Solution Approaches for the Single Row Facility Layout Problem based on Semidefinite Programming

Masterarbeit

von

Jan Schwiddessen

angefertigt am

Lehrstuhl Management Science
Fakultät Wirtschaftswissenschaften
Technische Universität Dortmund

unter Anleitung von

JProf. Dr. Anja Fischer

Dortmund, Mai 2020

Abstract

The *single row facility layout problem* (SRFLP) is the strongly $\mathcal{NP}$-hard optimization problem of arranging n facilities of given lengths on a straight line, while minimizing a weighted sum of distances between all facility pairs. In this thesis, we review different lower bounding techniques for SRFLP and demonstrate that, despite their practical challenges, strong semidefinite relaxations are among the most promising solution approaches. To this end, we propose a new semidefinite approach for SRFLP and tighten the existing semidefinite relaxations for the first time with pentagonal and other inequalities. Additionally, we make use of an algorithmic method that has never been applied to semidefinite relaxations of SRFLP before. Using our novel approach, we significantly reduce the best known duality gaps for all benchmark instances from the literature with up to 100 facilities, and report optimal solutions for almost all instances with up to 81 facilities. Moreover, our approach turns out to be much faster than any other exact approach for SRFLP. Finally, we address a branch-and-bound approach which uses our semidefinite bounds and provide some useful tools such as primal heuristics and branching rules.

Notation

$\mathbb{R}$	space of real numbers
$\mathbb{R}_+$	space of nonnegative real numbers
$\mathbb{R}^n$	space of real n-dimensional column vectors
$\mathbb{R}^n_+$	space of nonnegative real n-dimensional column vectors
$\mathcal{S}_n$	space of $n \times n$ symmetric matrices
$\mathcal{S}_n^+$	space of $n \times n$ positive semidefinite matrices
$\mathcal{S}_n^-$	space of $n \times n$ negative semidefinite matrices
$A \succeq 0$	matrix A is (symmetric) positive semidefinite
$A \succ 0$	matrix A is (symmetric) positive definite
$A \preceq 0$	matrix A is (symmetric) negative semidefinite
min	minimum, minimize
max	maximum, maximize
inf	infimum
sup	supremum
e	vector of all ones of appropriate dimension
$\operatorname{tr}(A)$	trace of $A \in \mathcal{S}_n$
$\langle A, B \rangle$	inner product in $\mathcal{S}_n$, $\langle A, B \rangle = \operatorname{tr}(BA)$
$\mathcal{A}(X)$	$= [\langle A_1, X \rangle, \ldots, \langle A_m, X \rangle]^\top$ for given $A_i \in \mathcal{S}_n$
$\mathcal{A}^\top(y)$	$= \sum_{i=1}^m y_i A_i$
$\operatorname{rank}(A)$	rank of A
$\operatorname{diag}(A)$	$\operatorname{diag}(A) = [a_{11}, \ldots, a_{nn}]^\top$
$\operatorname{conv}(\cdot)$	convex hull operator
$[n]$	$= \{1, \ldots, n\}$
Π_n	set of permutations of $[n]$
$\mathcal{E}$	elliptope
$\mathcal{M}$	metric polytope
$\mathcal{P}_C$	cut polytope
$\mathcal{P}_{LOP}$	linear ordering polytope
$\mathcal{P}_{BTW}$	betweenness polytope
$\mathcal{P}_{QO}$	quadratic ordering polytope
$\mathcal{P}_{LQO}$	linear-quadratic ordering polytope
`SRFLP`	single row facility layout problem

Contents

Introduction

Semidefinite programming is a relatively new branch of convex optimization and has been studied extensively in the last decades. One of its major areas of application is combinatorial optimization, where it is used to approximate $\mathcal{NP}$-hard optimization problems by semidefinite relaxations. These relaxations can often be strengthened in a canonical way by adding some kind of generic inequalities. The obtained bounds are typically much stronger than those of linear programming techniques when applied to problems that have some kind of quadratic characteristic. Especially for problems that are hard to handle with standard linear programming based approaches, semidefinite relaxations play to their strengths. Although all well-posed semidefinite programs have nice theoretical properties and can be solved with arbitrary precision within polynomial time, their practical solution can impose major challenges for interesting problem sizes and is still a very active research area. In order to efficiently apply semidefinite relaxations to large-scale problems, special customized approaches have to be designed. A particular class of combinatorial optimization problems for which semidefinite relaxations are known to be among the best approaches, are the so-called ordering problems (see e.g. [41, 44]). One specific problem among this class is the *single row facility layout problem* (`SRFLP`).

Problem statement. Suppose that we are given n one-dimensional facilities with positive integer lengths ℓ_i, $1 \leq i \leq n$, and integer weights c_{ij}, $1 \leq i < j \leq n$, between all pairs of facilities. The single row facility layout problem asks to arrange the n facilities on a straight line in a non-overlapping way such that the total weighted sum of distances between all pairs of facilities is minimized. Here, the distance d_{ij} of two facilities i and j is measured with respect to their respective centroids. Due to the nonnegative lengths and weights, it is sufficient to only consider layouts without gaps between adjacent facilities. Hence, a feasible layout or ordering of the facilities can be represented by an element $\pi \in \Pi_n$, where Π_n denotes the set of all permutations of $[n] := \{1, \ldots, n\}$. With this notation, `SRFLP` can be formulated as

$$\min_{\pi \in \Pi_n} \sum_{\substack{i,j \in [n] \\ i<j}} c_{ij} d_{ij}^{\pi},$$

where d_{ij}^{π} denotes the center-to-center distance of i and j with respect to the given layout π. For example, let $n = 3$ with $(\ell_1, \ell_2, \ell_3) = (3, 5, 6)$ and $(c_{12}, c_{13}, c_{23}) = (4, 8, 9)$ be given. An optimal ordering for this particular instance is $\pi^* = (1, 3, 2)$ which is illustrated in Figure 1. We have $d_{12}^{\pi^*} = 10$, $d_{13}^{\pi^*} = 4.5$, $d_{23}^{\pi^*} = 5.5$, and thus, the optimal value is $4 \cdot 10 + 8 \cdot 4.5 + 9 \cdot 5.5 = 125.5$. Obviously, `SRFLP` admits a certain kind of symmetry in the sense that each feasible

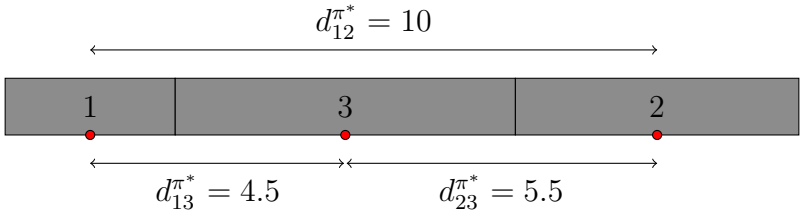

Figure 1: Optimal ordering of a SRFLP instance with $n = 3$, $(\ell_1, \ell_2, \ell_3) = (3, 5, 6)$ and $(c_{12}, c_{13}, c_{23}) = (4, 8, 9)$. The optimal value is $4 \cdot 10 + 8 \cdot 4.5 + 9 \cdot 5.5 = 125.5$.

vector of distances corresponds to two different permutations which yield the same objective value.

Applications and related problems. The single row facility layout problem, also known as the *one-dimensional space allocation problem*, was first considered by Simmons [78] in 1969. Since then, many practical situations that can be modeled by SRFLP have been identified in the literature. These include the arrangement of rooms on one side of a corridor in hospitals, supermarkets or office buildings [78], the assignment of disk cylinders to files and the optimal storage of equipment such as books on shelves [71]. Further applications include the assignment of airplanes to gates at an airport terminal [81] and the ordering of machines in flexible manufacturing systems, where automated guided vehicles travel between manufacturing cells on a straight line [37]. A common feature of these applications is that some objects, of possibly different dimensions, have to be arranged such that the (expected) total cost of communication between them is minimized. For this reason, the weights c_{ij} are often also called 'transportation costs', 'average daily traffic', 'transmission intensities', 'traffic intensities', 'communication costs' or 'connectivities'.

The SRFLP is also closely related to many other interesting combinatorial problems. First, it belongs to the general class of facility layout problems (FLPs), more precisely to the row facility layout problems which also include some extensions of SRFLP such as the double row facility layout problem (see e.g. [11]). Nonetheless, the SRFLP is by far the most studied of these variants. Moreover, if all facilities have unit length, we obtain the so-called single row equidistant facility layout problem (SREFLP) which is a special case of the notorious quadratic assignment problem (see [42]). Furthermore, if additionally all weights c_{ij} have values in $\{0, 1\}$, SRFLP reduces to the well-known minimum linear arrangement problem (see e.g. [32]) which is already $\mathcal{NP}$-hard in the strong sense [24]. Hence, SRFLP and SREFLP are also strongly $\mathcal{NP}$-hard. On the other hand, a generalization of SRFLP is given by the weighted betweenness problem, which itself is again a special case of the quadratic ordering problem (see [41]). As a result, investigating SRFLP can also yield valuable algorithmic approaches for some of the numerous related problems. Due to its computational complexity, plenty of heuristic approaches for SRFLP were proposed, especially in recent years (see e.g. [2, 13, 20, 22, 30, 46–49, 53, 54, 67, 69, 70, 74, 75, 80]). Certainly, these approaches do not provide any lower bounds for the optimal value of SRFLP.

Existing lower bounding techniques. Exact proposed approaches for SRFLP include a combinatorial branch-and-bound algorithm by Simmons [78, 79], dynamic programming approaches by Karp & Held [45] and Picard & Queyranne [71] and a nonlinear model by Heragu & Kusiak [38]. However, it turned out that appropriate approaches based on linear programming techniques and semidefinite relaxations are much more efficient in practice. Several mixed integer formulations for SRFLP were proposed by Love & Wong [60] and Amaral [1,3]. However, due to weak linear relaxations only instances with $n \leq 18$ could be solved with these approaches within several hours. The first polyhedral study of the so-called distance polytope for SRFLP by Amaral & Letchford [5] led to a considerable improvement on the lower bounds, and instances with up to 30 facilities were then solved. The current best approach for SRFLP based on linear programming was proposed by Amaral [4]. Using binary betweenness variables within a sole cutting plane algorithm, instances with up to $n = 35$ were solved.

Matrix-based relaxations have proven to be another promising way to obtain strong lower bounds for SRFLP. The first semidefinite relaxation for SRFLP was proposed by Anjos et al. [6], providing the first non-trivial lower bounds for instances with up to 80 facilities. A slightly weaker semidefinite relaxation was proposed by Anjos & Yen [12], which could be solved much faster and produced lower bounds for $n = 100$ which are provably only a few percentage points away from the optimum. However, the obtained lower bounds of these basic relaxations were too weak to solve any instance of considerable size. Therefore, Anjos & Vanelli [8, 10] improved the relaxations by adding the so-called triangle inequalities in a cutting plane approach. At that time, they solved many prior unsolved instances with $n \leq 30$ to optimality within dozens of hours of computing time. The current leading approach for SRFLP was proposed by Hungerländer and Rendl [44] and is based on a semidefinite relaxation that is particularly designed for quadratic ordering problems. Their approach involves an appropriate combination of optimization methods and handles the majority of constraints in a Lagrangian fashion. Hungerländer and Rendl [43, 44] solved instances with up to 42 facilities and greatly improved the best known optimality gaps for all unsolved instances with $n \leq 100$. Their approach is also much faster than any other exact approach for $n \geq 20$.

Research aims. On the one hand, there was seemingly no improvement on exact approaches for SRFLP in the last few years, and the best known optimality gaps for some well-known instances in the literature are still barely smaller than 2%. On the other hand, plenty of heuristic approaches were proposed, proving that the interest in SRFLP is still very high. The main goal of this thesis is to level up the existing semidefinite approaches for SRFLP and to solve considerably larger instances to global optimality while also significantly reducing the best known duality gaps for very large instances. A secondary goal is to show that strong semidefinite relaxations are the most promising way to tackle large SRFLP instances. However, they require further problem-dependent knowledge and well-suited algorithmic approaches.

In order to achieve these goals, we first review many existing lower bounding techniques for SRFLP, including linear and semidefinite approaches, and discuss their strengths and limitations. We then develop a novel semidefinite approach for SRFLP which involves a theoretically stronger relaxation and a solution method that was not applied to SRFLP before.

We strengthen the existing semidefinite relaxations by adding a well-suited subclass of the so-called pentagonal inequalities and also consider more generic inequalities for which we propose heuristic separation routines. We also propose primal heuristics that extract feasible layouts from approximate solutions of our semidefinite relaxations. The core of our novel approach for SRFLP is that we use the nonstandard semidefinite bounds introduced in [62] which heavily rely on Lagrangian duality. They are weaker than the usual semidefinite bounds, but their tightness can be controlled by a real parameter. Moreover, these bounds can be optimized via a quasi-Newton method, while their evaluation only requires a partial eigendecomposition of a matrix of order $\mathcal{O}(n^2)$. In order to obtain strong lower bounds for SRFLP, we use the algorithmic framework presented in [52] as a template and develop a bounding procedure particularly tailored for SRFLP. We also discuss the usage of our new bounds in a branch-and-bound algorithm.

Contribution. We improve the existing semidefinite relaxations for SRFLP and provide the first computational results concerning pentagonal and other inequalities. We also use a solution approach for these relaxations that has never been applied to SRFLP before. Our proposed approach greatly outperforms all existing exact approaches. We solve many instances with up to 81 facilities to global optimality for the first time. In fact, only a few benchmark instances from the literature with $n \leq 100$ remain unsolved. For all unsolved instances we significantly reduce the best known duality gaps, leading to nearly optimal lower bounds, even for $n = 100$. Additionally, our approach turns out to be much faster than any other approach in the literature for $n \geq 20$. It also computes the lower bounds of the current leading approach by Hungerländer & Rendl [44] in much less time and finds better feasible solutions. Finally, we present some important tools such as primal heuristics and branching rules, which possibly could be used within a branch-and-bound algorithm to solve even larger instances.

Structure. This thesis is divided into seven chapters: Chapter 1 briefly assembles some important aspects of semidefinite matrices, semidefinite programming and Lagrangian relaxation. In Chapter 2, we review various approaches based on linear programming for SRFLP and discuss their practical limitations. In Chapter 3, we present a bivalent quadratic formulation for SRFLP and deduce the already existing semidefinite relaxations. We also comment on the performance of direct solution methods for these relaxations. Chapter 4 is concerned with more suitable practical methods that use Lagrangian duality and can deal with large-scale problems. We present the current leading approach by Hungerländer & Rendl [44], as well as the motivation of our proposed approach. In Chapter 5, we present our novel approach in more detail. First, we propose a new stronger semidefinite relaxation together with some heuristic separation routines for certain classes of inequalities. We then present primal heuristics and provide an overview of our practical implementation. The chapter is concluded with a discussion on a branch-and-bound approach that makes use of our proposed semidefinite bounds. In Chapter 6, we provide extensive computational results on our approach and compare it to other exact approaches. Finally, some conclusions and possible directions of future work are given in Chapter 7.

Chapter 1

Preliminaries

In this chapter, we briefly introduce some important mathematical concepts that are used throughout this thesis.

1.1 Semidefinite matrices

In this thesis, all considered matrices are assumed to be symmetric. A matrix $A \in \mathbb{R}^{n\times n}$ is symmetric if $A = A^\top$ and the vector space of $n \times n$ real symmetric matrices is denoted by $\mathcal{S}_n$. The natural inner product between two elements $A = (a_{ij}), B = (b_{ij}) \in \mathcal{S}_n$ is given by

$$\langle A, B\rangle \coloneqq \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}b_{ij} = \operatorname{tr}(BA).$$

The associated norm

$$\|A\|_F \coloneqq \sqrt{\langle A, A\rangle}$$

is the so-called Frobenius norm.

Definition 1.1.1
A matrix $A \in \mathcal{S}_n$ is positive semidefinite ($A \in \mathcal{S}_n^+$, $A \succeq 0$) if

$$x^\top Ax \geq 0 \quad \forall x \in \mathbb{R}^n,$$

and negative semidefinite ($A \in \mathcal{S}_n^-$, $A \preceq 0$) if

$$x^\top Ax \leq 0 \quad \forall x \in \mathbb{R}^n.$$

Moreover, $A \in \mathcal{S}_n$ is positive definite ($A \succ 0$) if

$$x^\top Ax > 0 \quad \forall x \in \mathbb{R}^n \setminus \{0\}.$$

It is well-known that $\mathcal{S}_n^+$ is a full-dimensional, closed, pointed, self-dual, convex cone in $\mathbb{R}^{\binom{n+1}{2}}$ (see e.g. [33]). The projections (with respect to the Frobenius norm) of a matrix $A \in \mathcal{S}_n$ onto $\mathcal{S}_n^+$ and its polar cone $\mathcal{S}_n^-$ are given by

$$A_+ \coloneqq \operatorname*{argmin}_{X\succeq 0} \|X - A\|_F \qquad \text{and} \qquad A_- \coloneqq \operatorname*{argmin}_{X\preceq 0} \|X - A\|_F.$$

Due to Moreau's theorem (see [40]), these projections are characterized by

$$\langle A_+, A_- \rangle = 0 \qquad \text{and} \qquad A = A_+ + A_-. \tag{1.1}$$

Moreover, an explicit formula for these projections is known (see e.g. [39]). For this, let $A = \sum_i \lambda_i v_i v_i^\top$ be a spectral decomposition of A with eigenvalues λ_i and corresponding eigenvectors v_i which are assumed to be pairwise orthogonal and of unit length (see [62]). Then

$$A_+ = \sum_{\lambda_i > 0} \lambda_i v_i v_i^\top \qquad \text{and} \qquad A_- = \sum_{\lambda_i < 0} \lambda_i v_i v_i^\top. \tag{1.2}$$

Combining (1.1) and (1.2) yields

$$(-A)_- = -A_+ \tag{1.3}$$

for any $A \in \mathcal{S}_n$.

1.2 Semidefinite programming

Semidefinite programming (SDP) is an extension of linear programming (LP), where a linear function is optimized over the cone of semidefinite matrices subject to linear constraints, i.e., the vector variable $x \in \mathbb{R}^n_+$ is replaced by a matrix variable $X \in \mathcal{S}^+_n$. Moreover, SDP includes LP as a special case, namely when the matrix variable X is restricted to be diagonal (see e.g. [33]).

Let C and $A_1, \ldots, A_m$ be matrices in $\mathcal{S}_n$ and $b \in \mathbb{R}^m$. A (primal) semidefinite program in standard notation can then be written as the optimization problem

$$\text{(PSDP)} \qquad \begin{array}{ll} \min & \langle C, X \rangle \\ \text{s.t.} & \mathcal{A}(X) = b \\ & X \in \mathcal{S}^+_n, \end{array}$$

where $\mathcal{A}\colon \mathcal{S}_n \to \mathbb{R}^m$ is a linear operator of the form

$$\mathcal{A}(X) = \begin{pmatrix} \langle A_1, X \rangle \\ \vdots \\ \langle A_m, X \rangle \end{pmatrix}. \tag{1.4}$$

Since the semidefinite cone $\mathcal{S}^+_n$ is not polyhedral and has a nonlinear boundary, (PSDP) is a convex nonlinear optimization problem.

1.2.1 Duality theory

The duality theory in semidefinite programming requires more care than in linear programming, since strong duality does not hold in general. Nonetheless, strong duality is guaranteed to hold if a Slater constraint qualification is satisfied. In this case, semidefinite programs can theoretically and practically be solved by interior-point methods (IPMs) to any desired precision in polynomial time (see e.g. [33]).

The Lagrangian dual problem of (PSDP) is given by

$$\text{(DSDP)} \qquad \begin{array}{rl} \max & \langle b, y\rangle \\ \text{s.t.} & \mathcal{A}^\top(y) + Z = C \\ & Z \in \mathcal{S}_n^+ \\ & y \in \mathbb{R}^m, \end{array}$$

where $\mathcal{A}^\top : \mathbb{R}^m \to \mathcal{S}_n$ with

$$\mathcal{A}^\top(y) = \sum_{i=1}^m y_i A_i$$

is the adjoint operator to (1.4). Despite its appearance, (DSDP) is again a semidefinite program (see e.g. [33]). As usual, weak duality holds for semidefinite programming, i.e.,

$$\langle b, y\rangle \leq \langle C, X\rangle$$

for any triple (X, y, Z) that is feasible for the primal-dual pair consisting of (PSDP) and (DSDP). To state the strong duality theorem, we say that (PSDP) is strictly feasible if it admits a feasible point X that satisfies $X \succ 0$. Analogously, we say that (DSDP) is strictly feasible if it admits a feasible pair (y, Z) that satisfies $Z \succ 0$.

Theorem 1.2.1 (Strong duality, see e.g. [41])
Let a primal-dual pair of (PSDP) and (DSDP) be given and let

$$\begin{array}{rcll} p^* & = & \inf\{\langle C, X\rangle : \mathcal{A}(X) = b,\ X \succeq 0\} & \text{and} \\ d^* & = & \sup\left\{\langle b, y\rangle : \mathcal{A}^\top(y) + Z = C,\ Z \succeq 0\right\}. & \end{array}$$

Assume that $d^* < \infty$ (respectively $p^* > -\infty$) and assume that (DSDP) (respectively (PSDP)) is strictly feasible. Then $p^* = d^*$ and this value is attained for (PSDP) (respectively (DSDP)).

1.2.2 Applications in combinatorial optimization

A famous application of semidefinite programming is the max-cut problem. With a suitable choice of the matrix $C \in \mathcal{S}_n$, the max-cut problem can be formulated as (see e.g. [33])

$$\begin{array}{rl} \max & \langle C, X\rangle \\ \text{s.t.} & X \in \mathcal{P}_C, \end{array}$$

where $\mathcal{P}_C$ denotes the *cut polytope*

$$\mathcal{P}_C \coloneqq \operatorname{conv}\left\{xx^\top : x \in \{-1, 1\}^n\right\}.$$

It is well-known (see [55]) that $\mathcal{P}_C$ is contained in the so-called *elliptope*

$$\mathcal{E} \coloneqq \left\{X \in \mathbb{R}^{n\times n} \colon X \succeq 0,\ X_{ii} = 1,\ i = 1, \ldots, n\right\}.$$

Hence, an upper bound for the max-cut problem is obtained by solving the semidefinite program

$$\begin{array}{rl} \max & \langle C, X\rangle \\ \text{s.t.} & X \in \mathcal{E}. \end{array}$$

Analogously, SDP relaxations can be derived in a canonical way for any optimization problem that can be formulated over $\mathcal{P}_C$ with additional linear constraints in the matrix variable X. That is, a semidefinite relaxation of

$$\begin{array}{ll} \max & \langle C, X \rangle \\ \text{s.t.} & \mathcal{A}(X) \leq a \\ & \mathcal{B}(X) = b \\ & X \in \mathcal{P}_C \end{array}$$

is given by

$$\begin{array}{ll} \max & \langle C, X \rangle \\ \text{s.t.} & \mathcal{A}(X) \leq a \\ & \mathcal{B}(X) = b \\ & X \in \mathcal{E}. \end{array}$$

1.3 Lagrangian relaxation

Beyond the scope of convex optimization, Lagrangian duality is an essential methodology in combinatorial optimization. The universal technique of Lagrangian relaxation can be used to create dual bounds for the optimal value of any optimization problem. In practice, optimization problems are often difficult to solve directly if they involve a complex structure of the feasible set or simply have too many constraints. This also concerns convex optimization problems such as semidefinite programs which theoretically can be solved by interior-point methods in polynomial time. However, the cost to do so may be very high for large-scale problems. For the following short description of Lagrangian relaxation, we refer to [25, 57].

To illustrate the idea of Lagrangian relaxation, we consider the rather general optimization problem

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & a(x) \leq 0 \\ & b(x) = 0 \\ & x \in \mathcal{X}, \end{array} \tag{1.5}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$, $f\colon \mathbb{R}^n \to \mathbb{R}$, $a\colon \mathbb{R}^n \to \mathbb{R}^m$ and $b\colon \mathbb{R}^n \to \mathbb{R}^p$. We assume that f can comparatively easy be minimized over $\mathcal{X}$, but that all additional equality or inequality constraints make problem (1.5) much harder to solve. The Lagrangian relaxation dualizes these complicating constraints and lifts them into the objective function. For this purpose, we introduce the so-called *Lagrangian*

$$\mathcal{L}(x; \lambda, \mu) \coloneqq f(x) + \sum_{i=1}^{m} \lambda_i a_i(x) + \sum_{j=1}^{p} \mu_j b_j(x) = f(x) + \lambda^\top a(x) + \mu^\top b(x),$$

which is a function of the primal variable $x \in \mathcal{X}$ and of the dual variables $\lambda \in \mathbb{R}^m_+$, $\mu \in \mathbb{R}^p$. Moreover, for given dual variables λ and μ, we define the *dual function*

$$\theta(\lambda, \mu) \coloneqq \inf_{x \in \mathcal{X}} \mathcal{L}(x; \lambda, \mu).$$

It is now easy to see that *weak duality* holds, i.e., we have

$$\theta(\lambda, \mu) \leq f(x) \quad \text{for all } x \text{ feasible in (1.5) and all } \lambda \in \mathbb{R}^m_+,\ \mu \in \mathbb{R}^p.$$

Hence, each feasible pair of dual variables (λ, μ) yields a lower bound $\theta(\lambda, \mu)$ for the optimal value of problem (1.5). To find the best possible such lower bound, we have to solve the so-called *dual problem*

$$\begin{array}{ll} \sup & \theta(\lambda, \mu) \\ \text{s.t.} & \lambda \in \mathbb{R}^m_+ \\ & \mu \in \mathbb{R}^p. \end{array} \tag{1.6}$$

It is well-known that the domain

$$\operatorname{dom}(\theta) \coloneqq \left\{(\lambda, \mu) \in \mathbb{R}^m_+ \times \mathbb{R}^p \colon \theta(\lambda, \mu) > -\infty\right\}$$

is a convex set and that the function $\theta \colon \operatorname{dom}(\theta) \to \mathbb{R}$ is concave (see e.g. [25]). However, the dual function θ is often not differentiable on $\operatorname{dom}(\theta)$, and thus, methods of nonsmooth convex optimization have to be used to optimize the dual problem (1.6) in this case.

