is considered as an optimal one.

In the second experiment, we have tested two different types of multi-start lo-

Table 1: Size of the test instances.

Instance	$\|S\|$	$\|I\|$	$\|T\|$	$\|K\|$	$\|A\|$
A0	20	10	15	24	171
A1	20	10	15	24	179
A2	20	10	15	24	192
A3	20	10	15	24	218
A4	20	10	15	24	248
A5	20	10	15	24	277
A6	20	10	15	24	281
A7	20	10	15	24	325
A8	20	10	15	24	365
A9	20	10	15	24	407
B0	35	17	21	34	384
B1	35	17	21	34	515
B2	35	17	21	34	646
B3	35	17	21	34	790
B4	35	17	21	34	943
B5	35	17	21	34	1044
C0	60	15	50	40	811
C1	60	15	50	40	1070
C2	60	15	50	40	1278
C3	60	15	50	40	1451

cal optimization methods using the Q-formulation. We have run MINOS starting from several random initial solutions, where the value of each variable is drawn from the uniform distribution on the interval between its bounds. In each iteration, we keep the best solution found so far. This process is repeated until the maximum allowed CPU time is expired. We have also applied the multi-start heuristic in combination with the range reduction technique available in BARON. This is done until global optimality is proved or the maximum CPU time is elapsed.

Note that we have used the Q-formulation in the above mentioned multi-start heuristics rather than the P-formulation, because the Q-formulation has shown better performance when fed into local optimizers.

Table 2: Comparison between the construction heuristic, the multi-start heuristics, and the bounds from a global solver.

Inst.	Our heuristic with P		Our heuristic with PQ		Multi-start MINOS		M-S BARON	Global BARON	
	time	ub	time	ub	#pts	ub	ub	lb (time)	ub
A0	961.95	-18046.23	5.96	-32289.61	462	-34354.40	-25193.25	-36411.10	**-35812.33**
A1	1291.01	-23398.40	3505.52	-21146.23	1055	-23823.40	-10203.80	-29896.91	**-29276.56**
A2	520.07	-16828.37	9.34	-16314.79	665	-21769.98	-15820.54	(83.67)	**-23042.04**
A3	963.22	-32766.89	7.80	-32486.48	335	-37526.44	-29808.13	-40324.91	**-39446.54**
A4	1006.64	-35084.35	66.84	-33781.86	253	-10463.49	**-37310.68**	-42659.52	-33687.13
A5	2641.94	-20758.56	19.39	-13540.35	206	-983.64	-20777.85	-28257.75	**-24015.54**
A6	4.94	-29384.53	28.07	-32002.94	229	-3123.86	**-41281.11**	-42463.05	-37074.67
A7	966.00	-30608.72	216.29	-30958.66	168	-21269.97	**-41239.94**	-44682.25	-38074.67
A8	1070.09	-22965.36	573.24	-18317.72	203	-19434.11	-24184.25	-30666.87	**-28795.26**
A9	995.10	-15540.22	731.18	-15650.97	155	-10795.75	-0.00	(484.22)	**-21912.35**
B0	3429.23	**-32158.57**	1003.58	-29969.58	146	-14190.70	-19712.56	-45179.93	-20802.12
B1	2645.75	-48944.04	1361.20	-47505.28	159	–	-19974.64	-65048.03	**-50055.21**
B2	874.98	**-33388.34**	976.12	-31914.81	147	–	-10675.76	-56083.32	-18567.64
B3	544.12	-54752.44	1224.05	**-55964.56**	126	–	-351.39	-74050.47	-18327.40
B4	1235.88	**-37413.97**	375.96	-24873.08	105	–	-0.00	-59469.66	-3711.84
B5	1256.18	-35221.99	2174.79	**-36333.67**	123	–	0.00	-60696.36	-10653.72
C0	2275.42	**-66213.11**	1587.35	-61448.53	277	–	0.00	-96256.99	-15197.45
C1	2360.97	-80219.43	1507.25	**-79739.78**	380	–	0.00	-117856.16	-25196.93
C2	1313.91	**-68571.44**	486.67	-46049.96	322	–	-0.00	-135546.50	-7497.52
C3	1429.11	**-79446.89**	1181.58	-78767.32	233	–	-0.00	-130315.02	-7163.54

We compare all results to the best solution and the lower bound on the objective function value reported in (Alfaki and Haugland, 2012b). These results were obtained by use of the STP-formulation, the purpose of which is to provide sharp lower bounds to be exploited by global optimization algorithms.