Chapter 2

Linear Programming based Approaches for SRFLP

Linear programming (LP) is one of the most essential and powerful tools in mathematical optimization. Therefore, many LP-based approaches were suggested to tackle SRFLP in the last decades (see e.g. [1, 3–5, 38, 60]). All of these approaches provide lower bounds for the optimal value of SRFLP. Hence, they can be used to estimate the quality of any ordering that is obtained by a heuristic, or even to prove the global optimality of such an ordering. In contrast, the combinatorial branch-and-bound algorithm proposed in [78] and the dynamic programming approach suggested in [71] do not yield lower bounds until they terminate and output an optimal solution.

In the next sections, we present three specific approaches that outline the main ideas and research directions to tackle SRFLP with linear programming techniques. Especially the approach presented in [4], which uses so-called betweennness variables, is of great importance for the later introduced semidefinite relaxations in Chapter 3 and their further enhancement in Chapter 5. We close this chapter by discussing the practical usefulness of the presented LP-based approaches and by pointing out their strengths and limitations.

2.1 Intuitive mixed 0-1 LP formulations

An intuitive mixed 0-1 LP formulation for SRFLP was proposed by Love & Wong [60] and then later by Heragu & Kusiak [38] in a different setting. The model, they came up with, used $\mathcal{O}(n^2)$ binary variables, $\mathcal{O}(n^2)$ continuous variables and $\mathcal{O}(n^2)$ constraints.

Their model involved binary variables $u = (u_{ij}) \in \{0,1\}^{n(n-1)}$, $i \neq j$, with the meaning

$$u_{ij} = \begin{cases} 1, & \text{if facility } i \text{ lies to the left of facility } j \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

In order to link the binary variables (2.1) with each other, they also used position variables p_i for all $i \in [n]$. To express the objective function, additional distance variables d_{ij} for all pairs $i, j \in [n],\ i < j$, were required.

Let $M \coloneqq \sum_{i\in[n]} \ell_i$, then the formulation of Love & Wong [60] is similar to

$$\min \quad \sum_{\substack{i,j\in[n]\\ i<j}} c_{ij}d_{ij} \tag{2.2}$$

$$\text{s.t.} \quad u_{ij} + u_{ji} = 1, \qquad i,j \in [n],\ i<j, \tag{2.3}$$

$$d_{ij} \geq p_i - p_j, \qquad i,j \in [n],\ i<j, \tag{2.4}$$

$$d_{ij} \geq p_j - p_i, \qquad i,j \in [n],\ i<j, \tag{2.5}$$

$$p_i + (\ell_i + \ell_j)/2 \leq p_j + M(1-u_{ij}), \quad i,j \in [n],\ i\neq j, \tag{2.6}$$

$$\ell_i/2 \leq p_i \leq M - \ell_i/2, \qquad i \in [n], \tag{2.7}$$

$$d_{ij} \geq 0, \qquad i,j \in [n],\ i<j, \tag{2.8}$$

$$u_{i,j} \in \{0,1\}, \qquad i,j \in [n],\ i\neq j. \tag{2.9}$$

Equations (2.3) ensure that the relative positioning of two facilities i and j is well-defined, i.e., either i lies to the left of j or vice versa. These equations could also be used to eliminate half of the variables. They are linked with the position variables p_i by inequalities (2.6) that ensure a non-overlapping ordering of all facilities. Given concrete values of the binary variables and the position variables, a lower bound for the distance d_{ij} between two facilities i and j is computed with the help of (2.4) and (2.5). Since we are minimizing in (2.2), there is always an optimal solution for which $d_{ij} = |p_i - p_j|$ for all $i,j \in [n]$, $i<j$, holds. The remaining constraints (2.7), (2.8) and (2.9) correspond to bound constraints or integrality conditions.

Sadly, the above model has a very weak linear relaxation. To see this, let $d_{ij} = 0$ for all $i,j \in [n]$, $i<j$, $p_i = \max\{\ell_j \colon j \in [n]\}$ for all $i \in [n]$ and $u_{ij} = \frac{1}{2}$ for all $i,j \in [n]$, $i \neq j$. All constraints are then satisfied and the resulting lower bound is 0. This is why only instances with $n \leq 11$ were solved using this approach.

The variables (2.1) are often called *linear ordering variables* and the convex hull of all their incidence vectors forms the so-called *linear ordering polytope* $\mathcal{P}_{LOP}$ (see e.g. [29]). Since the linear ordering problem (LOP) is a special case of SRFLP (see [58]) and can be formulated over $\mathcal{P}_{LOP}$, it is a promising idea to exploit the knowledge of $\mathcal{P}_{LOP}$. To this end, Amaral [1] showed that SRFLP can be formulated over a projection of $\mathcal{P}_{LOP}$. Using the basic relaxation

$$\begin{aligned}\mathcal{B} \coloneqq \{u \in \mathbb{R}^{n(n-1)} \colon\ & u_{ij} + u_{ji} = 1, \quad i,j \in [n],\ i<j,\\ & u_{ij} + u_{jk} + u_{ki} \leq 2, \quad i,j,k \in [n],\ |\{i,j,k\}| = 3,\\ & u_{ij} \geq 0, \quad i,j \in [n],\ i\neq j\}\end{aligned}$$

of $\mathcal{P}_{LOP}$ (see [29]), he extended the model of Love & Wong [60] by adding more inequalities. The resulting model had $\mathcal{O}(n^3)$ constraints and its linear relaxation yielded an improved lower bound of

$$\frac{1}{2} \sum_{\substack{i,j\in[n]\\ i<j}} c_{ij}\left(\ell_i + \ell_j\right),$$

see [5]. Instances with up to $n = 15$ were then solved within an hour.

Another extension suggested by Amaral [3] uses quadratic expressions of the linear ordering variables u_{ij} to replace the distance variables d_{ij} in the objective function. Amaral [3] presented a linearization for this idea involving $\mathcal{O}(n^3)$ binary variables, $\mathcal{O}(n^3)$ continuous variables and $\mathcal{O}(n^3)$ constraints. The associated lower bounds are of a more sophisticated type (see [5]) and instances with $n \leq 18$ could be solved within a few hours.

2.2 The distance polytope for SRFLP

An in depth study on the polyhedral properties of SRFLP was done by Amaral & Letchford [5]. Since only the distances d_{ij} between all pairs of facilities are required to calculate the objective value of a valid ordering, they directly represented a permutation $\pi \in \Pi_n$ by its implicitly given distance vector d^{π}_{ij}. For given $n \geq 2$ and positive integer lengths $\ell \in \mathbb{Z}^n_{\geq 1}$, they defined the *distance polytope* $\mathcal{P}(n, \ell)$ as the convex hull of all valid distance vectors d, i.e.,

$$\mathcal{P}(n,\ell) \coloneqq \operatorname{conv}\left\{ d \in \mathbb{R}_+^{\binom{n}{2}} \colon \exists \pi \in \Pi_n \text{ with } d_{ij} = d^{\pi}_{ij}\ \forall i,j \in [n],\ i<j \right\}.$$

Due to symmetry, $\mathcal{P}(n, \ell)$ has $n!/2$ vertices. These vertices are not necessarily integral, since the distance between two facilities is measured with respect to their centroids. Nonetheless, an integral polytope can easily be obtained by a translation [5].

Amaral & Letchford [5] showed that $\mathcal{P}(n, \ell)$ has dimension $\binom{n}{2} - 1$, i.e., it is not full-dimensional. Its affine hull is described by the equation

$$\sum_{\substack{i,j\in[n]\\ i<j}} \ell_i \ell_j d_{ij} = \frac{1}{6}\left[\left(\sum_{i\in[n]} \ell_i \right)^3 - \sum_{i\in[n]} {\ell_i}^3 \right]. \tag{2.10}$$

An exponential class of valid and facet defining inequalities for $\mathcal{P}(n, \ell)$ can immediately be derived from equation (2.10), which they called 'clique' inequalities. Moreover, they proved that $\mathcal{P}(n, \ell)$ is contained in the well-known *cut cone* CC_n (see e.g. [21]). As a result, all known valid inequalities for CC_n are also valid for $\mathcal{P}(n, \ell)$. Amaral & Letchford [5] used many of these inequalities as cutting planes in a branch-and-cut algorithm. More precisely, they considered the so-called hypermetric inequalities which include the triangle inequalities

$$d_{ij} - d_{ik} - d_{jk} \leq 0, \quad i,j,k \in [n],\ |\{i,j,k\}| = 3,\ i<j,$$

as a special case, the pure negative-type inequalities and the rounded positive semidefinite inequalities. Amaral & Letchford [5] presented detailed conditions under which certain classes of inequalities induce facets of $\mathcal{P}(n, \ell)$ and how they can be separated heuristically. Their starting relaxation only included the trivial lower bounds $d_{ij} \geq (\ell_i + \ell_j)/2$ and the implicit equation (2.10). While feasibility could be achieved by adding binary variables to the model, Amaral & Letchford [5] presented a specific branching rule to avoid this. This branching rule amounts to enforcing certain triangle inequalities to hold with equality at each node of the branch-and-bound tree.

Exploring thousands of nodes, they successfully solved many instances with up to 30 facilities and also produced lower bounds for larger instances with up to $n = 100$ using a pure cutting plane approach. However, the running times were measured in several dozens of hours for instances with $n = 30$. Additionally, the lower bounds for larger instances were relatively weak when compared to those of already existing semidefinite approaches, e.g., the semidefinite approach in [12].

2.3 Betweenness-based approach

We now look at the probably most successful approach for SRFLP based on linear programming. Amaral [4] presented a quite unintuitive modeling which implicitly takes the quadratic nature of SRFLP more into account than all previous approaches. The key idea is that we only need to know which facilities are *between* a pair (i, k) in order to calculate their distance d_{ik}. For this, let $\pi = (\pi_1, \dots, \pi_n)$ be a valid ordering and let B^π_{ik} the set of facilities between i and k with respect to π. Then we have

$$d^\pi_{ik} = \frac{\ell_i + \ell_k}{2} + \sum_{j \in B^\pi_{ik}} \ell_j. \tag{2.11}$$

Motivated by this observation, Amaral [4] introduced the so-called *betweenness variables* b_{ijk}, $i, j, k \in [n]$, $|\{i, j, k\}| = 3$, $i < k$, with the meaning

$$b_{ijk} = \begin{cases} 1, & j \text{ lies between } i \text{ and } k \\ 0, & \text{otherwise.} \end{cases} \tag{2.12}$$

Each permutation $\pi \in \Pi_n$ corresponds to a vector $b \in \{0, 1\}^{n(n-1)(n-2)/2}$ collecting the betweenness variables defined in (2.12). In fact, the betweenness variables can be expressed in quadratic terms of the linear ordering variables (2.1) by

$$b_{ijk} = u_{ik}u_{kj} + u_{jk}u_{ki}.$$

Amaral [4] then defined the betweenness polytope

$$\mathcal{P}_{BTW} \coloneqq \operatorname{conv}\left\{b \in \{0, 1\}^{n(n-1)(n-2)/2} \colon b \text{ represents a permutation } \pi \in \Pi_n\right\}$$

and presented a partial linear description of $\mathcal{P}_{BTW}$ to optimize over it.

For each triple (i, j, k) of facilities, exactly one of these has to be located between the other two, i.e., the equations

$$b_{ijk} + b_{ikj} + b_{jik} = 1, \quad i, j, k \in [n],\ i < j < k, \tag{2.13}$$

are valid for $\mathcal{P}_{BTW}$. Indeed, the smallest linear subspace that contains $\mathcal{P}_{BTW}$ is exactly described by equations (2.13) (see [76]). If we consider yet another facility $h \in [n]$, h can only lie between at most two of the three pairs (i, j), (i, k) and (j, k). Thus, the inequalities

$$b_{ihj} + b_{ihk} + b_{jhk} \leq 2, \quad i, j, k, h \in [n],\ |\{i, j, k, h\}| = 4,\ i < j < k, \tag{2.14}$$

are valid for $\mathcal{P}_{BTW}$. It turns out that h can not be located between exactly one of these pairs, only between zero or two. This can be expressed by the inequalities

$$\begin{aligned} -b_{ihj} + b_{ihk} + b_{jhk} &\geq 0, \quad i,j,k,h \in [n],\ |\{i,j,k,h\}| = 4,\ i<j<k,\\ +b_{ihj} - b_{ihk} + b_{jhk} &\geq 0, \quad i,j,k,h \in [n],\ |\{i,j,k,h\}| = 4,\ i<j<k,\\ +b_{ihj} + b_{ihk} - b_{jhk} &\geq 0, \quad i,j,k,h \in [n],\ |\{i,j,k,h\}| = 4,\ i<j<k. \end{aligned} \tag{2.15}$$

Amaral [4] generalized the inequalities (2.15) by presenting a family of more sophisticated inequalities containing them: let $\beta \leq n$ be an even integer and let $R \subseteq [n]$ be such that $|R| = \beta$. Then for each $r \in R$, and for any partition (S,T) of $R \setminus \{r\}$ such that $|S| = \beta/2$, the inequality

$$\sum_{p,q\in S,\, p<q} b_{prq} + \sum_{p,q\in T,\, p<q} b_{prq} \leq \sum_{p\in S,\, q\in T,\, p<q} b_{prq} \tag{2.16}$$

is valid for $\mathcal{P}_{BTW}$.

With the help of (2.11), a suitable objective function for formulating SRFLP over $\mathcal{P}_{BTW}$ is given by

$$\min_{b\in\mathcal{P}_{BTW}} \sum_{\substack{i,j\in[n]\\ i<j}} c_{ij} \sum_{k\in[n]\setminus\{i,j\}} \ell_k b_{ikj} + \sum_{\substack{i,j\in[n]\\ i<j}} c_{ij}\frac{\ell_i+\ell_j}{2}. \tag{2.17}$$

A lower bound for (2.17) is then obtained by solving the linear relaxation that includes (2.13)–(2.15) and the box constraints

$$0 \leq b_{ijk} \leq 1, \quad i,j,k \in [n],\ |\{i,j,k\}| = 3,\ i<k. \tag{2.18}$$

In order to get stronger lower bounds, Amaral [4] suggested to use the inequalities (2.16) with $\beta = 6$ as cutting planes. The resulting lower bounds turned out to be the optimal value for all instances considered in [4]. Many instances with up to 35 facilities were solved to global optimality. While often several hours of computing time are required for instances with $n = 35$, the betweenness approach seems to outperform all other approaches in terms of shorter running times for smaller instances with $n \leq 20$. We will discuss the betweenness approach in more detail in Section 2.4.

Some theoretical results are known supporting the strong computational results obtained by solving only the linear relaxation. First, Amaral [4] showed that

$$\begin{aligned} \min\quad & \sum_{\substack{i,j\in[n]\\ i<j}} c_{ij} \sum_{k\in[n]\setminus\{i,j\}} \ell_k b_{ikj} + \sum_{\substack{i,j\in[n]\\ i<j}} c_{ij}\frac{\ell_i+\ell_j}{2} \\ \text{s.t.}\quad & b_{ijk} + b_{ikj} + b_{jik} = 1, && i,j,k \in [n],\ i<j<k,\\ & b_{ihj} + b_{ihk} + b_{jhk} \leq 2, && i,j,k,h \in [n],\ |\{i,j,k,h\}| = 4,\ i<j<k,\\ & -b_{ihj} + b_{ihk} + b_{jhk} \geq 0, && i,j,k,h \in [n],\ |\{i,j,k,h\}| = 4,\ i<j<k,\\ & +b_{ihj} - b_{ihk} + b_{jhk} \geq 0, && i,j,k,h \in [n],\ |\{i,j,k,h\}| = 4,\ i<j<k,\\ & +b_{ihj} + b_{ihk} - b_{jhk} \geq 0, && i,j,k,h \in [n],\ |\{i,j,k,h\}| = 4,\ i<j<k,\\ & b_{ijk} \in \{0,1\}, && i,j,k \in [n],\ |\{i,j,k\}| = 3,\ i<k, \end{aligned} \tag{BTW}$$

is indeed a formulation for SRFLP. The equations (2.13) can be used to eliminate some of the betweenness variables. The resulting projection of $\mathcal{P}_{BTW}$ and $\mathcal{P}_{BTW}$ itself have dimension $n(n-1)(n-2)/3$ (see [76]). Second, the betweenness polytope $\mathcal{P}_{BTW}$ induces a face of the cut polytope $\mathcal{P}_C$ [83]. Additionally, the inequalities (2.14) and (2.15) correspond to the so-called triangle inequalities that are facet-defining for $\mathcal{P}_C$. In fact, the inequalities (2.16), which include (2.15) for $\beta = 4$, are proven to be facet-defining for $\mathcal{P}_{BTW}$, whereas the inequalities (2.14) do not induce a facet of $\mathcal{P}_{BTW}$ in general (see [76]). Moreover, the family of inequalities (2.16) corresponds to the so-called clique inequalities which are facet-defining for $\mathcal{P}_C$. As a result, $\mathcal{P}_{BTW}$ preserves some of the nice structural properties inherited from the cut polytope $\mathcal{P}_C$.

2.4 Discussion on LP-based approaches for SRFLP

The intuitive mixed 0-1 LP formulation of Love & Wong [60] clearly is not useful for practical purposes and even fails on solving very small instances with $n \leq 10$ in reasonable time. Although Amaral [1,3] improved the formulation by using known results for the linear ordering polytope $\mathcal{P}_{LOP}$, these attempts are still dominated by the approaches presented in Section 2.2 and Section 2.3. Since the associate lower bounds are very weak, excessive memory and computing time requirements arise. The branch-and-bound algorithm effectively degenerates to complete enumeration. Already struggling on small instances, we get no acceptable lower bounds for instances with $n \geq 20$.

The study on the distance polytope $\mathcal{P}(n, \ell)$ by Amaral & Letchford [5] and their resulting practical approach were a huge progress compared to the previous mixed 0-1 LP formulations. Instances with up to $n = 20$ could be solved in a few seconds using this approach. The gaps for all 'classical' SRFLP instances in the literature with $n \leq 100$ remained smaller than 8% using only a cutting plane approach. Even though the corresponding lower bounds were computed with a time limit of one day, this was the first time linear programming was applied to such large instances. Many of the the valid inequalities for $\mathcal{P}(n, \ell)$ were derived through the connection to the cut cone CC_n. While this is a fruitful source for obtaining a better linear description of $\mathcal{P}(n, \ell)$, this also might hint at severe drawbacks of this approach. In contrast to the betweenness polytope $\mathcal{P}_{BTW}$, the exact relation between the cut cone CC_n and $\mathcal{P}(n, \ell)$ is unknown. Even for $n \leq 5$ and given lengths $\ell \in \mathbb{Z}^n_{\geq 1}$, one can easily find a vector $d \in \mathbb{R}_+^{n(n-1)/2}$ that satisfies all classes of inequalities presented in [5], but does not belong to $\mathcal{P}(n, \ell)$. Additionally, it is well known that linear programming is not suited for solving dense problems over the cut polytope (see e.g. [72]). Formulating SRFLP over $\mathcal{P}(n, \ell)$ basically leads to such a dense problem. Therefore, we expect that the approach in [5] cannot be used to solve instances considerably larger than $n \geq 30$ in reasonable time, even if it is enhanced with a better linear description of the distance polytope $\mathcal{P}(n, \ell)$.

The betweenness approach presented in Section 2.3 has some interesting computational properties. By using only a partial linear description of the betweenness polytope $\mathcal{P}_{BTW}$, Amaral [4] solved many instances with $n \leq 35$ to optimality. The lower bounds were quite strong and always matched the optimal values. Although this is the case for all instances considered in [4], we have found randomly generated instances with only $n = 6$, where these lower

bounds are indeed not sufficient. Our tests indicate that the linear relaxation is optimal for $n \leq 5$. On the other hand, we have also solved instances with up to $n = 50$ with this approach. Hence, the result is the following: if one is fortunate, the betweenness approach can easily solve relatively large instances by using modern LP solvers and hardware; but if one is unfortunate, even very small instances cannot be solved. Of course, more valid inequalities for $\mathcal{P}_{BTW}$ could be added to further strengthen the linear relaxation, but again, this does not guarantee that one will find an optimal solution. This is a major downside of the betweenness approach compared to the *exact* approaches in Section 2.1 and Section 2.2. A natural idea to fix this issue, is to include the integrality conditions of the betweenness variables, i.e., solving the model (BTW) with a branch-and-bound approach. We will now explain, why this does not lead to a satisfying exact solution approach.

The first problem arises when it comes to solving the basic linear relaxation of model (BTW). It turns out that the simplex algorithm is incapable of solving it in reasonable time. That is why Amaral [4] used an interior-point method for this purpose. Using Gurobi 9.0 on one core on a Linux system with an Intel i7-4790k processor, the barrier algorithm solves the linear relaxation for $n = 42$ in about 10 minutes. A concurrent run of primal and dual simplex, each on one core, is not even finished after one day of running time. Unfortunately, the computation time cannot be reduced by a cutting plane approach, i.e., by separating only inequalities that are violated by the current fractional solution. Doing so is even worse, because too many inequalities are needed. This would increase the number of simplex iterations, which are very expensive due to the number of variables and constraints, more and more. The poor behavior of the simplex algorithm also concerns the reoptimization step after branching. Our tests show that it is more effective to solve each node in the branch-and-bound tree from scratch with the barrier algorithm than using the dual simplex. The other problem is simply the number of nodes that have to be explored in a branch-and-bound approach. The linear relaxation has a very odd property: even if the lower bound matches the optimal value, the majority of variables is still greatly fractional. Fixing one of the betweenness variables often barely improves the lower bound, if at all. Sometimes hundred of nodes have to be explored for randomly generated instances with only $n = 20$.

The best way to use the betweenness variables in an exact solution approach for SRFLP, is probably to tighten the linear relaxation as far as possible with further valid inequalities. Branching should only be done if it is really necessary, i.e., no violated inequality can be found in reasonable time. By doing so, and accepting the very high memory and computing time requirements, we believe that quite large instances could be solved with this approach. The main question remains, how often the linear relaxation is insufficient and branching is inevitable. However, we expect the lower bounds to be relatively close to the optimum in all cases.

All in all, very tight bounds seem to be the key for solving the largest possible instances of SRFLP. For this purpose, semidefinite relaxations (see e.g. [6, 8, 10, 12, 41, 43, 44]) are an excellent alternative to the LP-based approaches presented in this chapter. We will address existing semidefinite relaxations for SRFLP in Chapter 3 and further enhance them in Chapter 5. Chapter 4 is concerned with their suitable algorithmic treatment. The best semidefinite relaxations will turn out to be much stronger than the linear relaxation of the betweenness approach by Amaral [4]. Moreover, they will reveal no weaknesses on small instances as the

betweenness approach did. With an appropriate solution method, we will not only solve the majority of instances in a noticeably smaller amount of time, but also use only a tiny fraction of memory space compared to the interior-point methods for linear programming.

Chapter 3

Semidefinite Relaxations for SRFLP

The final conclusion at the end of Chapter 2 suggests to aim for stronger lower bounds than possible with linear programming techniques. To achieve this, semidefinite relaxations are a promising alternative for SRFLP. In this chapter, we present all semidefinite relaxations used in [6, 8, 10, 12, 41, 42, 44] and how they can be constructed. More precisely, in Section 3.1 we show in detail how SRFLP can be formulated as a constrained quadratic optimization problem in bivalent variables, which was first done in the initiating paper [6]. The semidefinite relaxations for SRFLP are then deduced in a canonical way in Section 3.2. All other semidefinite approaches, that were proposed later, concentrated on reducing the number of constraints [12], strengthening the relaxations with additional inequalities [8, 44], or using a suitable combination of optimization methods for solving the relaxations [44].

We remark that the presented semidefinite relaxations are not restricted to SRFLP and easily can be adapted for some other related problems. They can be used for more general quadratic ordering problems by adapting the objective function appropriately, see [44]. However, we mainly refer to the setting of SRFLP. Using semidefinite relaxations for these problems is very promising, since they can be formulated as a max-cut problem with some additional constraints. In fact, the feasible set is a face of a cut polytope [17]. Thus, conveying methods that are known to be efficient for the max-cut problem to SRFLP, or quadratic ordering problems in general, seems to be a natural idea.

We will also prove in Section 3.2 that the semidefinite relaxations for SRFLP are indeed at least as strong as the linear relaxation of the betweenness formulation (BTW) (see Section 2.3). Section 3.3 is concerned with the application of interior-point methods to the presented semidefinite relaxations and will reveal some major issues when this direct solution approach is used. More suitable algorithmic approaches, which are based on Lagrangian duality, are addressed in Chapter 4.

3.1 A bivalent quadratic formulation

Many combinatorial optimization problems, including SRFLP, can be formulated as a binary quadratic optimization problem. A standard semidefinite relaxation for these problems is

then obtained in a routine manner. Concerning SRFLP, the ordering variables (2.1) provide the basis for a binary quadratic formulation, respectively for the semidefinite relaxations. However, it is more convenient to use variables with values in $\{-1, 1\}$. Therefore, we introduce the following bivalent ordering variables y_{ij}, $i, j \in [n]$, $i < j$:

$$y_{ij} = \begin{cases} +1, & \text{if } i \text{ lies to the left of } j \\ -1, & \text{otherwise.} \end{cases} \tag{3.1}$$

These variables have a close connection to the betweenness variables (2.12). For instance, for $i, j, k \in [n]$, $i < j < k$, we have the equivalences

$$b_{ijk} = 1 \quad \Longleftrightarrow \quad (y_{ij} = 1 \ \wedge \ y_{jk} = 1) \ \vee \ (y_{ij} = -1 \ \wedge \ y_{jk} = -1) \quad \Longleftrightarrow \quad y_{ij}y_{jk} = 1.$$

Said in words: j lies between i and k if and only if the relative ordering of i and j equals the relative ordering of j and k. By case analysis, the betweenness variables can be expressed in quadratic terms of the bivalent ordering variables (3.1) in the following way (see [43]):

$$\begin{aligned} &b_{ijk} = \frac{1 - y_{ji}y_{jk}}{2}, \ i, j, k \in [n], \ j < i < k, \quad b_{ijk} = \frac{1 + y_{ij}y_{jk}}{2}, \ i, j, k \in [n], \ i < j < k, \\ &b_{ijk} = \frac{1 - y_{ij}y_{kj}}{2}, \ i, j, k \in [n], \ i < k < j. \end{aligned} \tag{3.2}$$

Using these transformations, we can rewrite the objective function (2.17) as

$$\sum_{\substack{i,j \in [n] \\ i<j}} \sum_{k \in [n]} \frac{c_{ij}}{2} \ell_k - \sum_{\substack{i,j \in [n] \\ i<j}} \frac{c_{ij}}{2} \left[\sum_{\substack{k \in [n] \\ k<i}} \ell_k y_{ki} y_{kj} - \sum_{\substack{k \in [n] \\ i<k<j}} \ell_k y_{ik} y_{kj} + \sum_{\substack{k \in [n] \\ k>j}} \ell_k y_{ik} y_{jk} \right]. \tag{3.3}$$

Furthermore, we need constraints on the ordering variables (3.1) that restrict them to valid orderings only. Tucker [82] and Younger [84] showed that if the so-called 3-cycle inequalities

$$-1 \leq y_{ij} + y_{jk} - y_{ik} \leq 1, \quad i, j, k \in [n], \ i < j < k, \tag{3.4}$$

are satisfied, then the vector $y \in \{-1, 1\}^{n(n-1)/2}$ of ordering variables represents a valid ordering. The inequalities (3.4) can also be written as

$$|y_{ij} + y_{ik} - y_{jk}| = 1, \quad i, j, k \in [n], \ i < j < k.$$

Squaring both sides leads to (see [44])

$$y_{ij}^2 + y_{jk}^2 + y_{ik}^2 + 2(y_{ij}y_{jk} - y_{ij}y_{ik} - y_{ik}y_{jk}) = 1, \quad i, j, k \in [n], \ i < j < k,$$

which is equivalent to

$$y_{ij}y_{jk} - y_{ij}y_{ik} - y_{ik}y_{jk} = -1, \quad i, j, k \in [n], \ i < j < k, \tag{3.5}$$

as $y_{ij}^2 = 1$ for all $i, j \in [n]$, $i < j$. Throughout this thesis, we will call the equations (3.5) 3-cycle equations. Interestingly, we also obtain these 3-cycle equations if we substitute the betweenness variables in (2.13) by using transformations (3.2):

$$\begin{aligned} b_{ijk} + b_{ikj} + b_{jik} = \frac{1 + y_{ij}y_{jk}}{2} + \frac{1 - y_{ik}y_{jk}}{2} + \frac{1 - y_{ij}y_{ik}}{2} = 1 \\ \iff \qquad y_{ij}y_{jk} - y_{ik}y_{jk} - y_{ij}y_{ik} = -1. \end{aligned}$$

Hence, we obtain the following bivalent quadratic formulation for SRFLP:

$$\begin{aligned} \min \quad & K - \sum_{\substack{i,j \in [n] \\ i<j}} \frac{c_{ij}}{2} \left[\sum_{\substack{k \in [n] \\ k<i}} \ell_k y_{ki} y_{kj} - \sum_{\substack{k \in [n] \\ i<k<j}} \ell_k y_{ik} y_{kj} + \sum_{\substack{k \in [n] \\ k>j}} \ell_k y_{ik} y_{jk} \right] \\ \text{s.t.} \quad & y_{ij}y_{jk} - y_{ij}y_{ik} - y_{ik}y_{jk} = -1, && i, j, k \in [n],\ i < j < k, \\ & y_{ij} \in \{-1, 1\}, && i, j \in [n],\ i < j, \end{aligned} \tag{3.6}$$

where $K \coloneqq \sum_{\substack{i,j \in [n] \\ i<j}} \sum_{k \in [n]} \frac{c_{ij}}{2} \ell_k$.

It is shown in [17] that the 3-cycle equations (3.5) describe the smallest linear subspace that contains the so-called *quadratic ordering polytope*

$$\mathcal{P}_{QO} \coloneqq \operatorname{conv} \left\{ yy^\top : y \in \{-1, 1\}^{\binom{n}{2}},\ |y_{ij} + y_{jk} - y_{ik}| = 1,\ i, j, k \in [n],\ i < j < k \right\}. \tag{3.7}$$

We can now see that SRFLP basically corresponds to a max-cut problem with additional equality constraints, namely the 3-cycle equations (3.5). Moreover, we have $\mathcal{P}_{QO} \subsetneq \mathcal{P}_C$ for $n \geq 3$ and $\mathcal{P}_{QO}$ is a face of $\mathcal{P}_C$ (see [17]).

For the sake of completeness, and for a better understanding of the problem structure, we will now prove why the 3-cycle equations (3.5) are indeed both necessary and sufficient in order to model only valid orderings. As a byproduct, this also gives a hint at how good feasible solutions, i.e., orderings of the facilities, can be found heuristically. Such primal heuristics that are based on the solutions of semidefinite relaxations for SRFLP are discussed in Section 5.2.

For modeling only valid orderings, we have to ensure transitivity: if the relative ordering of i and j is the same as the the relative ordering of j and k, the same must hold for i and k, e.g., if i lies to the left of j and j lies to the left of k, i also must lie to the left of k. By introducing additional variables y_{ij}, $i, j \in [n]$, $i > j$, with $y_{ij} = -y_{ji}$ satisfying the identity

$$y_{ij} + y_{ji} = 0, \quad i, j \in [n],\ i \neq j, \tag{3.8}$$

we can formally state this transitivity condition as

$$y_{ij} = y_{jk} \quad \implies \quad y_{ik} = y_{ij}, \quad i, j, k \in [n],\ |\{i, j, k\}| = 3. \tag{3.9}$$

This can also be written as the quadratic constraints (see [6])

$$(y_{ij} + y_{jk})(y_{ik} - y_{ij}) = 0, \quad i, j, k \in [n],\ |\{i, j, k\}| = 3,$$

which expand to

$$y_{ij}y_{ik} - y_{ij}^2 + y_{jk}y_{ik} - y_{jk}y_{ij} = 0, \quad i,j,k \in [n],\ |\{i,j,k\}| = 3. \tag{3.10}$$

Since $y_{ij}^2 = 1$, the latter equations contain the 3-cycle equations (3.5). However, both sets of equations are equivalent, because (3.10) contains only redundant equations for choices other than $i < j < k$.

We now know that the 3-cycle equations (3.5) are indeed necessary. In order to prove that they are sufficient, we define the set

$$Y_n \coloneqq \left\{ y \in \{-1,1\}^{n(n-1)/2} \colon y \text{ satisfies (3.5)} \right\}.$$

It is then shown in [6], how each $y \in Y_n$ corresponds to a unique permutation $\pi \in \Pi_n$, and vice versa. It is straight forward to construct a corresponding vector $y \in Y_n$ for a given permutation $\pi = (\pi_1, \ldots, \pi_n) \in \Pi_n$. For all $i, j \in [n]$, $i < j$, we set

$$y_{\pi_i \pi_j} \coloneqq 1, \tag{3.11}$$

and then use the identity (3.8). The resulting vector clearly satisfies the transitivity conditions (3.9) by construction, and hence, also the derived equations (3.5). Moreover, each permutation $\pi \in \Pi_n$ yields a different $y \in Y_n$, confirming that $|Y_n| \geq |\Pi_n| = n!$.

The 3-cycle equations (3.5) are also sufficient, because there is a bijection $f \colon Y_n \to \Pi_n$, which was proven in [17] in a more general setting. However, we follow the proofs in [6, 8, 12] which are closely related to SRFLP. Such a bijection $f \colon Y_n \to \Pi_n$, $y \mapsto (\pi_1, \ldots, \pi_n)$, is therein given by

$$\pi_k \coloneqq \frac{n + 1 - P_k}{2}, \quad k \in [n], \tag{3.12}$$

where

$$P_k \coloneqq \sum_{\substack{j \in [n] \\ j \neq k}} y_{kj} = \sum_{\substack{j \in [n] \\ j < k}} -y_{jk} + \sum_{\substack{j \in [n] \\ j > k}} y_{kj}, \quad k \in [n]. \tag{3.13}$$

Due to their definition, the P_k values belong to the set

$$\mathcal{P} \coloneqq \{-(n-1), -(n-3), \ldots, n-3, n-1\}. \tag{3.14}$$

Thus, we have $\pi_k \in \{1, \ldots, n\}$ for all $k \in [n]$. Therefore, $\pi = (\pi_1, \ldots, \pi_n)$ is a member of Π_n if and only if all P_k values are distinct.

Lemma 3.1.1 ([6, Lemma 1])
If $y \in Y_n$, then all P_k values are distinct, i.e., $f(y)$ is a valid ordering.

Proof. Assume that we have $P_{k_1} = P_{k_2}$ for $k_1, k_2 \in [n]$, $k_1 \neq k_2$, w.l.o.g. $k_1 < k_2$. Definition (3.13) gives

$$\sum_{\substack{k \in [n] \\ k < k_1}} -y_{kk_1} + \sum_{\substack{k \in [n] \\ k_1 < k < k_2}} y_{k_1 k} + \sum_{\substack{k \in [n] \\ k_2 < k}} y_{k_1 k} + y_{k_1 k_2} = \sum_{\substack{k \in [n] \\ k < k_1}} -y_{kk_2} + \sum_{\substack{k \in [n] \\ k_1 < k < k_2}} -y_{kk_2} + \sum_{\substack{k \in [n] \\ k_2 < k}} y_{k_2 k} - y_{k_1 k_2},$$

which leads to

$$\begin{aligned}&\sum_{\substack{k\in[n]\\k<k_1}} -y_{kk_1}y_{k_1k_2} + \sum_{\substack{k\in[n]\\k_1<k<k_2}} y_{k_1k}y_{k_1k_2} + \sum_{\substack{k\in[n]\\k_2<k}} y_{k_1k}y_{k_1k_2} + 1\\ = &\sum_{\substack{k\in[n]\\k<k_1}} -y_{kk_2}y_{k_1k_2} + \sum_{\substack{k\in[n]\\k_1<k<k_2}} -y_{kk_2}y_{k_1k_2} + \sum_{\substack{k\in[n]\\k_2<k}} y_{k_2k}y_{k_1k_2} - 1,\end{aligned}$$

by multiplying by $y_{k_1k_2}$ on both sides and using $y_{k_1k_2}^2 = 1$. Rearranging then yields

$$\begin{aligned}\sum_{\substack{k\in[n]\\k<k_1}} -\left(y_{kk_2}y_{k_1k_2} + y_{kk_1}y_{k_1k_2}\right) + &\sum_{\substack{k\in[n]\\k_1<k<k_2}} \left(y_{kk_2}y_{k_1k_2} - y_{k_1k}y_{k_1k_2}\right)\\ &+ \sum_{\substack{k\in[n]\\k_2<k}} \left(y_{k_2k}y_{k_1k_2} - y_{k_1k}y_{k_1k_2}\right) = 2.\end{aligned}$$

Due to $y \in Y_n$, the following equations hold:

$$\begin{aligned}-y_{kk_2}y_{k_1k_2} + y_{kk_1}y_{k_1k_2} &= y_{kk_1}y_{kk_2} - 1, && k \in [n],\ k < k_1 < k_2,\\ -y_{kk_2}y_{k_1k_2} - y_{k_1k}y_{k_1k_2} &= -y_{k_1k}y_{kk_2} - 1, && k \in [n],\ k_1 < k < k_2,\\ y_{k_2k}y_{k_1k_2} - y_{k_1k}y_{k_1k_2} &= y_{k_1k}y_{k_2k} - 1, && k \in [n],\ k_1 < k_2 < k.\end{aligned}$$

Thus, we obtain

$$\sum_{\substack{k\in[n]\\k<k_1}} \left(y_{kk_1}y_{kk_2} - 1\right) + \sum_{\substack{k\in[n]\\k_1<k<k_2}} \left(-y_{k_1k}y_{kk_2} - 1\right) + \sum_{\substack{k\in[n]\\k_2<k}} \left(y_{k_1k}y_{k_2k} - 1\right) = 2,$$

which is equivalent to

$$\sum_{\substack{k\in[n]\\k<k_2}} y_{kk_1}y_{kk_2} - \sum_{\substack{k\in[n]\\k_1<k<k_2}} y_{k_1k}y_{kk_2} \sum_{\substack{k\in[n]\\k_2<k}} y_{k_1k}y_{k_2k} = n.$$

This is a contradiction, because the left-hand side is bounded from above by $n-2$. □

It remains to prove that each permutation $\pi \in \Pi_n$ is attained for exactly one $y \in Y_n$. For this purpose, we consider a system of n equations in $\binom{n}{2}$ variables $y_{ij} \in \{-1, 1\}$, $i, j \in [n]$, $i < j$. We know that all values P_k, $k \in [n]$, as defined in (3.13) are distinct (see Lemma 3.1.1) and have values in the set $\mathcal{P}$, see (3.14). Therefore, each permutation $\pi \in \Pi_n$ is represented by exactly one $y \in Y_n$ if and only if the system

$$P_k = \sum_{\substack{j\in[n]\\j<k}} -y_{jk} + \sum_{\substack{j\in[n]\\j>k}} y_{kj} = \beta_k, \quad k \in [n], \tag{3.15}$$

has a unique solution, where the values β_k on the right hand side satisfy

$$\mathcal{P} = \{-(n-1), -(n-3), \ldots, n-3, n-1\} = \{\beta_k \colon k \in [n]\}. \tag{3.16}$$

Lemma 3.1.2 ([8, Lemma 3])
The system (3.15) has a unique solution.

Proof. The proof is by induction. We have two base cases for $n \in \{2, 3\}$. For $n = 2$, we have $\mathcal{P} = \{-1, 1\} = \{\beta_1, \beta_2\}$ and the system (3.15) reads

$$\begin{aligned} y_{12} &= \beta_1, \\ -y_{12} &= \beta_2 = -\beta_1, \end{aligned}$$

which always has a unique solution. For $n = 3$, we have $\mathcal{P} = \{-2, 0, 2\} = \{\beta_1, \beta_2, \beta_3\}$ and get the system

$$\begin{aligned} y_{12} + y_{13} &= \beta_1, \\ -y_{12} + y_{23} &= \beta_2, \\ -y_{13} - y_{23} &= \beta_3. \end{aligned}$$

We can see by case analysis that this system admits a unique solution for all choices of $\beta_k,\ k = 1, 2, 3$. For example, for $\beta_1 = -2$, $\beta_2 = 0$, $\beta_3 = 2$ the unique solution is $y_{12} = -1$, $y_{13} = -1$, $y_{23} = -1$. This concludes the base cases.

Let $n \geq 4$ and $k_1, k_2 \in [n]$ be such that $\beta_{k_1} = -(n-1)$ and $\beta_{k_2} = n - 1$, which exist by assumption. In terms of (3.12), k_1 is located at position n and k_2 is located at position 1. Given k_1 and k_2, any solution of system (3.15) must satisfy

$$\begin{aligned} &y_{jk_1} = +1 \text{ for all } j \in [n],\ j < k_1, \qquad && y_{k_1,j} = -1 \text{ for all } j \in [n],\ j > k_1, \\ &y_{jk_2} = -1 \text{ for all } j \in [n],\ j < k_2, \qquad && y_{k_2,j} = +1 \text{ for all } j \in [n],\ j > k_2. \end{aligned} \tag{3.17}$$

Now consider any equation of system (3.15) for $k \in [n] \setminus \{k_1, k_2\}$. We have four possible cases:

- $k < k_1$, $k < k_2$ where $y_{kk_1} = 1$, $y_{kk_2} = -1$:

$$\sum_{\substack{j \in [n] \\ j < k}} -y_{jk} + \sum_{\substack{j \in [n] \\ j > k \\ j \neq k_1, k_2}} y_{kj} + y_{kk_1} + y_{k,k_2} = \beta_k,$$

- $k_1 < k < k_2$ where $y_{k_1 k} = -1$, $y_{kk_2} = -1$:

$$\sum_{\substack{j \in [n] \\ j < k \\ j \neq k_1}} -y_{jk} - y_{k_1 k} + \sum_{\substack{j \in [n] \\ j > k \\ j \neq k_2}} y_{kj} + y_{k,k_2} = \beta_k,$$

- $k_2 < k < k_1$ where $y_{kk_1} = 1$, $y_{k_2 k} = 1$:

$$\sum_{\substack{j \in [n] \\ j < k \\ j \neq k_2}} -y_{jk} - y_{k_2 k} + \sum_{\substack{j \in [n] \\ j > k \\ j \neq k_1}} y_{kj} + y_{k,k_1} = \beta_k,$$

- $k > k_1$, $k > k_2$ where $y_{k_1k} = -1$, $y_{k_2k} = 1$:

$$\sum_{\substack{j\in[n]\\ j<k\\ j\neq k_1,k_2}} -y_{jk} - y_{k_1k} - y_{k_2k} + \sum_{\substack{j\in[n]\\ j>k}} y_{kj} = \beta_k.$$

In all four cases, cancellation due to the conditions (3.17) occurs. Hence, we get a reduced system with only $n-2$ equations and $\binom{n-2}{2}$ unknowns. This system has the same form as (3.15). By our induction hypothesis, the reduced system has a unique solution. The solution can be extended to a uniquely defined solution of the original system with the help of (3.17). We conclude that system (3.15) has a unique solution. □

As a result, Y_n contains at most $n!$ elements. Thus, we have $|Y_n| = |\Pi_n|$ and any solution of system (3.15) yields a $y \in Y_n$.

Theorem 3.1.3 ([8, Theorem 1])
The function $f\colon Y_n \to \Pi_n$ as defined in (3.12) and (3.13) is a bijection.

Proof. Assume, we have $y_1, y_2 \in Y_n$ with $f(y_1) = f(y_2)$. Then we get $y_1 = y_2$ by applying Lemma 3.1.2. Thus, f is injective. Moreover, applying Lemma 3.1.2 and the fact $|Y_n| = |\Pi_n|$ also yields a $y \in Y_n$ with $f(y) = \pi$ for each $\pi \in \Pi_n$. Hence, f is surjective. □

3.2 Semidefinite relaxations

Recall that an exact formulation for SRFLP (see (3.6)) is given by

$$\begin{aligned} \min \quad & K - \sum_{\substack{i,j\in[n]\\ i<j}} \frac{c_{ij}}{2} \left[\sum_{\substack{k\in[n]\\ k<i}} \ell_k y_{ki} y_{kj} - \sum_{\substack{k\in[n]\\ i<k<j}} \ell_k y_{ik} y_{kj} + \sum_{\substack{k\in[n]\\ k>j}} \ell_k y_{ik} y_{jk} \right] \\ \text{s.t.} \quad & y_{ij}y_{jk} - y_{ij}y_{ik} - y_{ik}y_{jk} = -1, && i,j,k \in [n],\ i<j<k, \\ & y_{ij} \in \{-1,1\}, && i,j \in [n],\ i<j. \end{aligned} \tag{3.18}$$

3.2.1 Basic relaxations

We now move to matrix-based relaxations for SRFLP. For this purpose, we collect all ordering variables (3.1) in a vector y and define the matrix

$$Y \coloneqq yy^\top \in \mathbb{R}^{\binom{n}{2}\times\binom{n}{2}}.$$

Clearly, the matrix Y has rank one, $\operatorname{rank}(Y) = 1$, since all rows are pairwise linearly dependent. Additionally, Y is symmetric, $Y \in \mathcal{S}_{\binom{n}{2}}$, and we have

$$v^\top Y v = v^\top yy^\top v = \left(v^\top y\right)^2 \geq 0$$

for all $v \in \mathbb{R}^{\binom{n}{2}}$. Hence, Y is positive semidefinite, $Y \succeq 0$. Finally, the main diagonal entries are all ones, since they correspond to $y_{ij}^2 = 1$. For this, we write $\operatorname{diag}(Y) = e$, where e is the vector of all ones of appropriate dimension. With a suitable choice of the symmetric cost matrix C, SRFLP can be formulated as the following matrix-based optimization problem, primarily done in [6]:

$$\begin{aligned} \min \quad & \langle C, Y \rangle + K \\ \text{s.t.} \quad & Y_{ij,jk} - Y_{ij,ik} - Y_{ik,jk} = -1, \qquad i,j,k \in [n],\ i < j < k, \\ & \operatorname{diag}(Y) = e, \\ & \operatorname{rank}(Y) = 1, \\ & Y \succeq 0. \end{aligned} \tag{3.19}$$

Removing the nonconvex rank-one constraint yields the basic semidefinite relaxation

$$\begin{aligned} \min \quad & \langle C, Y \rangle + K \\ \text{s.t.} \quad & Y_{ij,jk} - Y_{ij,ik} - Y_{ik,jk} = -1, \qquad i,j,k \in [n],\ i < j < k, \\ & \operatorname{diag}(Y) = e, \\ & Y \succeq 0. \end{aligned} \tag{SDP$_1$}$$

This relaxation has $\mathcal{O}(n^3)$ constraints, and thus, is difficult to solve with interior-point methods for even medium-sized instances. This is why Anjos & Yen [12] suggested to sum up the 3-cycle equations (3.5) over k. This yields the $\mathcal{O}(n^2)$ constraints

$$\sum_{\substack{k \in [n] \\ k \neq i,j}} Y_{ij,jk} - \sum_{\substack{k \in [n] \\ k \neq i,j}} Y_{ij,ik} - \sum_{\substack{k \in [n] \\ k \neq i,j}} Y_{ik,jk} = -(n-2), \quad i,j \in [n],\ i < j. \tag{3.20}$$

In fact, if we substitute the 3-cycle equations in (3.19) by equations (3.20), the resulting optimization problem

$$\min \{ \langle C, Y \rangle + K : Y \text{ satisfies (3.20)},\ \operatorname{diag}(Y) = e,\ \operatorname{rank}(Y) = 1,\ Y \succeq 0 \} \tag{3.21}$$

is another exact formulation for SRFLP, see [12]. Dropping the rank-one constraint yields the semidefinite relaxation

$$\min \{ \langle C, Y \rangle + K : Y \text{ satisfies (3.20)},\ \operatorname{diag}(Y) = e,\ Y \succeq 0 \} \tag{SDP$_0$}$$

for SRFLP, which is cheaper, but also weaker than (SDP$_1$).

3.2.2 Strengthened relaxations

To strengthen the relaxation (SDP$_1$), we consider the particular cut polytope

$$\mathcal{P}_C \coloneqq \operatorname{conv} \left\{ yy^\top : y \in \{-1,1\}^{\binom{n}{2}} \right\}$$

and its semidefinite relaxation

$$\left\{Y \in \mathbb{R}^{\binom{n}{2}\times\binom{n}{2}} \colon \operatorname{diag}(Y) = e,\ Y \succeq 0\right\}.$$

Since SRFLP corresponds to a max-cut problem with additional constraints (3.5), all valid inequalities for the cut polytope $\mathcal{P}_C$ can be used to tighten (SDP_1). In particular, Anjos & Vanelli [8] suggested to add the well-known triangle inequalities that are facet-defining for the cut polytope $\mathcal{P}_C$ (see e.g. [21]). These inequalities define the so-called *metric polytope*

$$\mathcal{M} \coloneqq \left\{ Y \in \mathbb{R}^{\binom{n}{2}\times\binom{n}{2}} \colon \begin{pmatrix} 1 & 1 & 1 \\ 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} Y_{ij} \\ Y_{ik} \\ Y_{jk} \end{pmatrix} \geq \begin{pmatrix} -1 \\ -1 \\ -1 \\ -1 \end{pmatrix},\ 1 \leq i < j < k \leq \binom{n}{2} \right\}. \tag{3.22}$$

Thus, Anjos & Vanelli [8] proposed the relaxation

$$\begin{aligned} \min \quad & \langle C, Y\rangle + K \\ \text{s.t.} \quad & Y_{ij,jk} - Y_{ij,ik} - Y_{ik,jk} = -1, && i,j,k \in [n],\ i<j<k, \\ & Y_{i,j} + Y_{i,k} + Y_{j,k} \geq -1, && 1 \leq i < j < k \leq n(n-1)/2, \\ & Y_{i,j} - Y_{i,k} - Y_{j,k} \geq -1, && 1 \leq i < j < k \leq n(n-1)/2, \\ & -Y_{i,j} + Y_{i,k} - Y_{j,k} \geq -1, && 1 \leq i < j < k \leq n(n-1)/2, \\ & -Y_{i,j} - Y_{i,k} + Y_{j,k} \geq -1, && 1 \leq i < j < k \leq n(n-1)/2, \\ & \operatorname{diag}(Y) = e, \\ & Y \succeq 0. \end{aligned} \tag{3.23}$$

(SDP_2)

Since there are $\mathcal{O}(n^6)$ triangle inequalities, Anjos & Vanelli [8] suggested to include only a small subset of them by using them in a cutting plane approach.

We now show that (SDP_2) always yields stronger lower bounds than the linear relaxation of the betweenness model (BTW) in Section 2.3. For this purpose, we assume that the optimum in (SDP_2) is attained. This property will be addressed in Section 3.3.

Proposition 3.2.1
(SDP_2) is at least as strong as the linear relaxation of model (BTW).