Table 2 shows the results of the experiments described above, where instance identifiers are given in the first column. Columns 2-3 report the CPU-time (in seconds) and the objective function value of the best solution found using our heuristic with the P-formulation, respectively. Columns 4-5 report the same information when we use our heuristic with the PQ-formulation. Columns 6-7 give the total number of starting points and the objective function value of the best solution found by MINOS with multi-start. The best objective function value provided by BARON as multi-start local optimizer is reported in column 8. The only stopping criterion for the multi-start heuristics is the time limit (one CPU-hour of time) and therefore we have not reported the CPU-time for the multi-start heuristics. Columns 9-10 report the lower and upper bounds on the global minimum cost provided by BARON as global solver after one CPU-hour of computations. Under the lower bound column, if the solver managed to solve the instance within the time limit, the required CPU-time is given in parentheses. A stroke (–) in the table means that no feasible solution was found by the algorithm.

We observe that in the smallest instances, A0–A9, multi-start local optimization is mainly superior to our heuristic. Also, the global optimization approach reported in (Alfaki and Haugland, 2012b) provides better results after one CPU-hour than does the suggested construction heuristic in these instances. The two instances for which the global optimum is known (A2 and A9) could not be solved to optimality of any of the heuristic methods, although multi-start with MINOS came fairly close in instance A2.

When moving to instances of larger scale, the results are considerably more in the favor of our construction method. In instances B4–B5 and C0–C3, the multi-start methods found at best the trivial zero solution, whereas our method, regardless of the formulation used, found non-trivial feasible solutions. The corresponding cost is significantly lower than the cost of the solutions reported in

(Alfaki and Haugland, 2012b), and thus the solution produced by the heuristic is, to the best of our knowledge, the best solution known to date.

5 Concluding remarks and further work

In this paper, we have proposed a construction heuristic for the pooling problem. The heuristic considers a sequence of subgraphs, each of which contains a single terminal, and an associated bilinear program for optimizing the flow to the terminal. The optimal solution serves as a feasible augmentation of the total flow accumulated so far. Experimental results on 20 large-scale standard pooling problem instances indicate that, in large instances, our heuristic algorithm outperforms multi-start local optimization techniques provided by commercially available software.

In our future plans, we will investigate a similar heuristic that instead of constructing the iterations on the set of terminals we can consider the set of sources. Extending the idea for other extensions of the pooling problem is also a possible direction for future work.

Acknowledgements This research was sponsored by the Norwegian Research Council, Gassco, and Statoil under contract 175967/S30

References

Alfaki, M., and Haugland, D. (2012a). A multi-commodity flow formulation for the generalized pooling problem. *Journal of Global Optimization*. doi:10.1007/s10898-012-9890-7

Alfaki, M., and Haugland, D. (2012b). Strong formulations for the pooling problem. *Journal of Global Optimization*. doi:10.1007/s10898-012-9875-6

Almutairi, H., and Elhedhli, S. (2009). A new Lagrangian approach to the pooling problem. *Journal of Global Optimization*, **45** (2), 237–257.

Amos, F., Rönnqvist, M., and Gill, G. (1997). Modelling the pooling problem at the New Zealand Refining Company. *Journal of the Operational Research Society*, **48** (8), 767–778.

Audet, C., Brimberg, J., Hansen, P., Le Digabel, S., and Mladenović, N. (2004). Pooling problem: Alternate formulations and solution methods. *Management science*, **50** (6), 761–776.

Bagajewicz, M. (2000). A review of recent design procedures for water networks in refineries and process plants. *Computers & Chemical Engineering*, **24** (9–10), 2093–2113.

Baker, T.E., and Lasdon, L.S. (1985). Successive linear programming at Exxon. *Management Science*, **31** (3), 264–274.

Ben-Tal, A., Eiger, G., and Gershovitz, V. (1994). Global minimization by reducing the duality gap. *Mathematical Programming*, **63** (2), 193–212.