Proof. Let Y^* be an optimal solution of (SDP_2) and let z^* be the corresponding optimal value. We now construct a feasible solution $b \in [0,1]^{n(n-1)(n-2)/2}$ for the linear relaxation of (BTW) that has the same objective value z^*. For this, we define

$$b_{ijk} \coloneqq \begin{cases} \frac{1-Y^*_{ji,jk}}{2}, & i,j,k \in [n],\ j<i<k \\ \frac{1+Y^*_{ij,jk}}{2}, & i,j,k \in [n],\ i<j<k \\ \frac{1-Y^*_{ij,kj}}{2}, & i,j,k \in [n],\ i<k<j. \end{cases} \tag{3.24}$$

The constraints $\operatorname{diag}(Y^*) = e$ and $Y^* \succeq 0$ together imply that all entries of Y^* are in $[-1,1]$, and hence, the betweenness variables defined in (3.24) satisfy the bound constraints (2.18).

For $i,j,k \in [n]$, $i<j<k$, we have

$$\begin{aligned} b_{ijk}+b_{ikj}+b_{jik} &= \frac{1+Y^*_{ij,jk}}{2}+\frac{1-Y^*_{ik,jk}}{2}+\frac{1-Y^*_{ij,ik}}{2} \\ &= \frac{3}{2}+\frac{1}{2}\left(\underbrace{Y^*_{ij,jk}-Y^*_{ik,jk}-Y^*_{ij,ik}}_{=-1 \text{ by (3.23)}}\right) \\ &= \frac{3}{2}-\frac{1}{2}=1, \end{aligned}$$

and thus, the variables b_{ijk} satisfy (2.13). To avoid case analysis in the following, we assume no particular ordering of the indices i,j,k,h. Instead, we use the property

$$b_{abc}=\frac{1+Y_{ab,bc}}{2}=\frac{1-Y_{ba,bc}}{2}. \tag{3.25}$$

Using (3.25) we obtain

$$\begin{aligned} b_{ihj}+b_{ihk}+b_{jhk} &= \frac{1+Y^*_{ih,hj}}{2}+\frac{1+Y^*_{ih,hk}}{2}+\frac{1+Y^*_{jh,hk}}{2} \\ &= \frac{3}{2}+\frac{1}{2}\left(Y^*_{ih,hj}+Y^*_{ih,hk}+Y^*_{jh,hk}\right) \\ &= \frac{3}{2}+\frac{1}{2}\left(\underbrace{-Y^*_{hi,hj}-Y^*_{hi,hk}-Y^*_{hj,hk}}_{\leq 1}\right) \\ &\leq \frac{3}{2}+\frac{1}{2}=2, \end{aligned}$$

for all $i,j,k,h \in [n]$, $|\{i,j,k,h\}|=4$, since $Y^*_{hi,hj}+Y^*_{hi,hk}+Y^*_{hj,hk} \geq -1$ corresponds to a triangle inequality on the pairs (hi), (hj) and (hk). Thus, the inequalities (2.14) are satisfied. Moreover, we have

$$\begin{aligned} -b_{ihj}+b_{ihk}+b_{jhk} &= -\frac{1+Y^*_{ih,hj}}{2}+\frac{1+Y^*_{ih,hk}}{2}+\frac{1+Y^*_{jh,hk}}{2} \\ &= \frac{1}{2}+\frac{1}{2}\left(-Y^*_{ih,hj}+Y^*_{ih,hk}+Y^*_{jh,hk}\right) \\ &= \frac{1}{2}+\frac{1}{2}\left(\underbrace{Y^*_{hi,hj}-Y^*_{hi,hk}-Y^*_{hj,hk}}_{\geq -1}\right) \\ &\geq \frac{1}{2}-\frac{1}{2}=0, \end{aligned}$$

since again, a triangle inequality on the pairs (hi), (hj) and (hk) arises. Analogously, the same holds for all inequalities (2.15). As a result, the vector b of betweenness variables defined in (3.24) is feasible for the linear relaxation of (BTW). By construction of the objective function (3.3), the objective value of b in (BTW) coincides with z^*. Hence, the optimal value of the linear relaxation of (BTW) is less than or equal to z^*, which is the optimal value of (SDP_2). □

As already mentioned in Section 2.3, we see in the proof of Proposition 3.2.1 that the inequalities (2.14) and (2.15) correspond to the triangle inequalities of the cut polytope. While (SDP_2) involves $\mathcal{O}(n^6)$ triangle inequalities, the linear relaxation of (BTW) contains only $\mathcal{O}(n^4)$ of them. This difference comes from the more global modeling of the matrix-based approach, since it involves products of ordering variables that do not share a common facility. This is not possible with the linear model (BTW) which exploits some kind of sparsity. Moreover, it is not possible to express the bivalent ordering variables (3.1) in terms of the betweenness variables (2.12), see [7]. Due to the clearly larger set of triangle inequalities and the additional semidefinite constraint $Y \succeq 0$, (SDP_2) yields much tighter lower bounds than the linear relaxation of (BTW).

All semidefinite relaxations presented so far are formulated on the quadratic ordering polytope $\mathcal{P}_{QO}$, see (3.7). Hungerländer & Rendl [44] worked with the *linear-quadratic ordering polytope*

$$\mathcal{P}_{LQO} \coloneqq \operatorname{conv}\left\{ \begin{pmatrix} 1 \\ y \end{pmatrix} \begin{pmatrix} 1 \\ y \end{pmatrix}^\top : y \in \{-1,1\}^{\binom{n}{2}},\ y \text{ satisfies (3.5)} \right\} \tag{3.26}$$

instead, and defined the matrix

$$\overline{Y} \coloneqq \begin{pmatrix} 1 \\ y \end{pmatrix} \begin{pmatrix} 1 \\ y \end{pmatrix}^\top = \begin{pmatrix} 1 & y^\top \\ y & yy^\top \end{pmatrix} = \begin{pmatrix} 1 & y^\top \\ y & Y \end{pmatrix}.$$

Consider now again the matrix Y. We have

$$Y = yy^\top \quad \Longleftrightarrow \quad Y - yy^\top = 0.$$

Since the zero matrix is positive semidefinite, Hungerländer & Rendl [44] relaxed the nonconvex equation $Y - yy^\top = 0$ to

$$Y - yy^\top \succeq 0 \quad \Longleftrightarrow \quad \overline{Y} \succeq 0,$$

which is convex due to the Schur complement lemma (see e.g. [41]). As $\overline{Y} \succeq 0$ is in general a stronger constraint than $Y \succeq 0$ [43], the semidefinite relaxation (SDP_2) can be improved by replacing Y by $\overline{Y}$.

To further improve the relaxation, Hungerländer & Rendl [44] suggested to use an additional class of $\mathcal{O}(n^5)$ 'matrix cuts'. To this end, consider the 3-cycle inequalities (3.4) which also can be written as

$$1 - y_{ij} - y_{jk} + y_{ik} \geq 0, \quad 1 + y_{ij} + y_{jk} - y_{ik} \geq 0, \quad i,j,k \in [n],\ i < j < k. \tag{3.27}$$

A generic approach by Lovász and Schrijver [59] then multiplies the inequalities (3.27) by the nonnegative expressions $(1 - y_{lm})$ and $(1 + y_{lm})$ for any $l, m \in [n]$, $l < m$. Thus, for all $i,j,k,l,m \in [n]$, $i < j < k$, $l < m$, we obtain the Lovász-Schrijver-cuts

$$\begin{aligned} -1 - y_{lm} &\leq y_{ij} + y_{jk} - y_{ik} + y_{ij,lm} + y_{jk,lm} - y_{ik,lm} \leq 1 + y_{lm}, \\ -1 + y_{lm} &\leq y_{ij} + y_{jk} - y_{ik} - y_{ij,lm} - y_{jk,lm} + y_{ik,lm} \leq 1 - y_{lm}. \end{aligned} \tag{3.28}$$

Similar to the metric polytope $\mathcal{M}$, we define

$$\mathcal{LS} \coloneqq \left\{ \overline{Y} \in \mathbb{R}^{\binom{\binom{n}{2}+1}{2}} : \overline{Y} \text{ satisfies (3.28)} \right\}.$$

Altogether, Hungerländer & Rendl [44] proposed the following semidefinite relaxation for SRFLP:

$$\begin{aligned} \min \quad & \langle C, Y \rangle + K \\ \text{s.t.} \quad & Y_{ij,jk} - Y_{ij,ik} - Y_{ik,jk} = -1, \qquad i,j,k \in [n],\ i<j<k, \\ & \operatorname{diag}(\overline{Y}) = e, \\ & \overline{Y} \in \mathcal{M}, \\ & \overline{Y} \in \mathcal{LS}, \\ & \overline{Y} \succeq 0. \end{aligned} \tag{SDP$_3$}$$

Additionally, we can break the natural symmetry of SRFLP by fixing any single ordering variable y_{ij}, e.g., $y_{12} = 1$.

3.3 Performance of interior-point methods for SRFLP

Interior-point methods (IPMs) are theoretically the most efficient solution method for all reasonably defined semidefinite programs [33]. Due to their second-order nature, they are very robust and yield highly accurate solutions within polynomial time. This makes them especially interesting for problems, for which no customized algorithms are available [23]. Numerical results show that primal-dual path-following interior-point methods are the method of choice when it comes to optimizing over the elliptope in a reliable, efficient and accurate way [23, 41]. Therefore, it is not surprising that such methods were applied to the semidefinite relaxations for SRFLP, initially done in [6]. In the following, we recap how interior-point methods were used in [6,8–10,12] and how they performed. Afterwards, we give some insights into their scalability for large SRFLP instances and the associated limitations.

Although (SDP$_2$) has $\mathcal{O}(n^6)$ constraints, all semidefinite relaxations presented in Section 3.2 have a polynomial size in the number of facilities n. Hence, if they admit primal or dual strictly feasible solutions, they can be solved by primal-dual interior-point methods in polynomial time. Indeed, (SDP$_1$) and its corresponding dual problem, denoted by (DSDP$_1$), are strictly feasible [44], i.e., strong duality holds and the optimal value is attained in both problems. For instance, our numerical tests have shown that the symmetric matrix Y with the entries

$$Y_{ij,kl} \coloneqq \begin{cases} 1, & \text{if } i=k \ \wedge \ j=l \\ -\frac{1}{3}, & \text{if } j=k \ \wedge \ i<j<l \\ \frac{1}{3}, & \text{if } (i=k \ \wedge \ i<j<l) \ \vee (j=l \ \wedge \ i<j<k) \\ 0, & \text{otherwise} \end{cases} \tag{3.29}$$

on its upper triangular part is positive definite for all $n \leq 100$. Clearly, it is also feasible for (SDP$_1$). By setting the dual variables of all 3-cycle-equations (3.5) to zero, the strict

feasibility of ($DSDP_1$) follows from the strict feasibility of the dual problem of the basic semidefinite max-cut relaxation (see e.g. [33]). Note that the matrix Y given by (3.29) also satisfies all inequalities of (SDP_3), and of course, all equations of (SDP_0). Analogously, these semidefinite programs and their corresponding dual problems are strictly feasible.

The very first computational results for the basic semidefinite relaxation (SDP_1) were presented in [6] and the subsequent paper [9]. Using the interior-point solver CSDP (version 5.0) [15, 16] on a 2.0 GHz Dual Opteron with 16 GB RAM, (SDP_1) was solved for up to $n = 40$. The relaxations were solved within 3000 seconds and the optimality gaps typically remained smaller than 1%. These bounds were also used in a branch-and-bound algorithm in [9]. However, the results were quite sobering, since branching improved the bound of the root node only slightly. As concluded in [9], considerably larger running times would be necessary to obtain optimal solutions this way.

This motivated Anjos & Vanelli [8,10] to work with tighter relaxations by adding the triangle inequalities to (SDP_1), yielding the improved relaxation (SDP_2). Using the same hardware and software, they first solved the relaxation (SDP_1) and then iteratively added some violated triangle inequalities in a cutting plane approach. At each iteration, the optimal solution of the relaxation is used to find feasible layouts heuristically (see Section 5.2) and to find the 300 to 400 most-violated triangle inequalities. These inequalities are added to the relaxation which is then again solved from scratch. Instances with up to $n = 30$ were solved to global optimality using this approach. However, the running times were at least a couple of hours for $n \geq 25$, and even more than two days for one particular instance. The number of cutting plane iterations was at most 31 and not more than 11000 inequalities were in total added for every instance. Note that the relaxation does not need to be exact in order to prove global optimality. An absolute distance of the computed lower bound to the value of a known feasible solution smaller than one is sufficient, since the values of all solutions differ from each other by integral values only.

Lower bounds for larger instances with up to $n = 100$ were obtained in [12] by using the relaxation (SDP_0) and the same interior-point solver CSDP (version 5.0) [15, 16]. The calculations were carried out on a faster machine with a 2.4 GHz Quad Opteron processor and 16 GB RAM. The computed lower bounds were often about 1% weaker than the bounds for (SDP_1) and even considerably weaker on a few instances. For $n \leq 56$, the time saving compared to (SDP_1) was up to 66%. The optimality gaps typically remained smaller than 3% by using (SDP_0), but more than ten days of CPU time were still required to solve the relaxation for $n = 100$.

The above results show that applying interior-point methods to semidefinite relaxations for SRFLP are very double-edged. On the one hand, many previously unsolved instances with up to 30 facilities were solved in [8, 10] by using a sole cutting plane approach, i.e., without branching. Additionally, relatively strong lower bounds, which are only a few percentage points away from the optimum, were obtained for up to 100 facilities in [12]. On the other hand, the computing times increased dramatically with the number of facilities and added inequalities. The running times for successfully solved small instances were almost always much higher than those of the betweenness approach presented in [4], see Section 2.3. Although the semidefinite bounds are remarkably tight, an exact solution approach with interior-point

methods suffers from too many constraints or too slowly moving lower bounds in an branch-and-bound framework. In the following, we investigate why applying interior-point methods to semidefinite relaxations for SRFLP, such as (SDP_2), is challenging in practice.

Although interior-point methods have polynomial time complexity and require only a few iterations, the computational effort for a single iteration may be very high and depends on the structure of the concrete given problem [16]. Whereas the memory requirement increases quadratically, the running time has a cubic growth in the matrix dimension and the number of constraints for problems with sparse constraints, which is the case for our semidefinite relaxations [16]. Therefore, the computational effort for one iteration is at least $\mathcal{O}(m^3)$, where m is the number of constraints. This clarifies why (SDP_0) with only $\mathcal{O}(n^2)$ constraints is computationally feasible for $n = 100$, whereas (SDP_1) with $\mathcal{O}(n^3)$ constraints and especially (SDP_2) with $\mathcal{O}(n^6)$ constraints are too expensive. When (SDP_3) is used, the number of cuts we can add to strengthen the relaxation is severely limited. Since efficient warm start strategies are not available for interior-point methods in general, the relaxation must be solved from scratch several times, leading to significantly increasing running times. Hence, finding an appropriate small selection of promising cuts and removing cuts that are not needed, is crucial for the number of cutting plane iterations and the overall performance.

These difficulties are not restricted to SRFLP and also arise in other semidefinite relaxations for combinatorial problems [36], e.g., the max-cut problem [23]. Typically, many inequalities are violated by the current fractional solution and it is challenging to identify a small subset of those inequalities that are likely to be active at the optimum [23]. Computational results show that adding the most-violated inequalities at each iteration leads to a large number of cutting plane iterations. This is mainly because only a small fraction of inequalities added this way is actually active at the optimum [23].

Despite their nice theoretical properties, interior-point methods are not well-suited to solve tight semidefinite relaxations for large-scale SRFLP instances. They lack the possibility to exploit the sparsity of constraint matrices [33] and lead to excessive running times when many constraints are present. To solve (SDP_2) or even tighter relaxations such as (SDP_3) for large-scale SRFLP instances, different solutions methods have to be applied. Such customized methods should be able to deal with much more constraints and should allow to exploit the sparsity of constraint matrices. Typically, this leads to a tradeoff between the accuracy of the solution and the running time. The potential loss of accuracy is not that critical for combinatorial problems such as SRFLP, since lower bounds can be 'rounded' up or can be used in a branch-and-bound algorithm. In the next chapter, we present two specific approaches that fulfill our needs for large semidefinite relaxations and SRFLP instances.

Chapter 4

Practical Solution Methods based on Lagrangian Relaxation

Throughout this chapter, we consider any semidefinite relaxation for SRFLP that is formulated over the linear-quadratic ordering polytope $\mathcal{P}_{LQO}$ (3.26), such as (SDP_3). However, in view of our further enhancements in Chapter 5, we also allow other inequalities than those involved in (SDP_3).

To simplify the notation, we use $n_* \coloneqq \binom{n}{2} + 1$ in the following. Note that we can write any semidefinite relaxation for SRFLP over $\mathcal{P}_{LQO}$ in the compact form

$$\begin{array}{ll} \min & \langle C, X \rangle \\ \text{s.t.} & \mathcal{A}(X) \leq a \\ & \mathcal{B}(X) = e \\ & X \succeq 0, \end{array} \tag{4.1}$$

where $C, X \in \mathcal{S}_{n_*}$ and the cost matrix C is chosen appropriately. Here, the linear operator $\mathcal{B}\colon \mathcal{S}_{n_*} \to \mathbb{R}^{m_E}$ collects all $m_E = n_* + \binom{n}{3}$ equations, i.e., $\operatorname{diag}(X) = e$ and the 3-cycle equations (3.5). Moreover, the linear operator $\mathcal{A}\colon \mathcal{S}_{n_*} \to \mathbb{R}^{m_I}$ collects all considered inequalities, where the right hand side $a \in \mathbb{R}^{m_I}$ is chosen appropriately.

We already know (see Section 3.3) that a semidefinite relaxation of type (4.1) cannot be solved by interior-point methods for interesting problem sizes, say $n \geq 50$. However, an approximate solution of the relaxation is already sufficient for practical purposes. To handle relaxations with many constraints for large-scale problems, we have to use a solution approach that exploits the sparsity of the constraint matrices. The first attempt to solve the basic semidefinite relaxation (SDP_1) with up to 80 facilities was done in [6]. Therein, the spectral bundle (SB) solver [34,35] was used. The implemented spectral bundle method is well-known to be suited for large-scale semidefinite programs. However, its usage comes at the cost of a poor convergence rate, since it is only a first-order method. In fact, the computational results in [6] and [12] demonstrate, that solving the weaker relaxation (SDP_0) by using a primal-dual interior-point method yields much stronger lower bounds in significantly less time than applying the spectral bundle method to (SDP_1). Another drawback of the latter approach is that we have no direct access to the primal variable X. This makes finding good feasible orderings and especially separating the inequalities much harder.

More efficient and accurate solution approaches use the idea of Lagrangian relaxation to handle the large set of constraints in an elegant way. Hungerländer & Rendl [44] adapted the specific bundle approach in [23] to solve their proposed relaxation (SDP_3). This approach was prior also successfully applied to the max-cut problem in [72]. The bundle approach consists of a suitable combination of interior-point methods and the bundle method. While a few constraints are maintained explicitly in a subproblem that is solved by using an interior-point method, the majority of constraints is lifted into the objective function via Lagrangian duality. This leads to a convex but nonsmooth optimization problem, where the bundle method is used to approximate the optimal dual multipliers. We will deal with this approach in Section 4.1.

In Section 4.2, we present our suggested solution approach for the semidefinite relaxations for `SRFLP`. It consists of the ideas in [62] and the practical implementation in [52]. Whereas Hungerländer & Rendl [44] work with a partial Lagrangian, the method in [62] first reformulates the exact formulation (3.19) and then dualizes all constraints yielding a regularized problem. The corresponding nonstandard semidefinite bounds are slightly weaker than the usual ones, but can explicitly be computed by efficient methods of smooth optimization, such as quasi-Newton methods. This approach was also applied to the max-cut problem in [50] and outperformed the bundle approach in [72]. The latter is the foundation of the current leading method for `SRFLP` by Hungerländer & Rendl [44]. We conclude this chapter with a theoretical comparison of the presented methods when applied to large-scale `SRFLP` instances.

4.1 A bundle method approach

Let $\mathcal{E}$ be the particular elliptope

$$\mathcal{E} \coloneqq \left\{X \in \mathbb{R}^{n_* \times n_*} : \ \operatorname{diag}(X) = e, \ X \succeq 0\right\}.$$

Since minimizing the objective function of (4.1) over the elliptope $\mathcal{E}$ is computational feasible for even $n = 100$ by using an interior-point method, Hungerländer & Rendl [44] maintained the constraint $X \in \mathcal{E}$ explicitly and handled all other constraints through Lagrangian duality, i.e., by applying Lagrangian relaxation. For this purpose, consider the problem

$$z_{SDP} = \inf\left\{\langle C, X\rangle : X \in \mathcal{E}, \ \mathcal{A}(X) \leq a, \ \mathcal{B}(X) = e\right\}, \tag{4.2}$$

where $\mathcal{B}(X) = e$ now only collects the 3-cycle equations (3.5) symbolically.

Hungerländer & Rendl [44] then lifted the linear constraints $\mathcal{A}(X) \leq a$ and $\mathcal{B}(X) = e$ into the objective function and defined the partial Lagrangian

$$\mathcal{L}(X; \lambda, \mu) \coloneqq \langle C, X\rangle + \lambda^\top(\mathcal{A}(X) - a) + \mu^\top(\mathcal{B}(X) - e).$$

The associated dual function is given by

$$f(\lambda, \mu) \coloneqq \inf_{X \in \mathcal{E}} \mathcal{L}(X; \lambda, \mu) = -a^\top\lambda - e^\top\mu + \inf_{X \in \mathcal{E}} \left\langle C + \mathcal{A}^\top(\lambda) + \mathcal{B}^\top(\mu), X\right\rangle. \tag{4.3}$$

It follows from weak duality that we have

$$f(\lambda, \mu) \leq z_{SDP}$$

for all $\lambda \geq 0$, μ. Hence, we obtain a valid lower bound for SRFLP by evaluating f at any feasible point (λ, μ). Computing the strongest bound possible amounts to solving the dual problem

$$\sup_{\lambda \geq 0,\, \mu} f(\lambda, \mu).$$

Since (SDP_3) has strictly feasible points (see Section 3.3), strong duality holds and the optimum of the dual problem is attained at some point (λ^*, μ^*):

$$f(\lambda^*, \mu^*) = \max_{\lambda \geq 0,\, \mu} f(\lambda, \mu) = z_{SDP}.$$

It is well-known that the function f is concave but nonsmooth. Therefore, Hungerländer & Rendl [44] used the bundle method to minimize the convex function $f' := -f$. For a current given feasible point (λ, μ), an iteration of the bundle method requires the function value $f'(\lambda, \mu)$ and a subgradient of f' at (λ, μ). Both are obtained by solving a semidefinite program over the elliptope $\mathcal{E}$, see (4.3). Whereas the feasible set of this problem always remains the same, the objective function changes in every iteration.

Since including all inequalities $\mathcal{A}(X) \leq a$ at the same time is still too expensive, only a small subset $\mathcal{I}$ of inequalities is actually present. This subset $\mathcal{I}$ is adapted dynamically, i.e., nearly inactive inequalities with dual multipliers close to zero are dropped and new promising inequalities are added. As a result, the set of constraints that are dualized and also f itself will change from time to time. It can be shown that the iterates (λ_k, μ_k) in this dynamic bundle method converge to an optimal solution, see [14]. However, the bundle method has a weak asymptotic behavior. Hence, Hungerländer & Rendl [44] limited the number of bundle method iterations to a few hundred. Thus, the obtained solution is only an approximation to the optimal solution.

4.2 A regularized approach

We will now present in detail the motivation of our suggested approach for SRFLP. The generic method presented in [62] also uses Lagrangian duality (or relaxation), but is quite different from the approach of Hungerländer & Rendl [44] in some aspects. The main distinctions are the usage of nonstandard semidefinite bounds through some kind of regularization and the dualization of all involved constraints. This leads to theoretically weaker bounds, but also to much cheaper function evaluations and a smooth optimization problem.

Note that adding the rank-one constraint $\text{rank}(X) = 1$ to (4.1) again yields an exact formulation for SRFLP. The key idea is now to replace the rank-one constraint by the so-called *spherical constraint* [61], yielding another exact formulation.

Proposition 4.2.1 ([61, Theorem 1])
Let $X \in \mathcal{S}_p$ be such that $X \succeq 0$ and $\text{diag}(X) = e$. Then we have

$$\|X\| \leq p,$$

and

$$\operatorname{rank}(X) = 1 \quad \Longleftrightarrow \quad \|X\| = p.$$

Therefore, by adding the constraint $\|X\|^2 = n_*^2$ to the semidefinite relaxation (4.1), we obtain the following exact formulation for SRFLP:

$$\begin{array}{ll} \min & \langle C, X\rangle \\ \text{s.t.} & \mathcal{A}(X) \le a \\ & \mathcal{B}(X) = e \\ & X \succeq 0 \\ & \|X\|^2 = n_*^2. \end{array} \tag{4.4}$$

Note that the whole nonconvexity of problem (4.4) is now concentrated in the single nonconvex quadratic equation $\|X\|^2 = n_*^2$ (see [62]).