DeWitt, C.W., Lasdon, L.S., Waren, A.D., Brenner, D.A., and Melhem, S.A. (1989). OMEGA: An improved gasoline blending system for Texaco. *Interfaces*, **19** (1), 85–101.

Floudas, C.A., and Aggarwal, A. (1990). A decomposition strategy for global optimization search in the pooling problem. *Operations Research Journal On Computing*, **2** (3), 225–235.

Foulds, L.R., Haugland, D., and Jörnsten, K. (1992). A bilinear approach to the pooling problem. *Optimization*, **24** (1), 165–180.

Gounaris, C.E., Misener, R., and Floudas, C.A. (2009). Computational comparison of piecewise–linear relaxation for pooling problems. *Industrial & Engineering Chemistry Research*, **48** (12), 5742–5766.

Griffith, R.E., and Stewart, R.A. (1961). A nonlinear programming technique for the optimization of continuous processing systems. *Management Science*, **7**, 379–392.

Haverly, C.A. (1978). Studies of the behavior of recursion for the pooling problem. *ACM SIGMAP Bulletin*, **25**, 19–28.

Haverly, C.A. (1979). Behavior of recursion model-more studies. *ACM SIGMAP Bulletin*, **26**, 22–28.

Li, X., Armagan, E., Tomasgard, A., and Barton, P.I. (2011). Stochastic pooling problem for natural gas production network design and operation under uncertainty. *AIChE Journal*, **57** (8), 2120–2135.

Liberti, L., and Pantelides, C.C. (2006). An exact reformulation algorithm for large nonconvex NLPs involving bilinear terms. *Journal of Global Optimization*, **36** (2), 161–189.

Main, R.A. (1993). Large recursion models: Practical aspects of recursion techniques. In T. Ciriani and R. Leachman (Eds.), *Optimization in Industry: Mathematical Programming and Modeling Techniques in Practice* (pp. 241–249). New York, USA: John Wiley & Sons Ltd.

McCormick, G.P. (1976). Computability of global solutions to factorable nonconvex programs: part I - convex underestimating problems. *Mathematical Programming*, **10** (1), 147–175.

Meyer, C.A., and Floudas, C.A. (2006). Global optimization of a combinatorially complex generalized pooling problem. *AIChE Journal*, **52** (3), 1027–1037.

Misener, R., and Floudas, C.A. (2010). Global optimization of large-scale generalized pooling problems: Quadratically constrained MINLP models. *Industrial & Engineering Chemistry Research*, **49** (11), 5424–5438.

Palacios-Gomez, F., Lasdon, L.S., and Engquist, M. (1982). Nonlinear optimization by successive linear programming. *Management Science*, **28** (10), 1106–1120.

Quesada, I., and Grossmann, I.E. (1995). Global optimization of bilinear process networks with multi-component flows. *Computers & Chemical Engineering*, **19** (12), 1219–1242.

Rigby, B., Lasdon, L.S., and Waren, A.D. (1995). The evolution of Texaco's blending systems: from OMEGA to StarBlend. *Interfaces*, 64–83.

Rømo, F., Tomasgard, A., Hellemo, L., Fodstad, M., Eidesen, B.H., and Pedersen, B. (2009). Optimizing the Norwegian natural gas production and transport. *Interfaces*, **39** (1), 46–56.

Sarker, R.A., and Gunn, E.A. (1997). A simple SLP algorithm for solving a class of nonlinear programs. *European Journal of Operational Research*, **101** (1), 140–154.

Sherali, H.D., and Adams, W.P. (1999). *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Simon, J.D., and Azma, H.M. (1983). Exxon experience with large scale linear and nonlinear programming applications. *Computers & Chemical Engineering*, **7** (5), 605–614.

Tawarmalani, M., and Sahinidis, N.V. (2002). *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Visweswaran, V., and Floudas, C.A. (1990). A global optimization algorithm (GOP) for certain classes of nonconvex NLPs–II. Application of theory and test problems. *Computers & chemical engineering*, **14** (12), 1419–1434.

Zhang, J.H., Kim, N.H., and Lasdon, L. (1985). An improved successive linear programming algorithm. *Management Science*, **31** (10), 1312–1331.