We now dualize all constraints of problem (4.4). This yields the Lagrangian

$$\mathcal{L}(X; \lambda, \mu, Z, \alpha) \coloneqq \langle C, X\rangle + \lambda^\top(\mathcal{A}(X) - a) + \mu^\top(\mathcal{B}(X) - e) + \frac{\alpha}{2}\left(\|X\|^2 - n_*^2\right) - \langle Z, X\rangle,$$

which is a function of the primal variable $X \in \mathcal{S}_{n_*}$ and the dual variables $(\lambda, \mu, Z, \alpha) \in \mathcal{D} \coloneqq \mathbb{R}_+^{m_I} \times \mathbb{R}^{m_E} \times \mathcal{S}_{n_*}^+ \times \mathbb{R}$. To simplify the notation, we introduce

$$\begin{aligned} c(\lambda, \mu, \alpha) &\coloneqq -a^\top\lambda - e^\top\mu - \frac{\alpha}{2}n_*^2 \\ C(\lambda, \mu) &\coloneqq C + \mathcal{A}^\top(\lambda) + \mathcal{B}^\top(\mu). \end{aligned}$$

The associated concave dual function is then given by

$$\begin{aligned} f(\lambda, \mu, Z, \alpha) &\coloneqq \inf_{X \in \mathcal{S}_{n_*}} \mathcal{L}(X; \lambda, \mu, Z, \alpha) \\ &= c(\lambda, \mu, \alpha) + \inf_{X \in S_{n_*}} \left\{ \frac{\alpha}{2}\|X\|^2 + \langle C(\lambda, \mu) - Z, X\rangle \right\}. \end{aligned} \tag{4.5}$$

By weak duality, $f(\lambda, \mu, Z, \alpha)$ yields a lower bound for the optimal value of (4.4) for all $(\lambda, \mu, Z, \alpha) \in \mathcal{D}$, and hence, also for SRFLP. Again, we are interested in an approximate solution of the dual problem and the corresponding dual bound, i.e., we want to solve

$$z' \coloneqq \sup_{(\lambda,\mu,Z,\alpha)\in\mathcal{D}} f(\lambda, \mu, Z, \alpha). \tag{4.6}$$

The practical usefulness and the tightness of the presented bounds depend on the sign of α. Note that we get $f(\lambda, \mu, Z, \alpha) = -\infty$ for $\alpha < 0$. Therefore, only $\alpha \ge 0$ remains promising. Consider now the special case $\alpha = 0$. In this case, we obtain

$$f(\lambda, \mu, Z, 0) = \begin{cases} -a^\top\lambda - e^\top\mu, & \text{if } C(\lambda, \mu) - Z = 0 \\ -\infty, & \text{otherwise.} \end{cases}$$

Since $Z \succeq 0$, the dual problem (4.6) for $\alpha = 0$ is equivalent to the usual semidefinite dual problem (see [62])

$$\begin{array}{ll} \sup & -a^\top\lambda - e^\top\mu \\ \text{s.t.} & C + \mathcal{A}(\lambda) + \mathcal{B}(\mu) = Z \\ & Z \succeq 0. \end{array}$$

Thus, we have $z' \geq z_{SDP}$. This is not surprising, since we have followed the standard Lagrangian dual for semidefinite programming while also dualizing the additional constraint $\|X\|^2 = n_*^2$, and then setting the corresponding dual multiplier to zero. In fact, the usual semidefinite bound z_{SDP} is also an upper bound for the dual problem (4.6), i.e., $z' = z_{SDP}$. To see this, let $\alpha > 0$ be fixed, and consider the problem

$$\begin{array}{ll} \inf & \langle C, X\rangle - \dfrac{\alpha}{2}\left(n_*^2 - \|X\|^2\right) \\ \text{s.t.} & \mathcal{A}(X) \leq a \\ & \mathcal{B}(X) = e \\ & X \succeq 0. \end{array} \tag{4.7}$$

Problem (4.7) is the primal semidefinite relaxation (4.1) with an additional term in the objective function. Dualizing this problem exactly yields the dual problem (4.6) for $\alpha > 0$ fixed. Since we have

$$\frac{\alpha}{2}\left(n_*^2 - \|X\|^2\right) \geq 0 \tag{4.8}$$

by Proposition 4.2.1, it follows that $z' \leq z_{SDP}$, and hence, $z' = z_{SDP}$.

The inequality (4.8) reveals the spirit of a penalty approach, since any violation of the spherical constraint is penalized in the objective function. When the penalty parameter α vanishes, we get the strongest bound. Hence, the dual problem (4.6) could actually be solved by using (theoretically) efficient methods for semidefinite programming. This includes interior-point methods or the approach by Hungerländer & Rendl [44], see Section 4.1. However, the latter approach leads to a nonsmooth optimization problem to which the bundle method is applied.

Although the corresponding lower bounds are weaker, the case $\alpha > 0$ yields much more interesting computational properties than the case $\alpha = 0$. Additionally, these bounds are only slightly weaker if $\alpha > 0$ is small enough. Since strong duality holds in our setting, one can show that we can get arbitrarily tight bounds (see [62]), i.e.,

$$\lim_{\alpha \searrow 0} \max_{(\lambda,\mu,Z,\alpha)\in\mathcal{D}} f(\lambda, \mu, Z, \alpha) = z_{SDP}. \tag{4.9}$$

4.2.1 Computing the bounds

We will now consider the case $\alpha > 0$ in more detail. It turns out that we have an explicit expression of the dual function f in this case. Moreover, adding the penalty term (4.8) regularizes the dual problem (4.6) for $\alpha > 0$ fixed, i.e., methods of smooth optimization can be applied to it.

Theorem 4.2.2 ([62, Theorem 1])
Let $(\lambda, \mu, Z, \alpha) \in \mathcal{D}$ with $\alpha > 0$ be given. Then the minimum of the Lagrangian $\mathcal{L}(X; \lambda, \mu, Z, \alpha)$ is attained at

$$X^* = \frac{1}{\alpha}\left(Z - C\left(\lambda, \mu\right)\right),$$

and we have

$$f\left(\lambda, \mu, Z, \alpha\right) = c(\lambda, \mu, \alpha) - \frac{1}{2\alpha}\left\|Z - C(\lambda, \mu)\right\|^2.$$

Proof. Since $\alpha > 0$, the Lagrangian, as a function of X, is strongly convex and differentiable. Hence, its unique minimizer X^* is given by

$$\begin{aligned} 0 &\overset{!}{=} \nabla_X \mathcal{L}(X; \lambda, \mu, Z, \alpha) = \alpha X + C(\lambda, \mu) - Z \\ &\iff X = \frac{Z - C(\lambda, \mu)}{\alpha}. \end{aligned}$$

Thus, we get

$$\begin{aligned} f(\lambda, \mu, Z, \alpha) &= \mathcal{L}(X^*; \lambda, \mu, Z, \alpha) \\ &= c(\lambda, \mu, \alpha) + \frac{\alpha}{2} \left\| \frac{Z - C(\lambda, \mu)}{\alpha} \right\|^2 + \left\langle C(\lambda, \mu) - Z, \frac{Z - C(\lambda, \mu)}{\alpha} \right\rangle \\ &= c(\lambda, \mu, \alpha) + \frac{\alpha}{2} \frac{1}{\alpha^2} \left\| Z - C(\lambda, \mu) \right\|^2 - \frac{1}{\alpha} \left\| C(\lambda, \mu) - Z \right\|^2 \\ &= c(\lambda, \mu, \alpha) + \left(\frac{1}{2\alpha} - \frac{1}{\alpha} \right) \left\| Z - C(\lambda, \mu) \right\|^2 \\ &= c(\lambda, \mu, \alpha) - \frac{1}{2\alpha} \left\| Z - C(\lambda, \mu) \right\|^2. \end{aligned}$$

□

We also see that $f(\lambda, \mu, Z, \alpha) = c(\lambda, \mu, \alpha) - \frac{1}{2\alpha} \left\| Z - C(\lambda, \mu) \right\|^2$ is differentiable at any $(\lambda, \mu, Z, \alpha) \in \mathcal{D}$ for $\alpha > 0$. The partial derivatives are given by (see [62])

$$\begin{aligned} \partial_\lambda f(\lambda, \mu, Z, \alpha) &= \frac{1}{\alpha} \mathcal{A}(Z - C(\lambda, \mu)) - a, \\ \partial_\mu f(\lambda, \mu, Z, \alpha) &= \frac{1}{\alpha} \mathcal{B}(Z - C(\lambda, \mu)) - e. \end{aligned}$$

However, for given dual variables (λ, μ, α) with $\alpha > 0$, we also have to deal with the dual variable $Z \succeq 0$. Fortunately, we can explicitly maximize the dual function over Z in this case. For this purpose, we introduce the simplified dual function [62]

$$f(\lambda, \mu, \alpha) \coloneqq \max_{Z \succeq 0} f(\lambda, \mu, Z, \alpha) = \max_{Z \succeq 0} \left\{ c(\lambda, \mu, \alpha) - \frac{1}{2\alpha} \left\| Z - C(\lambda, \mu) \right\|^2 \right\}.$$

The optimal Z^* is given by the uniquely defined solution of

$$\min_{Z \succeq 0} \left\| Z - C(\lambda, \mu) \right\|^2,$$

which is the projection of $C(\lambda, \mu)$ onto the cone of positive semidefinite matrices, i.e.,

$$Z^* = C(\lambda, \mu)_+.$$

Due to (1.3), we obtain

$$Z^* - C(\lambda, \mu) = C(\lambda, \mu)_+ - C(\lambda, \mu) = -C(\lambda, \mu)_-.$$

Therefore, the minimizer of the Lagrangian $\mathcal{L}(X;\lambda,\mu,Z,\alpha)$ becomes

$$X^* = -\frac{C(\lambda,\mu)_-}{\alpha},$$

and the simplified dual function is given by

$$\begin{aligned} f(\lambda,\mu,\alpha) &= c(\lambda,\mu,\alpha) - \frac{1}{2\alpha}\left\|-C(\lambda,\mu)_-\right\|^2 \\ &= c(\lambda,\mu,\alpha) - \frac{1}{2\alpha}\left\|C(\lambda,\mu)_-\right\|^2. \end{aligned} \tag{4.10}$$

Despite the projection onto the cone of negative semidefinite matrices, one can show that $f(\lambda,\mu,\alpha)$ is differentiable at any (λ,μ,α) with $\alpha > 0$, and the partial derivatives are given by (see [62])

$$\begin{aligned} \partial_\lambda f(\lambda,\mu,\alpha) &= -\frac{1}{\alpha}\mathcal{A}(C(\lambda,\mu)_-) - a, \\ \partial_\mu f(\lambda,\mu,\alpha) &= -\frac{1}{\alpha}\mathcal{B}(C(\lambda,\mu)_-) - e. \end{aligned} \tag{4.11}$$

4.2.2 The bounding procedure

We will now show how the above results can be used to compute tight semidefinite bounds for SRFLP. We refer to [50–52] in which bounding procedures based on the presented bounds [62] are proposed. Here, however, we only present the basic ideas of such a bounding procedure and postpone our exact implementation for SRFLP to Section 5.3.

First, recall that we can get arbitrarily close to the usual semidefinite bound z_{SDP}, if we take $\alpha > 0$ small enough and find appropriate dual variables $\lambda \geq 0$ and μ, see (4.9). Second, one can show that the bounds $f(\lambda,\mu,\alpha)$ are monotonic with respect to α in the following sense. For $0 < \alpha \leq \alpha'$, we have (see [62])

$$\max_{\lambda\geq 0,\,\mu} f(\lambda,\mu,\alpha') \leq \max_{\lambda\geq 0,\,\mu} f(\lambda,\mu,\alpha).$$

Therefore, α can be interpreted as a tightness parameter that dictates the quality of the best possible lower bound with respect to the dual variables λ and μ. Choosing a smaller value of α allows to compute stronger bounds. Third, as for the approach by Hungerländer & Rendl [44], the set of inequalities is too large to be maintained as a whole, even for medium-sized problems. Hence, we restrict ourselves to a subset $\mathcal{I}$ of inequalities which we update from time to time. We denote the associated dual function by $f_{\mathcal{I}}(\lambda,\mu,\alpha)$.

For $\alpha > 0$ fixed, the differentiability of $f_{\mathcal{I}}$ motivates to use the partial derivatives (4.11) to find approximately optimal dual multipliers λ and μ. For this, consider the following simpler optimization problem for $\alpha > 0$ and $\mathcal{I}$ fixed [62]:

$$\max_{\lambda\geq 0,\,\mu} f^\alpha_{\mathcal{I}}(\lambda,\mu) \coloneqq \max_{\lambda\geq 0,\,\mu} f_{\mathcal{I}}(\lambda,\mu,\alpha). \tag{4.12}$$

Problem (4.12) is a convex optimization problem (or equivalent to), since the feasible set is convex and the objective is to maximize a concave function over this set. Moreover, the

objective function $f_{\mathcal{I}}^{\alpha}$ is differentiable and the partial derivatives are given in (4.11). Hence, any efficient method of nonlinear optimization that can handle the box constraints $\lambda \geq 0$ can be used to solve (4.12) approximately. Although the so-called second-order semismooth Newton method could be used for this purpose (see [52]), it is suggested in [62] to use a quasi-Newton method, which easily can handle any number of constraints. In particular, the limited-memory quasi-Newton code `L-BFGS-B` [63,86] is used in [50–52], due to its simplicity and robustness. We will also use the `L-BFGS-B` solver in our experiments.

Throughout its optimization process, the quasi-Newton method requires many evaluations of $f_{\mathcal{I}}^{\alpha}$ and its gradient for given α, (λ, μ) and $\mathcal{I}$. As the matrix $C(\lambda, \mu)$ can efficiently be constructed, the by far most expensive part is the projection onto the cone of negative semidefinite matrices. This can be done by using the explicit formula (1.2), i.e., by computing a partial eigendecomposition of the matrix $C(\lambda, \mu)$. In order to do so, the routine `DSYEVR` of the Intel Math Kernel Library (`MKL`) is called in [50–52]. This routine requires $\mathcal{O}(p^3)$ arithmetic operations for a matrix of dimension p, i.e., the effort is $\mathcal{O}(n^6)$ for a `SRFLP` instance with n facilities.

Although the quality of the bounds is controlled by the tightness parameter α and the mentioned routines are known to be efficient and robust in general, we should not take α unnecessarily small. The expressions of the simplified dual function (4.10) and its gradient (4.11) indicate that problem (4.12) becomes ill-conditioned if α tends to zero. Since the gradient has a very sharp behavior for very small values of α, we have to expect a huge number of `L-BFGS-B` iterations (see [62]).

To overcome this complication and to get a good ratio of tightness to computing time, the authors in [50–52] use an adaptive approach, in which α starts at a fairly large value and is then iteratively reduced if necessary. An outline of their iterative bounding procedure is shown in Algorithm 1. For a given subset $\mathcal{I}_k$ of inequalities and values $\alpha_k > 0$, $\varepsilon_k > 0$ in iteration k, the `L-BFGS-B` solver is warm-started from previous dual variables $(\lambda_{k-1}, \mu_{k-1})$ to compute (λ_k, μ_k) such that the primal variable

$$X_k := -\frac{(C + \mathcal{A}_{\mathcal{I}_k}(\lambda_k) + \mathcal{B}(\mu_k))_-}{\alpha_k}$$

satisfies the termination criterion

$$\max\left\{\left\|[\mathcal{A}(X_k) - a]_-\right\|_\infty, \|\mathcal{B}(X_k) - e\|_\infty\right\} < \varepsilon_k.$$

The new set of inequalities $\mathcal{I}_{k+1}$ is then constructed from $\mathcal{I}_k$ by adding inequalities that are violated by the current iterate X_k, and by removing inequalities that are not active with respect to their corresponding dual variable λ_k. For iteration $k+1$, α_{k+1} and ε_{k+1} are chosen such that $\alpha_{k+1} \leq \alpha_k$ and $\varepsilon_{k+1} \leq \varepsilon_k$. One can show (see [52]) that under mild assumptions, the sequence of generated lower bounds $f_{\mathcal{I}_k}^{\alpha_k}(\lambda_k, \mu_k)$ by Algorithm 1 converges to the usual semidefinite bound z_{SDP}, if $\alpha_k \to 0$, $\varepsilon_k \to 0$ and $\mathcal{I}_k \to \mathcal{I}^*$, where $\mathcal{I}^*$ contains all inequalities that are active at the optimum.

Algorithm 1: Outline of the bounding procedure

Input: $\alpha > 0$, $\varepsilon > 0$
Output: Sequence of lower bounds $f_{\mathcal{I}_k}(\lambda_k, \mu_k, \alpha_k)$
Initialization: $\lambda_0 \leftarrow \mathbf{0}$, $\mu_0 \leftarrow \mathbf{0}$, $\alpha_1 \leftarrow \alpha$, $\varepsilon_1 \leftarrow \varepsilon$, $\mathcal{I}_1 \leftarrow \emptyset$
for $k = 1, 2, \ldots,$ **do**
  Use the quasi-Newton code `L-BFGS-B`, warm-started from $(\lambda_{k-1}, \mu_{k-1})$, to maximize $f_{\mathcal{I}_k}^{\alpha_k}$, i.e., compute (λ_k, μ_k) such that

$$\max\left\{\left\|[\mathcal{A}_{\mathcal{I}_k}(X_k) - a]_+\right\|_\infty, \|\mathcal{B}(X_k) - e\|_\infty\right\} < \varepsilon_k,$$

  where

$$X_k \leftarrow -\frac{(C + \mathcal{A}_{\mathcal{I}_k}(\lambda_k) + \mathcal{B}(\mu_k))_-}{\alpha_k}.$$

  Output the lower bound $f_{\mathcal{I}_k}(\lambda_k, \mu_k, \alpha_k)$.
  Choose an appropriate set of inequalities $\mathcal{I}_{k+1}$ with the help of X_k and λ_k; initialize the multipliers of new inequalities to zero.
  Choose $\alpha_{k+1} \leq \alpha_k$ and $\varepsilon_{k+1} \leq \varepsilon_k$.

4.3 Comparison of the methods

In this section, we want to point out some general similarities and differences between the approach suggested by Hungerländer & Rendl [44] (see Section 4.1) and our proposed approach for `SRFLP` (see Section 4.2). We mainly focus on theoretical properties such as hypothetical accuracy, running time and memory requirements for a single function evaluation or scalability with respect to large-scale `SRFLP` instances. For an in-depth numerical comparison, we refer to Chapter 6.

A first similarity is that both methods can handle many constraints due to the application of Lagrangian relaxation. This allows the usage of (SDP_1) as the underlying basic semidefinite relaxation, whereas only (SDP_0) could be used for large instances when interior-point methods are applied directly (see Section 3.3). Additionally, many inequalities can be added in the solution process to strengthen the bound. Of course, these inequalities are not limited to those involved in (SDP_3). The possibility of handling a large set of inequalities also encourages us to look for further valid inequalities. We address this topic in Chapter 5.

The possibility to handle a lot of inequalities also involves some hidden costs. While interior-point methods have a very high accuracy, this looks a bit different with the presented approaches based on Lagrangian duality. It is clear that it is probably not worthwhile to improve the semidefinite relaxations only theoretically by adding more inequalities. The practical value of any relaxation also depends on its practical feasibility. A theoretically weaker relaxation may perform better in practice, if it can be solved with higher accuracy in a smaller amount of time.

From a theoretical and algorithmic point of view, this is a major difference between the

approach by Hungerländer & Rendl [43, 44] and our proposed approach. The bundle method approach is more direct in the sense that no penalty term is used, i.e., it tries to approximate the usual semidefinite bound more directly. In contrast, our proposed approach depends on a tightness parameter and solves several different optimization problems, for which the optimal value of these problems comes closer and closer to the usual semidefinite bound.

The accuracy, and especially the needed running time to achieve a given accuracy, heavily depends on the optimization algorithms which are used to approximately solve the occurring optimization problems. Hungerländer & Rendl [44] work with a nonsmooth optimization problem which is solved by a specific version of the bundle method. Therefore, this approach only uses a first-order method which typically has a weak convergence behavior. In general, we should expect to get an approximate solution having a small relative error. Unfortunately, small inaccuracies may prevent a successful solution of poorly scaled instances, even if the semidefinite relaxation is actually sufficient. If the data of a given instance has relatively large numbers, a small relative error may not be sufficient to prove global optimality, since the calculated lower bound must have an absolute error of less than 1. In this case, more accurate methods are clearly favored.

Our suggested approach leads to a sequence of smooth optimization problems (see 4.12), for which explicit expressions of the partial derivatives are available. These optimization problems are solved with the quasi-Newton code `L-BFGS-B` [63, 86]. On the one hand, this quasi-Newton method is more efficient and has a better convergence behavior. It lies somewhere between first and second-order method (see [52]) and can compute more accurate solutions in less iterations than the bundle method. On the other hand, the tightness of the bounds depends on the penalty parameter α. As already mentioned, by taking α sufficiently small, we can get arbitrarily close to the usual semidefinite bound. Nonetheless, we have to take care of the sharp behavior of the gradient for very small values of α. However, even if this results in more iterations, our suggested approach still has a higher potential to compute more precise bounds than the bundle method, if necessary.

In addition to the more efficient quasi-Newton method, our approach has a much simpler dual function than the bundle approach by Hungerländer & Rendl [44]. The explicit expression of the dual function (4.10) and the use of robust and highly optimized linear algebra routines allow to perform much more adaptations of the dual variables in the same period of time. Denoting the number of facilities by n, a semidefinite program of dimension $\mathcal{O}(n^2)$ with $\mathcal{O}(n^2)$ constraints must be solved for a function and subgradient evaluation of the bundle method. Whereas with our approach, only a matrix of the same dimension has to be projected onto the cone of negative semidefinite matrices. To illustrate the different computational efforts, Table 4.1 compares the average running times for the respective function evaluations for $n \in \{30, 50, 70, 100\}$. The running times for the bundle approach are reported in [44]. Therein, a standard interior-point method on an Intel Xeon 5160 processor with 3 GHz is used to solve the semidefinite programs. Our results are carried out on an Intel Xeon E5-2640V4 processor with 2.4 GHz, using one or four cores respectively. We call the routine `DSYEVR` of the Intel Math Kernel Library (`MKL`) to perform the eigendecomposition (see Section 4.2).

Independently from the different hardware, the conclusion is obvious. The penalty method computes the function and gradient evaluations much quicker. Using only one core, about

n	BM [44]	PM (one core)	PM (four cores)
30	3	0.017	0.012
50	40	0.262	0.101
70	500	1.721	0.572
100	3000	19.282	6.986

Table 4.1: Average running times (in seconds) to perform a single function and (sub-)gradient evaluation for the bundle method (BM) and the penalty method (PM).

150 evaluations can be done while the bundle method accomplishes only one evaluation in the same period of time. Note that this is true for every value of n. Additionally, we can see that the computation times of the penalty method can be greatly reduced by using modern CPU architectures with many cores. In comparison to the expensive solution of the semidefinite program in the bundle approach, the computational effort for optimizing the dual variables with the bundle method is negligible. The quasi-Newton solver `L-BFGS-B` requires between $\mathcal{O}(mn')$ and $\mathcal{O}(m^2n')$ arithmetic operations for a single iteration, where m denotes the predefined number of memory corrections and n' is the number of dual variables [63,86]. Hence, for fixed m, the computational effort is linear in the number of dual variables. Ironically, the computational effort reaches its maximum, when all bounded variables are at their bounds, i.e., all inequalities are inactive. This suggests to avoid the excessive inclusion of too many unnecessary inequalities, at least for relatively small instances where the projection onto the cone of negative semidefinite matrices can be performed very efficiently.

Another property in favor of the penalty approach is that the quasi-Newton solver `L-BFGS-B` can easily be warm-started from previous known dual variables. Its cheap iterations and the efficient implemented detection of free variables (see [63, 86]) make it possible to determine inactive and useful inequalities very quickly. This encourages us to update the current set of inequalities much more frequently than with the bundle approach. Especially on very large instances, say $n \geq 70$, the bundle approach has some severe limitations to find a suitable selection of inequalities. Due to the very high cost of function evaluations and the weak convergence behavior of the bundle method, the separation routine can effectively be executed very rarely compared to the penalty approach. Therefore, we expect the penalty approach to perform better, when it comes to solving very large instances. Additionally, this opens the door to to tighten the relaxations even further by considering more inequalities, since the approach seems to have the potential of being both fast and accurate (if necessary).

Now consider the memory requirements of the two approaches. Clearly, both approaches require at least $\mathcal{O}(n^3)$ memory space for the dual variables, linearly increasing with the number of present inequalities. For a fixed number m of memory corrections, the `L-BFGS-B` solver only needs memory space linear in the number of dual variables involving a small constant. The exact requirement for the bundle method itself depends on how the occurring quadratic subproblems are solved. However, it is much higher in any case. An evaluation of the dual function requires $\mathcal{O}(n^4)$ memory space for both approaches. However, solving the semidefinite program with an interior-point method incorporates a much higher constant factor than the projection onto the cone of negative semidefinite matrices. Basically, the latter needs memory space twice as much as for storing the primal variable X. Hence the

Property	IPM	BM	PM
accuracy (theoretical)	very high	low	high
can handle many constraints	no	yes	yes
effort for a single function evaluation	very high	high	low
memory requirements	high	low	very low
yields lower bounds	yes	yes	yes
direct access to primal variable X	yes	yes	yes
subset of inequalities can be updated frequently for large-scale instances	no	no	yes

Table 4.2: Comparison of properties for different solution approaches for the semidefinite relaxations for `SRFLP`: interior-point method (IPM) approach, bundle method (BM) approach and penalty method (PM) approach.

overall memory requirement for the penalty method is very efficient, and increases linearly with the number of inequalities.

All in all, both methods require relatively few memory space, but the penalty approach is more modest than the bundle approach. The biggest distinction is definitely caused by the different running times for an evaluation of the dual function. The bundle approach seems to spend substantially more time (by a huge constant factor) on a single evaluation. In addition to the higher number of evaluations, the quasi-Newton method of the penalty approach is theoretically superior to the bundle method in terms of accuracy and convergence behavior. In view of solving very large `SRFLP` instances, it is more than questionable that the bundle method can achieve the desired accuracy in reasonable time, due to the high cost of function evaluations and the huge amount of inequalities. The penalty approach presumably offers the possibility to find reasonable good subsets of inequalities in much less time. We have summarized some important properties of the discussed approaches in Table 4.2.

Chapter 5

A Novel Approach for SRFLP

In this chapter, we present our suggested approach for SRFLP in more detail. In Section 5.1, we investigate how the existing semidefinite relaxations can be improved. For this purpose, we will consider certain classes of facet defining inequalities for the cut polytope $\mathcal{P}_C$. By selecting a suitable subset of them, we introduce a new strengthened semidefinite relaxation for SRFLP. This relaxation yields much stronger lower bounds when compared to prior semidefinite relaxations such as (SDP_3) by Hungerländer & Rendl [44], especially for large-scale instances. In addition, it has a positive effect on the running time of our proposed solution method, i.e., the stronger relaxation with more constraints leads to shorter computing times. Additionally, in Section 5.1 we propose two heuristic separation routines for huge classes of inequalities that cannot be enumerated explicitly.

In Section 5.2, we discuss and present several primal heuristics that use (approximate) solutions of the semidefinite relaxations to compute good feasible layouts. Moreover, a concise overview of our practical implementation for solving our proposed relaxations is given in Section 5.3. Since this only yields lower bounds for SRFLP, Section 5.4 is concerned with the embedding of our semidefinite relaxations into an exact branch-and-bound algorithm. For this, we illustrate how an efficient branch-and-bound approach should be designed and we also propose two branching rules which are particularly suited for SRFLP.

5.1 A considerably improved relaxation

Our discussion in Section 4.3 reveals that our suggested solution method for the semidefinite relaxations (see Section 4.2) is well-suited to deal with a huge number of constraints. It is very efficient in finding a good subset of inequalities quickly while also maintaining reasonable accuracy. On the one hand, we aim to find a much tighter semidefinite relaxation than (SDP_3) that is computational feasible, i.e., it should contain no more than $\mathcal{O}(n^6)$ constraints. On the other hand, we also do not refrain from considering larger classes of inequalities for which we propose heuristics to separate them heuristically. To the best of our knowledge, such a semidefinite approach has never been applied to SRFLP (or any quadratic ordering problem in general) before, see also [41, 43].

First steps towards a tighter semidefinite relaxation were done in [41] for the general quadratic ordering problem. Similar to (3.28), the idea of Lovász and Schrijver [59] can be applied to pairs of the 3-cycle-inequalities (3.4). This yields the inequalities

$$\begin{aligned} -1 - y_{ij} - y_{jk} + y_{ik} \leq y_{lm} + y_{mo} - y_{lo} + y_{ij,lm} + y_{ij,mo} - y_{ij,lo} + y_{jk,lm} + y_{jk,mo} \\ - y_{jk,lo} - y_{ik,lm} - y_{ik,mo} + y_{ik,lo} \leq 1 + y_{ij} + y_{jk} - y_{ik}, \\ -1 + y_{ij} + y_{jk} - y_{ik} \leq y_{lm} + y_{mo} - y_{lo} - y_{ij,lm} - y_{ij,mo} + y_{ij,lo} - y_{jk,lm} - y_{jk,mo} \\ + y_{jk,lo} + y_{ik,lm} + y_{ik,mo} - y_{ik,lo} \leq 1 - y_{ij} - y_{jk} + y_{ik}, \end{aligned} \tag{5.1}$$

for all $1 \leq i < j < k \leq n$, $1 \leq l < m < o \leq n$. By using PORTA [18] to compute the complete outer description of $\mathcal{P}_{LQO}$ in low dimensions, it was shown in [41] that some of these $\approx \frac{n^6}{9}$ constraints are indeed facets of $\mathcal{P}_{LQO}$ for $n = 4$. However, it remains an open question whether the Lovász-Schrijver-cuts (3.28) and (5.1) are facet defining for any linear quadratic ordering polytope in higher dimension [41]. Other valid inequalities for $\mathcal{P}_{LQO}$ can be obtained by inspecting the betweenness polytope $\mathcal{P}_{BTW}$. All facets of $\mathcal{P}_{BTW}$ for $n \in \{3, 4, 5\}$ were computed in [66] by using PORTA [18]. These facets can be rewritten as quadratic constraints in bivalent ordering variables (3.1) by using the transformations (3.2). It is suggested in [41] to add a subset of the resulting $\mathcal{O}(n^5)$ constraints alongside (5.1) to the semidefinite relaxation (SDP_3), since preliminary experiments would show that this pays off for all kinds of betweenness problems which also includes SRFLP.

However, we do not recommend to do this for SRFLP, at least when our proposed solution method is used. First, we do not observe any noteworthy improvement on the lower bounds when the Lovász-Schrijver-cuts (3.28) or (5.1) are used. They even have a negative impact on the required number of function evaluations which increases the overall computing time. Second, the inequalities that are derived from the facets of $\mathcal{P}_{BTW}$ for $n = 5$ sometimes marginally improve the lower bounds, but always significantly increase the running times. Additionally, their inclusion is never sufficient to solve a SRFLP instance when the sole set of triangle inequalities is insufficient. We observed that when these inequalities are added in the bounding procedure, they are then almost often quickly removed, since their dual multipliers stay near to zero. This hints at a very unstable behavior. As a result, we completely omit any Lovász-Schrijver-cuts and facets of the betweenness polytope $\mathcal{P}_{BTW}$ in low dimensions in our implementation. In the following, we present a much better alternative which also has a close connection to the betweenness polytope $\mathcal{P}_{BTW}$.

5.1.1 Facets of the cut polytope

As already done in Section 3.2 by introducing the triangle inequalities (3.22), we further exploit the structure of the underlying cut polytope $\mathcal{P}_C$. The facial structure of $\mathcal{P}_C$ has been studied extensively in the literature, e.g., see [21]. A remarkably general class of valid inequalities, the so-called *gap inequalities* [56], is given by

$$\left\langle bb^\top, X \right\rangle \geq B \tag{5.2}$$

for any $b \in \mathbb{Z}^n$ with $B \coloneqq \min\left\{ \left(b^\top x\right)^2 : x \in \{-1, 1\}^n \right\} > 0$. Inequalities for which the right hand side B equals to one, are called *hypermetric inequalities*. If $b \in \{-1, 0, 1\}^n$ and b has

an odd number of nonzero entries, we obtain the very important class of *clique inequalities*. In the following, we introduce certain classes of valid inequalities for any $X \in \mathcal{P}_C$ which can be derived from (5.2). Since $\mathcal{P}_{LQO} \subset \mathcal{P}_C$, these inequalities can also be used to strengthen the semidefinite relaxation (SDP_3) which is formulated over $\mathcal{P}_{LQO}$.

For a subset of three rows (or columns) $\{p_1, p_2, p_3\}$ of X, we have the inequalities

$$\sum_{1 \le i < j \le 3} \delta_i \delta_j X_{p_i,p_j} \ge -1,$$

where $\delta_k \in \{-1, 1\}$, $k = 1, 2, 3$. Note that this yields four different inequalities for each choice of $\{p_1, p_2, p_3\}$ and these are exactly the triangle inequalities (3.22). For five pairs $\{p_1, \ldots, p_5\}$ and all choices of $\delta_k \in \{-1, 1\}$, $k = 1, \ldots, 5$, we obtain the 16 following *pentagonal inequalities*:

$$\sum_{1 \le i < j \le 5} \delta_i \delta_j X_{p_i,p_j} \ge -2.$$

Analogously, for seven pairs $\{p_1, \ldots, p_7\}$ and all choices of $\delta_k \in \{-1, 1\}$, $k = 1, \ldots, 7$, we get the 64 inequalities

$$\sum_{1 \le i < j \le 7} \delta_i \delta_j X_{p_i,p_j} \ge -3,$$

which we call *heptagonal inequalities* for convenience. Moreover, we also consider the hypermetric *hexagonal inequalities*. They are given by

$$2 \sum_{i=2}^{6} \delta_1 \delta_i X_{p_1,p_i} + \sum_{2 \le i < j \le 6} \delta_i \delta_j X_{p_i,p_j} \ge -4,$$

where $\delta_k \in \{-1, 1\}$, $k = 1, \ldots, 6$, and $p_1, \ldots, p_6$ is any 6-tuple of different pairs.

There are some theoretical and computational results indicating that the triangle, pentagonal, hexagonal and heptagonal inequalities are by far the most attractive inequalities for semidefinite max-cut relaxations, see e.g. [28]. For $n \in \{3, 4\}$, all facets of $\mathcal{P}_C$ are given by the triangle inequalities. For $n = 5$, all facets are induced by the triangle or pentagonal inequalities, and also the hexagonal inequalities for $n = 6$. For $n \ge 7$, the structure and number of facet classes are more complex, but always include the heptagonal inequalities beside all other mentioned inequalities. Additionally, it is most likely that the above facets have the shortest distance to the barycenter of $\mathcal{P}_C$ in any dimension [21].

Some indication why the basic semidefinite relaxation (SDP_1) and especially (SDP_2) already yield excellent lower bounds for SRFLP is given in [8]. It can be shown that any solution of (SDP_1) for a SRFLP instance with n facilities automatically satisfies $4\binom{n}{3}$ triangle inequalities. Moreover, any solution of (SDP_2) automatically satisfies $90\binom{n}{4}$ pentagonal inequalities and $192\binom{n}{4}$ hexagonal inequalities. However, it is obvious that we cannot include all pentagonal, hexagonal and heptagonal inequalities in our semidefinite relaxations, since there are already $16\binom{\binom{n}{2}+1}{5} \approx \frac{1}{240} n^{10}$ pentagonal inequalities. To obtain a more economical approach, we choose a suitable subset of only $\mathcal{O}(n^6)$ pentagonal inequalities which can be enumerated explicitly, and separate the rest heuristically.

For this purpose, reconsider the betweenness approach in Section 2.3 and the exponential class of inequalities (2.16). It is common knowledge that these inequalities correspond to clique inequalities of the cut polytope (see e.g. [7]). For $\beta = 4$ we obtain a subset of the triangle inequalities (see the proof Proposition 3.2.1), for $\beta = 6$ we obtain a subset of the pentagonal inequalities and so on. Since it was proven that the inequalities (2.16) always induce facets of $\mathcal{P}_{BTW}$ (see [76]), it is a promising idea to include them in our semidefinite relaxations. For this, consider the particular case $\beta = 6$. Suppose that we are given any $R = \{i, j, k, l, m, r\} \subseteq [n]$ and any partition (S, T) of $R \setminus \{r\}$ with $|S| = 3$. Using the transformations (3.2), we then see that the corresponding inequality (2.16) is exactly a pentagonal inequality on the pairs (i, r), (j, r), (k, r), (l, r), (m, r). Up to symmetry, each choice of the partition (S, T) yields a pentagonal inequality on the above pairs but with different signs. Note that we only get 10 of the 16 different types of pentagonal inequalities, which is due to the restriction $|S| = 3$. Omitting this restriction yields all 16 types, but some of them do not define a facet of $\mathcal{P}_{BTW}$. However, our results show that all 16 types are worthy in practice.

Interpreting the pairs (i, r), (j, r), (k, r), (l, r), (m, r) as the edge set of an undirected graph yields a star with the six vertices $\{i, j, k, l, m, r\}$ and the center vertex r. Due to this property, we call pentagonal inequalities with the above structure *starlike pentagonal inequalities*. Each subset of $[n]$ with six facilities yields six different stars which again give 16 different starlike pentagonal inequalities each. Hence, their total number is $96\binom{n}{6}$ which is computational feasible. With the notation in Section 3.2, the foundation of our proposed semidefinite relaxations is then given by

$$\begin{aligned} \min \quad & \langle C, Y \rangle + K \\ \text{s.t.} \quad & Y_{ij,jk} - Y_{ij,ik} - Y_{ik,jk} = -1, \qquad i, j, k \in [n],\ i < j < k, \\ & \operatorname{diag}(\overline{Y}) = e, \\ & \overline{Y} \in \mathcal{M}, \\ & Y \in \mathcal{P}^*, \\ & \overline{Y} \succeq 0, \end{aligned} \tag{SDP$_4$}$$

where

$$\mathcal{P}^* := \left\{ Y \in \mathbb{R}^{\binom{n}{2} \times \binom{n}{2}} : Y \text{ satisfies all starlike pentagonal inequalities} \right\}.$$

Analogous to Proposition 3.2.1, we have the following result.

Proposition 5.1.1
(SDP$_4$) is at least as strong as the linear relaxation of model (BTW) enhanced by the cutting planes (2.16) with $\beta = 6$.

Note that the starlike pentagonal inequalities are only a subset of all pentagonal inequalities that involve exactly six different facilities. An obvious way to tighten (SDP$_4$) even further would be to include all missing pentagonal inequalities of the latter class, leading again to a relaxation with only $\mathcal{O}(n^6)$ constraints. However, their number involves an enormous constant factor which makes them unappealing in practice, especially for smaller instances. To demonstrate this, Table 5.1 lists the number of triangle inequalities and various subclasses

n	triangle inequalities	pentagonal inequalities		
		starlike	exactly 5 facilities	exactly 6 facilities
20	4,572,540	3,720,960	88,558,848	1,185,125,760
40	316,367,480	368,484,480	3,758,541,696	117,362,306,880
60	3,696,820,820	4,806,130,560	31,196,156,544	1,530,752,583,360
80	21,036,328,560	28,848,019,200	137,316,571,392	9,188,094,115,200
100	80,858,246,700	114,437,030,400	430,042,314,240	36,448,194,182,400

Table 5.1: Number of triangle inequalities, starlike pentagonal inequalities, pentagonal inequalities with exactly five facilities and pentagonal inequalities with exactly six facilities for $n \in \{20, 40, 60, 80, 100\}$.

of pentagonal inequalities for different values of n. We can see that the number of starlike pentagonal inequalities is always comparable to the number of triangle inequalities. Hence, their inclusion does not increase the running time of exact separation that much. The number of pentagonal inequalities with exactly five and particularly six facilities are much higher. Therefore, they should only be separated very rarely, if at all. Our tests indicate that their usage is not justified with respect to the running time, since their impact on the lower bound is quite small. Another idea for a promising subclass of pentagonal inequalities is to extend the 3-cycle-equations (3.5) with two additional pairs. Although this yields $\mathcal{O}(n^7)$ constraints, their number is still considerably smaller than those of pentagonal inequalities with exactly six facilities.

5.1.2 Heuristic separation

So far, we have only considered types of inequalities that can be separated exactly in reasonable time by enumeration. With this in mind, (SDP_4) yields excellent lower bounds and is a great backbone for our semidefinite relaxations, but it can still be improved by other pentagonal, hexagonal or heptagonal inequalities. Since the separation of clique inequalities (or hypermetric inequalities in general) is $\mathcal{NP}$-hard [21], we have to use heuristic procedures to separate them. Typically, these procedures would start with suitable choices of triangle inequalities and then add further pairs to them [36]. Such an approach is suggested in [41] for the quadratic ordering problem. It consists of determining a set of $\mathcal{O}(n^3)$ (almost) violated triangle inequalities which is then extended to pentagonal inequalities by enumeration. Hence, the overall effort is $\mathcal{O}(n^7)$. Clearly, choosing only $\mathcal{O}(n^2)$ triangle inequalities reduces the effort to $\mathcal{O}(n^6)$. There are different reasons why this approach is not a great idea in our setting.

Note that all types of inequalities have different importance, triangle inequalities are the most important ones, pentagonal inequalities are second and so on. That is why we restrict ourselves to triangle, pentagonal, hexagonal and heptagonal inequalities only, although there are much more known facets of the cut polytope. Especially for inequalities other than triangle inequalities, we want to find and include only the most-violated ones. Unfortunately, the proposed idea does a poor job in finding these. First, notice that it is not easy to transfer the idea to hexagonal or heptagonal inequalities. We have tested a lot of strategies that

are mixtures of choosing pairs from already present inequalities, picking random pairs and enumerating the remaining ones. Overall, we observed that this kind of approach does not lead to satisfying results. Second, our proposed bounding procedure (see Section 4.2) is designed to solve the occurring subproblems only approximately. Therefore, the iterates X_k typically violate a huge number of constraints, even those which are already present. It may happen that the mentioned heuristic produces the same pentagonal inequalities again and again. Hence, we do not find good inequalities if we only use the structure of already included inequalities.

We guess that the starlike pentagonal inequalities in addition to the triangle inequalities already greatly ensure the structural properties of the underlying problem. Therefore, we want our heuristics to be a bit more generic, i.e., they should fulfill the following conditions:

- yielding a sampling of all inequalities,
- finding many strongly violated inequalities,
- returning different inequalities on each run.

To achieve this, we propose two very simple randomized heuristics. They yield excellent results and their running time can be controlled by a given time limit. This makes them easy to use for different problem sizes and higher time limits should give more promising results. We state them here in terms of pentagonal inequalities, but they can also be used for other types of inequalities. The first heuristic consists of the following two steps as long as the time limit is not reached:

1. Randomly divide the whole set of pairs into chunks of $\frac{n}{2}$ pairs.
2. For each chunk of pairs, enumerate and separate all types of pentagonal inequalities on all pairs in it.

This simple heuristic clearly finds different inequalities on each run and pays particularly attention to the first condition, i.e., it is a kind of sampling algorithm. Compared to the separation of triangle and starlike pentagonal inequalities, it can be run hundreds of times. The second heuristic is especially concerned with finding strongly violated inequalities. It implements a 1-opt local search algorithm. For each set of five different pairs $\{p_1, \ldots, p_5\}$, we assign the function value

$$f\left(\{p_1, \ldots, p_5\}\right) = \min_{\delta_k \in \{-1,1\}} \left\{ \sum_{1 \le i < j \le 5} \delta_i \delta_j X_{p_i, p_j} \right\}$$

to it. We define the neighborhood of $\{p_1, \ldots, p_5\}$ as the set of pairs that differ in exactly one pair. The second heuristic then consists of the following two steps:

1. Randomly choose five different pairs $\{p_1, \ldots, p_5\}$.
2. Starting from $\{p_1, \ldots, p_5\}$, find a set of pairs which is locally optimal with respect to f and separate all types of pentagonal inequalities of all occurring neighbors during the execution.

The computational cost of this heuristic is not as high as one may think, and we typically restart it several thousand times. Note that a similar idea was independently proposed in [31]. We remark that the second heuristic should be implemented as randomly as possible, i.e., the neighborhood should be examined in random order. We can observe much better results this way when compared to a fixed order.

In all of our computational results in Chapter 6, we use the semidefinite relaxation (SDP_4) as the foundation and separate all inequalities therein by enumeration. We also show how the lower bounds can be improved by considering other pentagonal, hexagonal and heptagonal inequalities, separated by our heuristics. We connect the time limits for these heuristics to the time needed to separate all triangle and starlike pentagonal inequalities.

5.2 Primal heuristics

In this section, we address the important task of finding good feasible layouts, i.e., finding upper bounds for SRFLP. In the best case, our lower bounds can then be used to prove the optimality of a known solution. Beside strong dual bounds, primal heuristics are also an essential component for the efficiency of any branch-and-bound algorithm based an mathematical programming. Many heuristics and metaheuristics for SRFLP were proposed in the literature in recent years. These include swarm algorithms such as ant colony optimization [30, 80] and particle swarm optimization [75], local search metaheuristics such as simulated annealing [70], tabu search [47, 74] and other variants [2, 46, 69], population based algorithms [13, 20, 48, 49] and more problem dependent heuristics [22, 53, 54, 67]. Quite interestingly, these heuristics were often only tested on instances with $n \leq 110$, which seems to be very low for heuristic approaches, but demonstrates the hardness of SRFLP. Since our main goal is to solve as largest instances as possible to optimality, we will only concentrate on heuristics which exploit knowledge of our semidefinite relaxations.

For this purpose, we assume that we are given a semidefinite matrix X which arises in any solution method for any semidefinite relaxation for SRFLP. We neither require that X is an optimal solution of the relaxation, nor that X is feasible for it. The latter will happen when we use any approach based on Lagrangian duality. The semidefinite matrices X_k, which arise during our iterative bounding procedure, even may violate the box constraints of the elliptope. Anyhow, we expect a better performance of our heuristics when X can be seen as an approximate solution of the relaxation. We now aim to construct a feasible ordering, i.e., a permutation, or a feasible vector of ordering variables which represents such an ordering.

A first simple idea, assuming that we work with the linear quadratic ordering polytope, would be to extract the first column of X, yielding a vector of (fractional) ordering variables x. We can then apply any rounding strategy to obtain a ± 1-vector. If all entries of x are in $[-1, 1]$, we can also define $\tilde{x}_k = \frac{x_k+e}{2}$ and interpret the resulting entries as probabilities that the corresponding ordering variable will take the value $+1$. If X is very close to the optimal solution of SRFLP, this idea may be absolutely sufficient and works well for small instances. However, there are some points of criticism for this approach. First, we also want to obtain good feasible orderings for quite large instances where X can be farther afield from the optimal solution. Second, the constructed ± 1-vector in general does not satisfy

the 3-cycle-equations (3.5), i.e., it does not represent a valid ordering in this case. Third, in view of the fact that X (if feasible for the relaxation) is a correlation matrix, this approach does not exploit the global connection of all entries due to the semidefiniteness of X. We are more interested in heuristics that are more reliable and exploit all information hidden in the structure of X.

5.2.1 SDP-based heuristics

A more advanced heuristic was proposed in [6] and was later improved in [9]. It exploits the specific bijection constructed in Section 3.1 between the set of all ± 1-vectors representing valid orderings and the set of all permutations. It also yields many heuristic candidate solutions, more precisely, one for each pair ij of facilities. Let $x_{ij,kl}$ be the entries of X and let be ij any row of X. Then consider the n values

$$\omega_k^{ij} = \frac{1}{2}\left(n+1+\sum_{\ell\in[n],\ k\neq l} x_{ij,kl}\right), \quad k \in [n]. \tag{5.3}$$

Due to the bijection, the values ω_k^{ij} are all distinct and define a permutation of $[n]$, if X is rank-one [9]. However, $\operatorname{rank}(X) > 1$ will be more likely the case. Nonetheless, a permutation can be obtained by sorting the values ω_k^{ij} in either increasing or decreasing order. Due to symmetry, the objective value is the same in both cases. Applying this idea to each row yields $\binom{n}{2}$ possibly different layouts.

An advantage of this heuristic is its simplicity. Given the matrix X, the computational effort is only $\mathcal{O}(n^2)$ per row and thus $\mathcal{O}(n^4)$ in total. Moreover, it always yields feasible layouts. Since the dominant part of the solution of any semidefinite relaxations is at least $\mathcal{O}(n^6)$ for each approach, we could spent much more time on heuristics. Hence, a disadvantage is that the heuristic is deterministic and must be modified in some way to produce more different solutions. The rigidity could be relaxed by introducing some kind of random biases in the entries of X. However, the heuristic does not fully exploit the structure of X, since it considers each row isolated from all others. For this reason, we have not implemented this heuristic directly, but borrow the idea of sorting some values from it.

We use a similar approach to that of [41, 43, 44] by taking advantage of the connection to the max-cut problem. Since our semidefinite relaxations correspond to relaxations of the max-cut problem with additional constraints (3.5), it seems natural to adapt the famous hyperplane rounding heuristic by Goemans and Williamson [26, 27], which is depicted in Algorithm 2. The most expensive part of this randomized rounding procedure is to compute the Cholesky factorization $X = WW^\top$ in step 1. For a SRFLP instance with n facilities, this requires $\mathcal{O}(n^6)$ arithmetic operations. Compared to the lower bound computation, this is still negligible. Actually, if we use the proposed bounding procedure (see Section 4.2), the Cholesky factorization need not to be computed, since it is already available due to the eigendecomposition of X which is required for each function evaluation [52]. Step 2 can be done in $\mathcal{O}(n^2)$ and step 3 in $\mathcal{O}(n^4)$. Hence, applying the heuristic is overall relatively inexpensive and we can repeat steps 2 and 3 for many different ($\mathcal{O}(n^2)$) random unit vectors.

Algorithm 2: Goemans-Williamson hyperplane rounding

Input: A positive semidefinite matrix $X \in \mathbb{R}^{p \times p}$
Output: A vector $x \in \{-1, 1\}^p$

Compute $W = (w_1, \ldots, w_p) \in \mathbb{R}^{p \times p}$ such that $X = WW^\top$ (Cholesky factorization)
Generate a random vector $h \in \mathbb{R}^p$ on the unit sphere
for $k = 1$ **to** p **do**

$$x_i \leftarrow \begin{cases} +1, & \text{if } h^\top w_i \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

end
return x

5.2.2 Repair strategies

The hyperplane rounding algorithm obviously exploits the semidefiniteness of X and offers a fairly high diversity of candidate solutions vectors. However, it still suffers from the fact that it may compute ± 1-vectors that violate the 3-cycle-equations (3.5), i.e., it yields no feasible orderings in general. To make it work better in practice, we need a kind of repair strategy which converts any given ± 1-vector into a feasible one. This is done in [41, 43, 44] by an iterative procedure. Therein, the sign of exactly one of the three ordering variables in all violated 3-cycle-equations is flipped until the ordering variables satisfy all equations. Although the success of this approach might by questionable in theory, it is reported in [41, 43, 44] to work very well in practice and outperforms the deterministic heuristic based on the values (5.3).

However, we use some other repair strategies which a more reliable in our opinion and also yield excellent results. Our first strategy, shown in Algorithm 3, is designed to change as few signs of the ordering variables as possible and always outputs a feasible ordering. It is closely related to the idea in [6, 9] based on the values (5.3). The motivation is the same: if the Goemans-Williamson heuristic yields $x_{ij} = 1$ for a particular ordering variable, it is more likely that j should be placed on the right of i with respect to all other ordering variables (or on the left due to symmetry).

This repair strategy has only a computational effort of $\mathcal{O}(n^2)$ and presumably changes a minimum number of signs of ordering variables. Because of its low costs compared to the application of Algorithm 2, we suggest to implement the construction step of the permutation as random as possible, i.e., breaking ties of the R-values randomly. This allows to run the repair strategy several times with the same input yielding different outputs.

Although Algorithm 3 works perfectly fine in practice, we also present another strategy for some reasons. First, the computational cost of applying the Goemans-Williamson hyperplane rounding once is still relatively high compared to the repair strategy. We want to reuse the same (infeasible) ± 1-vector more often yielding more feasible orderings. Second, repair strategy 1 has a blemish when we claim some ordering variables to have specific values. If the ± 1-vector was constructed with some predefined fixations of variables, Algorithm 3 may ignore and destroy these fixations. Admittedly, this should happen less frequently as the

Algorithm 3: Repair strategy 1

Input: A vector $x \in \{-1, +1\}^{\binom{n}{2}}$ with entries x_{ij}, $1 \leq i < j \leq n$
Output: A permutation $\pi \in \Pi_n$ (a feasible ordering)
Initialization: $R[k] \leftarrow 0$, $k = 1, \ldots, n$

foreach *entry* x_{ij} *of* x **do**
  if $x_{ij} = +1$ **then**
    $R[i] \leftarrow R[i] + 1$
  else
    $R[j] \leftarrow R[j] + 1$

Construct $\pi = (\pi_1, \ldots, \pi_n) \in \Pi_n$ such that $R[\pi_1] \geq R[\pi_2] \geq \ldots \geq R[\pi_n]$
return π

matrix X approaches to the optimal solution. Nonetheless, if we work with some weaker semidefinite bounds in a branch-and-bound framework, this may be a design issue. As we propose a particular branching rule in Section 5.4 that requires feasible solutions within the current subproblem, we present an alternative repair strategy that preserves all given fixations of variables and only produces solutions feasible for the respective subproblem.

Now suppose that we are given a set of fixed ordering variables and their corresponding values. We assume that these fixations are consistent, i.e., we can still find a feasible vector of ordering variables by choosing the other values appropriately. Also, the set should be maximal in the sense that each non-fixed variable can still attain both possible values ± 1. We can then construct a vector of ordering variables which respects all fixations by iteratively fixing all other variables. At each iteration, we fix any single non-fixed ordering variable to its desired value. We then have to ensure that the resulting set of fixations again satisfies our assumptions, i.e., it is consistent and maximal. Doing so boils down to examining the 3-cycle-equations (3.5) for implicit fixations, since the were introduced to model transitivity in Section 3.1. This can lead to a snowball effect where many variables are fixed at once. We precede until all variables are fixed.

Our implementation of this repair strategy is given in Algorithm 4. Since a different order of the iterative fixations of all originally non-fixed variables can result in different feasible ordering vectors, and hence more diversity, a permutation of the ordering variables is also required as part of the input. Most of the time is spent in the routine which updates the set of (implicit) fixations, guaranteeing the correctness of Algorithm 4. This can be done by checking all 3-cycle-equations which involve just (implicitly) fixed ordering variables. As each ordering variable is only present in $\mathcal{O}(n)$ 3-cycle-equations, we can bound the overall computational effort of Algorithm 4 by $\mathcal{O}(n^3)$. Thus, we can afford to run repair strategy 2 with $\mathcal{O}(n)$ different permutations of the ordering variables for each ± 1-vector produced by the Goemans-Williamson hyperplane rounding. For this purpose, we mainly use random permutations, but also some heuristic choices, e.g., by sorting the fractional ordering variables in increasing or decreasing order. An extension to the repair strategy would be to compute the permutation only during its execution. Reasonable heuristic choices for the next ordering variable to be fixed include: choosing the one that would lead to the most implicit fixations,

Algorithm 4: Repair strategy 2

Input: A vector $x \in \{-1,+1\}^{\binom{n}{2}}$ with entries x_k, $1 \leq k \leq \binom{n}{2}$,
a permutation $(\pi_1, \ldots, \pi_{\binom{n}{2}}) \in \Pi_{\binom{n}{2}}$,
a consistent maximum index set $\mathcal{F} \subseteq \{1, \ldots, \binom{n}{2}\}$ of fixed variables with corresponding fixed values $\mathcal{V}^{\mathcal{F}} \in \{-1,+1\}^{\mathcal{F}}$

Output: A vector $\tilde{x} \in \{-1,+1\}^{\binom{n}{2}}$ of ordering variables representing a feasible ordering

```
for k = 1 to (n choose 2) do
    if π_k ∉ F then
        F ← F ∪ {π_k}
        V^{π_k} ← x_{π_k}
        Update F and V^F by checking all 3-cycle-equations for implicit fixations
for k = 1 to (n choose 2) do
    x̃_k ← V^k
return x̃
```

choosing the one that violates the fewest 3-cycle-equations if all remaining ordering variables would be fixed a once, and similar ideas. However, we stick to Algorithm 4 due to its simplicity and efficiency.

We also implemented a simple 2-opt local search which tries to improve a valid ordering by swapping the position of two facilities. Since the objective value of an ordering can be computed in $\mathcal{O}(n^2)$ and there are $\mathcal{O}(n^2)$ possibly swaps, we should not apply the local search improvement heuristic too often. We noticed that it works exceptional well and often finds the optimal solution for small and medium-sized instances very quickly, even if we are not anywhere near to the optimal solution of the semidefinite relaxation. Also on larger instances, the combination of our proposed SDP-based heuristics and the local search is quite effective. However, for hard instances it depends only on the SDP-based heuristics whether we find the optimal solution. In [41] another idea for improving the heuristics is proposed and was already used for the max-cut problem in [72, 73]. If $\hat{x}$ denotes the best vector of ordering variables found so far, a new semidefinite matrix $\hat{X}$ is computed as a convex combination of X and $\hat{x}\hat{x}^\top$. All heuristics are then also applied to $\hat{X}$. This is motivated by the fact that $\hat{X}$ might be nearer to the optimal solution than X, because it is shifted towards an already found good heuristic solution. However, we did not implement this idea, since it would require a Cholesky factorization of $\hat{X}$ and our heuristics already worked exceedingly well.

Let us now summarize how we exactly use our proposed heuristics. We apply them to each iterate X_k in the bounding procedure of Section 4.2. As already mentioned above, the Cholesky factorization of X_k is then already available. Depending on the number of facilities n, we apply the Goemans-Williamson hyperplane rounding n times, yielding n potentially infeasible vectors of ordering variables. For each such vector, we run repair strategy 1 (Algorithm 3) a few times and repair strategy 2 (Algorithm 4) with some different permutations. We then try to improve the best ordering found this way with the 2-opt local search algorithm.

Eventually, the resulting ordering is used to update the best known upper bound. If the absolute gap between the best known lower and upper bound is smaller than one, the bounding procedure can be stopped, since optimality was proven. We remark that we typically spend much less than 1% of the overall running time on our heuristics.

5.3 Outline of the implementation

Algorithm 1 in Section 4.2 only sketches the bounding procedure and leaves some important questions unanswered. For this reason, we now give an overview of our implementation for SRFLP in more detail which was used to produce the computational results in Chapter 6. Our program is completely written in the C programming language and calls routines from some external codes (see Section 4.2): L-BFGS-B [63,86] and DSYEVR of the Intel Math Kernel Library (MKL). We used the implementation in the free software *BiqCrunch* [52] as a template. However, we have rewritten it from scratch, only designed for SRFLP. Our implementation involves many minor tweaks, but also some crucial improvements which are needed to fulfill our requirements. This mainly concerns large-scale problems for which our code is much more efficient and robust. Although BiqCrunch [52] is a branch-and-bound solver, we are here only interested in computing lower bounds for SRFLP by solving a single semidefinite relaxation. A branch-and-bound approach is the topic in Section 5.4. Our code typically uses some adaptive settings which are chosen with respect to the given instance size, i.e., the number of facilities.

Two different versions. The bounds presented in Section 4.2 are known to be very flexible in the sense that they allow to compute cheap bounds very quickly, tight bounds with higher effort, but also some kind of balanced bounds where accuracy is traded for running time (see [52, 62]). In view of Algorithm 1, this can mainly be done by using different reduction schemes for the tightness parameter α_k and the tolerance ε_k. Also the underlying semidefinite relaxation and the corresponding update process for the inequalities can have a major impact on the bounds and running times. For example, we noticed that the heuristic separation of pentagonal, hexagonal and heptagonal inequalities often improves the lower bound, but always implies much higher running times. For many instances its usage is indeed unnecessary. Whereas tighter bounds can be achieved by applying this heuristic separation, a more economical or balanced approach would be to omit it. In the latter case, the corresponding bounds seem to be more attractive in a branch-and-bound approach. For this reason, and to show the efficiency of (SDP_4), we tested two different versions of our program. By potentially allowing small deviations of Algorithm 1, we present the settings of a more balanced version, named (BV), and a tighter version (TV). The main difference is that (BV) is directly based on (SDP_4), i.e., it only uses triangle and starlike pentagonal inequalities, and is designed to perform less function evaluations than (TV). As a result, (BV) requires much less time, but also fails on solving some large instances.

DSYEVR and L-BFGS-B settings. During each iteration of Algorithm 1, the L-BFGS-B solver requests many function and gradient evaluations. In view of the correctness of the computed

lower bounds, the gradient evaluations are a noncritical part, since inaccuracies in these can only lead to unreasonable trial points suggested by the L-BFGS-B solver. In contrast, the function evaluations should always be very accurate in order to avoid wrong lower bounds. Using the default settings of BiqCrunch, we sometimes observed for very small values of the tightness parameter α_k that L-BFGS-B aborts due to inconsistent function and/or gradient evaluations. If this happens, we would have to stop the bounding procedure prematurely, since it is likely to happen again and again. To overcome this issue, we demand the highest possible accuracy in the eigenvalue computation of DSYEVR. This only leads to a performance loss of about 10% compared to BiqCrunch in terms of the number of function evaluations in the same period of time. However, we noticed that this also benefits cases in which no issue would arise by a significantly lower number of required function evaluations to achieve the same given accuracy with respect to ε_k. Overall, it even speeds up the bounding procedure. Moreover, the authors of L-BFGS-B [63,86] suggest to use a number m of memory corrections between 3 and 20, since higher values would only lead to increased running times. However, our tests show that higher values of m still have a positive impact on the optimization process for SRFLP. For (TV) we use a constant value of $m = 42$ which leads to a reduced number of function evaluations for large instances, where α_k often takes very small values. For (BV) we choose m adaptively with respect to the number of facilities: $m = \left\lfloor \frac{n}{2} \right\rfloor$. As in BiqCrunch, we leave all other L-BFGS-B default settings unchanged except for its built-in termination rules which we disable.

Bounding procedure parameters. Our settings for the sequence of tightness parameters α_k and tolerance parameters ε_k are similar to those of BiqCrunch, since they also work well for SRFLP. However, our bounding procedure has some distinctions. We start with $\alpha_1 = 0.1$, $\varepsilon_1 = 0.1$ in (BV) and $\alpha_1 = 1$, $\varepsilon_1 = 0.1$ in (TV). Starting with these values, we impose a fixed schedule to reduce them. After exactly $\lceil \sqrt{n} \rceil$ iterations of Algorithm 1 with the same parameters α and ε, α is scaled-down by factor 0.5 and ε by factor 0.95 in both versions. This is continued until the instance is solved to optimality, α_k reaches a symbolic value of 10^{-10}, or a predefined time limit is exceeded. We remark that for all instances which were solved to optimality, the smallest required value of α was about 10^{-6}. Typically, much higher values of roughly 10^{-4} are sufficient. Due to its higher starting value, (TV) needs more iterations of the bounding procedure than (BV) to reach the same tightness level α. Also, for a fixed value of α, the tolerance parameter ε always takes smaller values in (TV) leading to more accurate solutions, but also to more function evaluations. To avoid unreasonable numbers of function evaluations in extreme cases, we limit the number of function evaluations per iteration. We allow a maximum number of $2\binom{n}{2}\sqrt{n}$ per iteration in (TV). In (BV) we allow a maximum number of $\frac{1}{p}\binom{n}{2}\sqrt{n}$ evaluations, where p is the number of already consecutive iterations with the same parameters α and ε. In addition, we also claim a minimum number of evaluations per iteration for (TV) which we set to $2n$. By doing so, we make possible that newly added inequalities are better incorporated. This leads to more accurate primal variables X_k and dual variables λ_k for medium values of α. As a result, the relaxation involves fewer but also more suitable inequalities, since the update routine is more effective.

Inequality update routine. As shown in Algorithm 1, the initial relaxation involves no inequalities. After the new iterate X_k is computed and the primal heuristics have been applied to it, the set of inequalities is updated. Our tests showed that the L-BFGS-B solver can effectively handle a huge number of inequalities simultaneously, especially for large instances where the function evaluation is by far the bottleneck. Therefore, we allow up to two million inequalities within the relaxation at any time. However, this limit is never reached in our experiments. The maximum number of inequalities was barely above one million for a very few instances with $n = 100$. The typical number for large instances is between three and six hundred thousand. The inequality update routine works as follows. First, in both versions we remove all inequalities that have a Lagrange multiplier less than 10^{-8}. The primal variable X_k is then used to find and separate violated inequalities. The types of inequalities which are separated (by enumeration or heuristically) again depend on the number p of already consecutive iterations with same parameters α and ε. Whereas triangle inequalities are always separated, starlike pentagonal inequalities are only separated if $p \geq 4$. In (TV), the heuristic procedures proposed in Section 5.1 are used to separate pentagonal inequalities if $p \geq 6$, heptagonal inequalities if $p \geq 7$ and hexagonal inequalities if $p \geq 8$. Hence, not all of theses types are used for all instances, e.g., the maximum value of p for $n \leq 36$ is 6, i.e., heptagonal and hexagonal inequalities are never separated. However, we have implemented a buffer in (TV) which saves a large number of heuristically separated inequalities of previous iterations. All inequalities therein are always separated again, independent of p. We use these different stages of separation to attach a much higher value to triangle and starlike pentagonal inequalities which are the most important ones. Additionally, we scale down the absolute value of violation for other than triangle inequalities. We use a scale factor of 0.7 for pentagonal inequalities, 0.5 for heptagonal inequalities and 0.4 for hexagonal inequalities. Hence, we treat a triangle inequality with an absolute violation of the right hand side by 0.1 as equal as a heptagonal inequality with an violation by 0.2. To be qualified to be added to the relaxation, the scaled violation of any inequality must be at least 10^{-2} in (BV) and 10^{-3} in (TV). Using the scaled violations, the most-violated $5n^2$ inequalities are then added to the relaxation in (BV) and up to $10n^2$ in (TV). Their corresponding dual variables are initialized to zero.

Miscellaneous. The natural symmetry of SRFLP, which also appears in the semidefinite relaxation (SDP_4), can be broken by fixing any single ordering variable to any value. We observed that the particular choice can have a huge impact on the running time. The required function evaluations may vary between a factor of about six. However, we always break symmetry by locating facility 1 on the left of facility 2, i.e., by enforcing $y_{12} = 1$. We also implemented a kind of safeguard against failure of the L-BFGS-B solver. Although we use DSYEVR settings with maximum accuracy, this still might happen rarely. It may occur when α takes too small values, or ironically, when the current dual variables are very close to their optimal values, leading to a vanishing gradient. In case of failure, we check whether the current stored dual variables are still feasible, i.e., all Lagrange multipliers of inequalities are greater than or equal to zero. Due to the implementation of L-BFGS-B, this should always be the case. If so, we immediately reduce α and ε and continue the bounding procedure with the next iteration. As shown in Table 4.1, the function evaluations can greatly take advantage

of multi-core processors. Therefore, we also sped up the primal heuristics and separation routines by parallelizing them with OpenMP. Since we have to manage significantly more inequalities in the update routine than BiqCrunch, we make use of some more advanced data structures such as hash tables and red-black trees.

5.4 Towards a branch-and-bound approach

Up to date, there is no proposed exact approach for SRFLP in the literature that is based on semidefinite programming. Only a very few computational results on the basic relaxation (SDP_1) within a branch-and-bound framework were given in [9]. However, not even medium-sized instances could be solved within reasonable time. All other approaches, including the current leading approach by Hungerländer & Rendl [44], only provide lower bounds for the optimal solution and heuristics for obtaining good feasible orderings. For the most part, we also focus on this line. First, we propose the new, potentially tighter relaxation (SDP_4), alongside other inequalities which are separated heuristically. Second, we use a well-suited algorithmic approach to approximate the associated lower bounds. Third, we apply reasonable designed heuristics to find high quality layouts. Combining these three things leads to astonishingly strong lower bounds and overall very small optimality gaps in reasonable time, if at all (see Chapter 6). As a result, our approach clearly outperforms all other approaches in the literature. However, there are still some very large instances for which our computed lower bounds are not sufficient to prove global optimality. Using our semidefinite bounds in a branch-and-bound framework is a natural idea to tackle the remaining instances.

Due to time restrictions and the high potential of solving instances without branching, we have not tried to solve very large instances this way yet. For a few first initial tests, we developed a prototype of a branch-and-bound solver for SRFLP. It is even a parallelized version of our bounding procedure, i.e., more than one node of the branch-and-bound tree can be evaluated at the same time (similar to [19]). It still lacks many important features that are required for an efficient solution process. Hence, we could not produce reasonable computational results yet. However, we already got some insights into its potential and provide our thoughts and experience in the following. We consider some important questions like: Which settings of the bounding procedure should be used? Which branching strategy is the best? When should we stop the bounding process and perform a branching step? How should such a branching step actually be performed? Which are types of instances for which a branch-and-bound approach may be more efficient, even if the instance could be solved at the root node? We remark that we tested the branch-and-bound approach only on small and medium-sized instances by using less accurate settings than those presented in Section 5.3. Not surprisingly, the running times are almost always much higher, since it typically requires many nodes to compensate the weaker lower bounds. One exception is when we have access to hardware which allows us to solve many nodes in the branch-and-bound tree concurrently, see below.

5.4.1 General ingredients

As fixations of variables occur in any branch-and-bound approach, we have to incorporate them into our semidefinite relaxations. In theory, there are actually two ways to do this. Note that such a fixation is already obtained by breaking the symmetry of SRFLP. We also always want to assure that the set of fixations is consistent and maximal as it was already claimed in Algorithm 4 by enforcing implicit fixations due to transitivity. The first idea to manage the fixations is to add explicit equations to the relaxation, i.e., constraints of the type $y_{ij} = b \in \{-1, 1\}$. If doing so, we also might want to add the so-called product constraints $y_{ij}y_{kl} = by_{kl}$, $k, l \in [n]$, $k < l$ to achieve better results (see [52]). Hence, the number of constraints, i.e., the number of dual variables, increases as more variables are fixed. An interesting advantage of this type of fixation is that it could allow a much better warm start of the dual variables after a branching step. Maybe some nodes could be pruned with minimal computational effort. However, this approach is not as robust as its alternative and the improvement of the lower bound additionally depends on the approximation of the corresponding Lagrange multipliers. Therefore, we did not consider this approach at all. The other more natural possibility is to substitute the fixed variables in all constraints and the objective function by their fixed values. This allows an exact consideration of all fixations and moreover reduces the problem size, leading to potentially lower running times. If all variables are fixed, we have an explicit expression of the lower bound, whereas we would have a gigantic optimization problem with the first approach. We will later argue that suitable fixations lead to significantly lower running times not only due to decreased problem sizes, but also due to a better behavior of the iterative solution process requiring much less function evaluations.

It turns out to be a challenging task to embed our bounding procedure into a branch-and-bound algorithm in an efficient way. From all settings, the control over the number of function evaluations is the most critical part. Hence, the tolerance parameters ε_k and possibly an upper limit on the function evaluations have to be chosen appropriately. Their impact is crucial, since our iterative bounding procedure only yields approximate solutions, and only slightly weaker bounds can lead to many more nodes in the branch-and-bound tree. On the other hand, the number of required function evaluations often grows quickly with decreasing α and ε. It is important to recognize when the computational effort gets too high and it is very unlikely that the current node can be pruned. We will later give a useful rule of thumb when a branching step is advisable. Overall, the main question is: which settings provide the best ratio of accuracy to running time? Unfortunately, there seems to be no simple answer to this question, since all instances behave differently. Nonetheless, we observed that the accuracy should have a much higher precedence over running time. Before we start to discuss how the effort for each node can be kept as small as possible, we want to emphasize one very important fact which is relevant for us: the tolerance parameter ε plays a much bigger or at least different role than one may think. As the optimization process goes on, smaller values of ε for sure yield better lower bounds. However, for small values of α it often happens that the lower bound only shows very slow and tiny progress, but a great many of iterations is still required to achieve the given tolerance ε. This is because of the iterates X_k are very sensitive to small changes of the dual variables. Choosing a higher tolerance ε or stopping the optimization process of dual variables prematurely does not weaken the current

lower bound that much, but can lead to a poor approximation X_k of the primal variable X. This results in an effectively superfluous separation process for the inequalities and the lower bounds indeed become weaker, but only in future iterations.

Our implementation in Section 5.3 illustrates that most of the time is spent on finding a good subset of inequalities, since we always run $\lceil \sqrt{n} \rceil$ iterations with the same values of α and ε. This is also done for relatively high values of α for which it is clear that the lower bound has poor quality. Hence, to design an efficient branch-and-bound approach, it is indispensable to endow the child nodes with all gathered information in the father node and prior ancestors. More precisely, when the branching step is performed, all active inequalities in the father node should be transferred to its children in such a way that these inequalities are not removed in update routines. The children themselves hand down only the inequalities to their children which are active at the end of their bounding procedure. As we already know, more and more function evaluations are required as α decreases. In addition to this, the consideration of newly added inequalities becomes also more expensive, since their corresponding Lagrange multipliers are always initialized to zero. Thus, the effort for incorporating new inequalities appropriately increases rapidly. We believe that adding new inequalities to the relaxation is not worthwhile within a branch-and-bound approach for small values of α, say $\alpha \leq 10^{-4}$, and should be omitted then. Once this threshold is reached, we suggest to reduce α more aggressively in order to get a good lower bound more quickly. However, inequalities should still be separated to supply them later to the child nodes. With increasing number of nodes, the knowledge of the instance grows. We guess that the tolerance parameter ε can then take higher values without any complications. In general, a much more flexible bounding procedure than proposed in Section 5.3 is necessary, especially the initial starting values of α and ε should be chosen adaptively. A fixed number of iterations also seems to be an odd choice in this setting.

So far we have only discussed how to redesign the bounding procedure to obtain efficient lower bounds. We now turn to the other aspects of any branch-and-bound approach. The primal heuristics presented in Section 5.2 also work very well on more rough approximations X_k of the primal variable X due to their highly randomized nature. To play it safe, they probably should be applied more often than we do in our bounding procedure in Section 5.3, especially when the bounding process of any node is finished. Also we highly recommend to use the 2-opt local search heuristic, since it always yields near optimal or even optimal solutions. As finding high quality layouts is relatively easy, we should follow a least lower bound first strategy, i.e., always considering the node with the least lower bound. There is one simple criterion which allows us to detect whether we still have the chance to prune the current node. For any instance solved to global optimality with the bounding procedure of Section 5.3, we observed that the optimality gap at least halves itself when α is reduced by factor 0.5 (assuming that we know the optimal solution). Hence, it is a good idea to finish the bounding process and perform a branching step if we can not notice such a behavior. This makes clear why primal heuristics, especially the 2-opt local search, are very important.

5.4.2 Branching strategies

We now consider different possible branching strategies for SRFLP. A natural approach is to choose a suitable single non-fixed ordering variable x_{ij} and to create two child nodes in which x_{ij} is fixed to -1 or $+1$ respectively. To do this, we extract the first column of the last iterate X_k yielding the fractional vector x of ordering variables. Commonly used branching strategies include the least-fractional and most-fractional rules (see e.g. [50,52,72,73]). Least-fractional means that we branch on the variable x_{ij} which is furthest from 0 and most-fractional that we branch on the variable x_{ij} which is closest to 0. Both strategies are reasonable and have their own strengths. Choosing the variable which is the furthest from 0 has the advantage that one of the two created subproblems is more likely to be pruned quickly, since the lower bound can have a huge surge. However, the lower bound in the other subtree can have only a little or no improvement at all. The most-fractional rule aims to resolve the most difficult decisions first, probably leading to noticeable improvements of the lower bounds in both subproblems. Therefore, the least-fractional strategy typically leads to a narrower, but also deeper branch-and-bound tree than the most-fractional strategy. Our experiments indicate that the most-fractional rule is superior for SRFLP, since it produces significantly fewer nodes and almost always has lower running times. This is mainly due to implicit fixations of ordering variables which occur much more frequently when the most-fractional strategy is used. This results in a stronger decline of the problem sizes than possible with the least-fractional strategy.

We want to introduce two more branching strategies which have proven to be very useful and take advantage of the structure of SRFLP. The first strategy addresses an issue which may arise when the simple most-fractional rule is used. The latter is very sensible with respect to the approximation X_k of X. Especially when we terminate the bounding procedure quite early or when we cannot achieve the desired accuracy in reasonable time for small values of α, the most-fractional rule might produce unwanted and unhelpfully subproblems. This can lead to many nodes in the branch-and-bound tree, comparable to the least-fractional strategy. In particular, the path which contains the optimal solution can be of high length. Our suggested strategy tries to keep this length as small as possible and leads to a high number of implicit fixations. We propose the following branching rule.

(R1): Among all ordering variables x_{ij} for which i is adjacent to j in the best known ordering for the current subproblem, branch on the most-fractional one.

Applying branching rule (R1) requires feasible orderings for each subproblem. These can be easily found by Algorithm 4. Depending on the best known solution for the current subproblem and already fixed variables therein, there are at most $n-1$ candidate variables to branch on. Moreover, if these variables are fixed to values equal to those in the best known solution (or any other solution), all ordering variables are fixed due to implicit fixations. Hence, assuming that we already find the optimal solution at the root node, the path in the branch-and-bound tree which contains the optimal solution has a maximum length of $n-1$, since the best known feasible solution for all nodes on this path will always be the same. Branching rule (R1) less depends on the quality of the approximate solution X_k as it always yields a reasonable branching decision. However, finding a good and hopefully the

best solution within a subproblem becomes more important than ever before. We observed that the most-fractional branching rule and (R1) often produce the same branching decision, especially for good approximations X_k. In general, they seem to yield similar number of nodes for small instances of SRFLP, but the most-fractional rule is outperformed by (R1) as the instance size grows or weaker bounds are used.

The other branching strategy does not follow the standard template and may produce significantly more than two subproblems, but has a remarkable impact on the performance of the bounding procedure.

(R2): Choose either left or right. Determine the outermost position on the chosen side on which under consideration of already fixed variables more than one facility can still be placed. Create a subproblem for each facility which can be placed at this specific position and fix the ordering variables appropriately.

Obviously, branching rule (R2) does not depend on any approximation X_k or best known solution. It also provides few degrees of freedom, since we only have to choose between left or right. The number of produced subproblems depends on already fixed variables. Taking the symmetry breaking into account, (R2) produces at most $n-1$ subproblems, and of course at least two. Although the number of created nodes is relatively high, more variables are fixed simultaneously than with other branching strategies: the number of newly fixed variables in all created subproblems is at least as high the number of created subproblems. Therefore, the improvements of the lower bounds in many subtrees may be very high so that they can be pruned very quickly. This is indeed the case for many instances, but leads to huge computational efforts for some instances where this is not the case. However, we actually do not want to use (R2) as a branching strategy on its own, since its true value lies somewhere else. We suggest to apply this strategy only a very few times, maybe only at the root node, and to use other branching strategies like (R1) thereafter. Without any exception we observed that (R2) significantly reduces the number of function evaluations, at least by factor 2, but often considerably higher, leading to a behavior similar to the choice of symmetry breaking. This positive effect seems to be much stronger for larger instances, in particular when small values of α are involved. Provided suitable hardware for solving many subproblems concurrently, branching strategy (R2) may be very valuable to solve the remaining unsolved large instances. In fact, we were able to solve some small instances faster than with the bounding procedure of Section 5.3. However, the CPU times were much higher.

Finally, let us remark that there are types of instances which are much more suited for a branch-and-bound approach combined with our bounding procedure than others. We made the experience that instances with many solutions having objective values near to the optimal solution are much harder so solve. For such instances a bounding procedure with high accuracy is clearly preferable. On the other hand a branch-and-bound approach may be more efficient if all feasible solutions are well separated from each other. This is especially the case for randomly generated instances with a high range of coefficients in its data. Whereas it can be expensive to achieve the necessary absolute accuracy at the root node, branching rule (R1) plays out its strengths and allows to use considerably weaker bounds. For these instances, the optimal solutions can easily be found heuristically and subproblems not containing the optimal solution are pruned quickly.

Chapter 6

Computational Results

In this chapter we provide extensive computational results on our proposed approach for SRFLP. We test our two different versions (BV) and (TV) on many small, medium and large instances from the literature and compare them to other successful approaches, especially the current leading approach by Hungerländer & Rendl [44]. Our set of test instances includes almost all publicly available benchmark instances.[1][2] This set covers many different types: randomly generated instances, instances with unit facility lengths, instances derived from other problems such as the quadratic assignment problem and more. On all of these, (BV) significantly reduces the required computing times for all instances successfully solved by the current leading approach by Hungerländer & Rendl [44]. It always computes their obtained lower bounds in significantly less time and also yields much tighter bounds, especially for large instances. The inherent accuracy allows to solve remarkably larger instances to global optimality for the very first time. At the cost of higher running times, the version (TV) solves even more instances leaving only a few remaining unsolved. Additionally, our primal heuristics turn out to be much better. Taken together, our obtained lower and upper bounds almost always reduce the optimality gap compared to the approach by Hungerländer & Rendl [44] at least by a factor of about 10, usually significantly more. For all unsolved instances, we compare our best upper bounds to the best known solutions found by heuristic approaches in the literature.[3][4] We also demonstrate that the simple 2-opt local search heuristic is worth its application, since it can find the optimal solution of even fairly large instances quickly. To emphasize this, we let it run on randomly generated permutations for exactly one second before the actual optimization process in (TV) starts. Hence, the running time of (TV) is at least one second for all instances.

All results in [41, 43, 44] regarding the bundle approach by Hungerländer & Rendl [44] (see Section 4.1) were carried out on a machine with an Intel Xeon 5160 processor with 3 GHz and 2 GB RAM. We obtained our results on an Intel Xeon E5-2640V4 processor with 2.4 GHz and 1 GB RAM. For very large instances with $n = 100$ we used all of its cores, but limited the number of cores used to only four for all other instances.

[1] https://www.miguelanjos.com/flplib

[2] https://www.philipphungerlaender.com/benchmark-libraries/layout-lib/row-layout-instances/

[3] https://www.miguelanjos.com/ImprovedResults.txt

[4] https://www.philipphungerlaender.com/benchmark-libraries/layout-lib/single-row-benchmarks/

Typical behavior of our solution approach. Before we start with the actual results, let us first illustrate the typical behavior of (BV) and (TV) on instances that were solved to optimality. More precisely, we look at how the optimality gap vanishes and the number of performed function evaluations increases as the tightness parameter α decreases. In Figure 6.1, we show the typical qualitative characteristics, vicariously for all instances, for the particular instance 'Y-60' solved by (TV). Assume now that we already know the optimal objective value of the given instance. Dependent on α, Figure 6.1 then depicts the courses of the optimality gap, which is the relative difference of the current lower bound and the optimal value, and the accumulated number of function evaluations. Note that all axes have a logarithmic scale. Since the plot of the gap is nearly a straight line, the gap is always reduced by nearly the same constant factor as α is halved. If we take a closer look, we can see that this constant factor is greater than 2. This can be observed for any instance that was successfully solved by (BV) or (TV). In almost all cases, the factor was between 2 and 3. Hence, the bounding procedure quickly produces relatively small optimality gaps. But it can take much more time to prove optimality, especially for badly scaled instances for which a high relative accuracy is necessary. Figure 6.1 also shows that the number of required function evaluations strongly increases as α attains smaller values, but flattens a bit when the current solution is very close to the optimal solution. For instances which could not be solved, the number of function evaluations typically continued to increase even further as α was halved.

Comparison on 'classical' instances. We first consider some well-known 'classical' SRFLP benchmark instances which were solved by many solution approaches in the literature before. These include linear and semidefinite programming approaches presented in this thesis other than the approach by Hungerländer & Rendl [44]: the betweenness approach of Section 2.3 in [4], the distance approach of Section 2.2 in [5] and the semidefinite approach in [10] using (SDP_2) as discussed in Section 3.3. The results in [4] were obtained on an Intel Core Duo processor with 1.73 GHz and 1 GB RAM, in [5] a Pentium Dual Core PC with 2.5 GHz and 2 GB RAM was used and a machine with a 2.0 GHz Dual Opteron processor with 16 GB RAM was employed in [10].

In Table 6.1 we compare the above approaches, as well as the approach by Hungerländer & Rendl [44] and our versions (BV) and (TV). The first two columns identify 14 instances by their name and where they are taken from. The seven 'S' instances were introduced in the initiating paper [78] for SRFLP and are the oldest of all instances. The two 'H' instances originate from [37] and are constructed from well-known instances of the Quadratic Assignment Problem [64]. The remaining five instances were generated randomly in [4] or are modifications of some already existing SRFLP instances [1, 3]. The next two columns of Table 6.1 show the size of each instance and its optimal value. All subsequent columns list the required running times by all approaches to find an optimal ordering and prove its optimality.

First, we notice that all approaches can solve very small instances with $n \leq 11$ quickly. However, the running times of the distance approach [5] and the semidefinite approach in [10], using an interior-point method, rise dramatically for $n \geq 20$. Taking the slower machine into account, the betweenness approach [4] is highly competitive for all instances with up to 20 facilities. It has significantly lower running times compared to the approach by

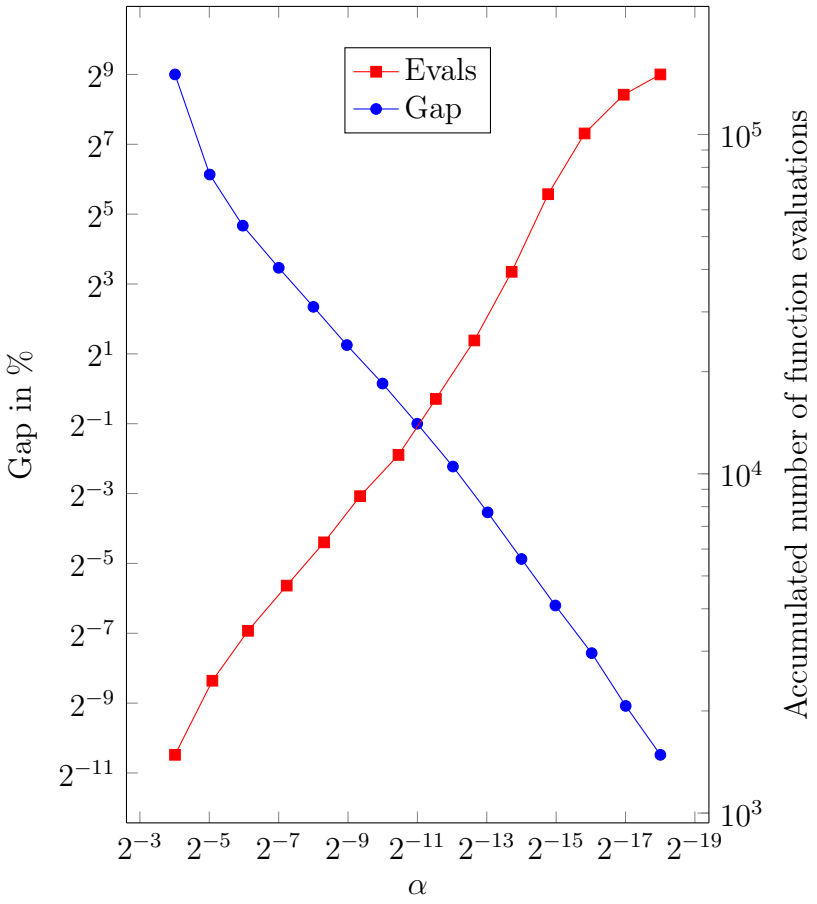

Figure 6.1: Typical courses of the optimality gap and the accumulated number of function evaluations. The shown data correspond to the instance 'Y-60' which was solved by (TV).

Instance	Source	n	Optimum	LP-based approaches [4]	LP-based approaches [5]	SDP-based approaches (SDP_2) [10]	SDP-based approaches (SDP_3) [44]	(BV)	(TV)
S5	[78]	5	151	0.1	0.1	–	0.1	0.0	1.0
S8	[78]	8	801	0.1	0.5	–	0.6	0.0	1.2
S8H	[78]	8	2324.5	0.1	0.1	0.2	2.3	0.0	1.1
S9	[78]	9	2469.5	0.1	0.1	–	0.7	0.0	1.3
S9H	[78]	9	4695.5	0.1	2.4	–	9.2	0.1	1.5
S10	[78]	10	2781.5	0.2	0.4	3.4	0.6	0.1	1.5
S11	[78]	11	6933.5	0.3	0.7	32.6	1.3	0.2	1.8
P15	[1]	15	6305	2.8	–	–	19.7	1.6	12.5
P17	[3]	17	9254	8.4	–	–	34.9	2.7	23.1
P18	[3]	18	10650.5	13.3	–	–	32.5	2.9	22.5
H20	[38]	20	15549	30.8	2:22	26:54	54.3	4.9	36.1
H30	[38]	30	44965	27:35	28:07:49	15:50:57	9:07	48.5	6:34
Am33_03	[4]	33	69942.5	2:22:32	–	–	36:33	1:30	9:49
Am35_03	[4]	35	69002.5	2:17:52	–	–	53:14	2:57	18:53

Table 6.1: Comparison of most successful LP-based and SDP-based approaches for some SRFLP benchmark instances with up to 35 facilities. The running times are given in sec, in min:sec or h:min:sec respectively. A '–' sign indicates that the instance was not considered by the respective approach.

Hungerländer & Rendl [44]. Note that for the latter sometimes 'accidents' happen, i.e., the bundle approach shows a strongly non-monotonic behavior in the number of facilities, even for such small problem sizes. Despite our faster machine, the version (BV) seems to be on a par with the betweenness approach for $n \leq 20$. This not true for (TV), since it requires much higher computing times on all such instances. However, it still has a performance similar to the bundle method approach. For $n \geq 30$, the betweenness approach is clearly dominated by the semidefinite approaches using Lagrangian relaxation. Especially our version (BV) outperforms all other approaches by far. We also notice that the ratio of running times between (BV) and (TV) flattens on larger instances, i.e., (TV) catches up and is then also superior to the bundle approach by Hungerländer & Rendl [44].

We remark that the bundle approach by Hungerländer & Rendl [44] is the only approach in the literature which is capable of solving some larger instances to optimality in reasonable time and was tested on many more instances than all other approaches. It also provides much stronger lower bounds for large-scale instances compared to [6] and [12] which only use (SDP_1) or (SDP_0) respectively. Therefore, for the rest of this chapter, we only compare our versions (BV) and (TV) against each other and the approach by Hungerländer & Rendl [44].

More small and medium-sized instances. Next, we consider many instances with unit facility lengths which were derived in [68] from various other problems such as the Quadratic Assignment Problem. All of these instances have up to 35 facilities and were successfully solved by the bundle method approach [44]. Table 6.2 shows the required running times for the bundle approach and our two versions (BV) and (TV). It also contains results for five larger randomly generated instances with $n = 40$ that were introduced by Hungerländer & Rendl [44].

We can see that (BV) is nearly always at least 10 times faster than the bundle approach and sometimes even more than 20 times faster. In general, the computing times of (BV) grow much smoother with the number of facilities than those of the bundle approach. For instances with $n = 40$, (TV) is about 5 times slower than (BV) but still almost always faster than the approach by Hungerländer & Rendl [44]. The running times also do not vary that much on equally sized instances compared to the bundle approach.

Optimal solutions for previously unsolved medium-sized instances. Let us now consider medium-sized instances with up to 60 facilities for which mostly a proven optimal solution was not known before. For these instances, the bundle approach is restricted to 500 function evaluations in [42, 43] in order to have a better control over the computational effort and to react to the slow convergence behavior of the bundle method. However, we impose no such limit for our approach on these instances. But in addition to our normal stopping criteria (see Section 5.3), we stop the bounding procedure when the lower bound improved by less than 0.5 after a reduction of the tightness parameter α and when it is still not sufficiently close to the upper bound.

The set of instances contains yet more larger 'Y' instances [85] with unit facility lengths, as well as parts of the 'ste' and 'sko' instances which were introduced in [12]. These are also derived from Quadratic Assignment Problem instances, but only the instances ending

Instance	Source	n	Optimum	(SDP_3) with bundle method [44]	(BV)	(TV)
O-5	[65]	5	150	1	0.0	1.0
O-6	[65]	6	292	1	0.0	1.0
O-7	[65]	7	472	1	0.0	1.1
O-8	[65]	8	784	1	0.1	1.3
O-9	[65]	9	1032	1	0.1	1.3
O-10	[65]	10	1402	1	0.1	1.7
O-15	[65]	15	5134	4	0.8	7.2
O-20	[65]	20	12924	37	4.9	30.7
S-12_t	[77]	12	4431	4	0.3	3.0
S-13_t	[77]	13	5897	4	0.4	3.8
S-14_t	[77]	14	7316	15	1.0	7.3
S-15_t	[77]	15	8942	19	1.1	7.5
S-16_t	[77]	16	11019	27	1.8	11.8
S-17_t	[77]	17	13172	50	2.2	19.7
S-18_t	[77]	18	15699	48	2.6	21.9
S-19_t	[77]	19	18700	1:43	3.4	27.9
S-20_t	[77]	20	21825	2:09	6.1	42.4
S-21_t	[77]	21	24891	3:20	18.1	1:15
S-22_t	[77]	22	28607	3:57	12.3	1:18
S-23_t	[77]	23	33046	3:15	9.4	1:25
S-24_t	[77]	24	37498	3:31	15.3	1:32
S-25_t	[77]	25	42349	7:07	21.7	2:27
N-15	[64]	15	2186	17	1.8	8.0
N-16a	[64]	16	3050	8	1.5	9.3
N-16b	[64]	16	2400	6	1.0	7.9
N-17	[64]	17	3388	13	1.6	17.8
N-18	[64]	18	3986	16	3.0	21.1
N-20	[64]	20	5642	1:11	7.4	48.2
N-21	[64]	21	5084	1:16	9.3	56.4
N-22	[64]	22	6184	1:20	9.2	1:11
N-24	[64]	24	8270	2:12	14.8	7:21
Y-6	[85]	6	1372	1	0.6	1.0
Y-7	[85]	7	1801	1	0.6	1.1
Y-8	[85]	8	2302	1	0.8	1.2
Y-9	[85]	9	2808	1	0.0	1.2
Y-10	[85]	10	3508	2	0.9	1.7
Y-11	[85]	11	4022	4	1.0	2.2
Y-12	[85]	12	4793	10	1.2	3.4
Y-13	[85]	13	5471	9	1.3	4.3
Y-14	[85]	14	6445	34	1.7	5.7
Y-15	[85]	15	7359	26	1.5	5.9
Y-20	[85]	20	12185	1:14	5.2	55.7
Y-25	[85]	25	20357	5:14	19.3	2:25
Y-30	[85]	30	27673	17:46	1:08	9:35
Y-35	[85]	35	38194	25:50	3:23	15:27
N40_1	[44]	40	107348.5	1:01:36	5:08	30:34
N40_2	[44]	40	97693	52:52	5:16	34:08
N40_3	[44]	40	78589.5	1:21:40	8:24	30:47
N40_4	[44]	40	76669	1:15:58	7:29	37:08
N40_5	[44]	40	103009	2:20:09	7:37	41:36

Table 6.2: Results on many small and medium-sized SRFLP instances with unit lengths (except the last five). The last five instances were randomly generated in [44]. The running times are given in sec, in min:sec or h:min:sec respectively.

on '1' have unit lengths. The other instances have randomly generated lengths. Table 6.3 shows the computational results on these instances for the bundle approach and (BV). For each instance, the column 'Best' lists the best known upper bound in the literature for this particular instance. 'LB' in Table 6.3 stands for the respective computed lower bounds of both approaches and 'UB' for the upper bounds obtained by the deployed heuristics. The corresponding gaps (in %) are given by $(\mathrm{UB} - \mathrm{LB})/\mathrm{LB} \cdot 100\%$. Additionally, the column 'Time best' shows which computing time is required by (BV) to find a solution with objective value at least as good as the best known solution in the literature. Moreover, the last column of Table 6.3 shows after which time (BV) has improved on the lower bound computed by Hungerländer & Rendl [42, 43].

We can see in Table 6.3 that the bundle approach only solves six instances to global optimality, the largest having 42 facilities. In contrast, (BV) successfully solves all but two of the largest instances in less than 90 minutes. Hence, (BV) computes much stronger lower bounds in much less time. This can also be seen in the last column showing that (BV) computes the same lower bounds between 10 to 20 times faster than the bundle approach by Hungerländer & Rendl [44]. We remark that for some instances, the triangle inequalities actually suffice for solving. Therefore, the mostly small, but not optimal gaps in the bundle approach indicate that the accuracy of the bundle method is most likely the limiting factor.

The gaps of (BV) for the two unsolved instances are very slight, but the running times to achieve these gaps are much higher. However, this is only due to the fixed nature of our bounding procedure and our loose stopping criteria. The high running times are caused by very small values of the tightness parameter α for which an excessive number of function evaluations is required, whereas the improvement on the lower bound is very small. Hence, much cheaper, but only slightly weaker bounds for these instances could be obtained by using more suited stopping criteria. However, (BV) still computes the lower bounds of Hungerländer & Rendl [44] in significantly less time on these instances.

(BV) always finds an ordering with objective value equal to best known solution in the literature, proving that an optimal ordering was known before. Note that this is also true for the two unsolved instances, see below. Moreover, these solutions are often found within only one minute. In all except one case, it is also found before the computed lower bound reaches the level of the bundle approach. In contrast, Hungerländer & Rendl [44] often do not find these optimal solutions with their proposed heuristics.

Table 6.4 shows the results of (TV) when applied to the instances of Table 6.3 and compares them to (BV). As we can see, (TV) is able to solve all instances to optimality, but requires running times which are about five times higher than those of (BV). For the smallest instances with $n = 36$, (TV) is sometimes even slower than the approach by Hungerländer & Rendl [44]. For the rest of the instances, (TV) still reaches the lower bound of the bundle approach in much less time and is capable of computing tighter bounds than (BV). Although (TV) solves the two instances which could not be solved by (BV), the needed running times for these are substantially higher than for all other instances. This shows that the bounds produced by (BV) can be improved, but most probably at the cost of a much higher computational effort. It is also striking that (TV) often finds the optimal ordering within one second, i.e., by using only the 2-opt local search heuristic. As the instance size grows, this becomes less likely.

				(SDP_3) with bundle method restricted to 500 function evaluations [42, 43]				(BV)					
Instance	Source	n	Best	LB	UB	Gap	Time	LB	UB	Gap	Time	Time best	Improve LB (SDP_3)
Y-40	[85]	40	47561	47561	47561	0%	2:14:47	47561	47561	0%	8:26	6.6	–
Y-45	[85]	45	62890	62849	62904	0.09%	3:44:43	62890	62890	0%	29:05	6:11	22:04
Y-50	[85]	50	83127	83086	83127	0.05%	6:03:27	83127	83127	0%	31:30	18:28	22:55
Y-60	[85]	60	112055	111884	112126	0.22%	19:57:35	112050.5751	112055	0.0039%	15:38:43	40.5	1:28:35
ste36-1	[12]	36	10287	10287	10287	0%	14:50	10287	10287	0%	2:43	23.9	–
ste36-2	[12]	36	181508	181508	181508	0%	25:25	181508	181508	0%	5:13	16.4	–
ste36-3	[12]	36	101643.5	101643.5	101643.5	0%	24:01	101643.5	101643.5	0%	3:57	31.6	–
ste36-4	[12]	36	95805.5	95805.5	95805.5	0%	16:15	95805.5	95805.5	0%	3:50	22.3	–
ste36-5	[12]	36	91651.5	91651.5	91651.5	0%	17:58	91651.5	91651.5	0%	3:55	9.0	–
sko42-1	[12]	42	25525	25521	25525	0.02%	2:23:09	25525	25525	0%	12:45	34.0	12:45
sko42-2	[12]	42	216120.5	216099.5	216120.5	0.01%	2:43:34	216120.5	216120.5	0%	8:27	12.0	8:06
sko42-3	[12]	42	173267.5	173245.5	173267.5	0.01%	2:47:18	173267.5	173267.5	0%	10:15	17.3	9:20
sko42-4	[12]	42	137615	137379	137615	0.17%	2:53:05	137615	137615	0%	14:45	17.7	12:37
sko42-5	[12]	42	248238.5	248238.5	248238.5	0%	1:08:42	248238.5	248238.5	0%	6:15	10.8	–
sko49-1	[12]	49	40967	40895	41012	0.29%	4:36:21	40967	40967	0%	53:49	9:33	21:58
sko49-2	[12]	49	416178	416142	416178	0.01%	8:27:34	416178	416178	0%	23:59	35.2	18:52
sko49-3	[12]	49	324512	324464	324512	0.02%	8:03:03	324512	324512	0%	25:46	15.8	22:56
sko49-4	[12]	49	236755.5	236718.5	236755.5	0.02%	9:15:14	236755.5	236755.5	0%	29:27	3:09	29:09
sko49-5	[12]	49	666143	666130	666143	0.002%	9:30:22	666143	666143	0%	12:52	51.1	12:15
sko56-1	[12]	56	64024	63971	64027	0.09%	12:36:33	64024	64024	0%	1:01:43	39:51	56:48
sko56-2	[12]	56	496561	496482	496561	0.02%	15:59:27	496561	496561	0%	50:34	58.7	45:48
sko56-3	[12]	56	170449	169644	171032	0.82%	16:22:56	170385.8504	170449	0.0371%	22:25:20	1:16:33	53:33
sko56-4	[12]	56	313388	312656	313497	0.27%	15:17:25	313388	313388	0%	1:23:46	49.7	44:57
sko56-5	[12]	56	592294.5	591915.5	592335.5	0.07%	17:46:46	592294.5	592294.5	0%	1:21:07	10:40	1:01:22

Table 6.3: Results of the bundle approach [44] and (BV) for medium-sized SRFLP instances. The column 'Best' lists all best known upper bounds in the literature. 'Time best' denotes the time required by (BV) to find an ordering with such objective value (or better). The last column shows after which time (BV) improves on the lower bound computed by the bundle approach [44]. The running times are given in sec, in min:sec or h:min:sec respectively.

Instance	Source	n	Best	(BV) LB	(BV) UB	(BV) Gap	(BV) Time	(TV) LB	(TV) UB	(TV) Gap	(TV) Time	Time best	Improve LB (SDP_3)
Y-40	[85]	40	47561	47561	47561	0%	8:26	47561	47561	0%	42:33	4.7	–
Y-45	[85]	45	62890	62890	62890	0%	29:05	62890	62890	0%	2:06:43	0.5	1:25:00
Y-50	[85]	50	83127	83127	83127	0%	31:30	83127	83127	0%	2:53:50	1:33:28	2:04:29
Y-60	[85]	60	112055	112050.5751	112055	0.0039%	15:38:43	112055	112055	0%	35:56:37	2:10:01	7:31:03
ste36-1	[12]	36	10287	10287	10287	0%	2:43	10287	10287	0%	14:19	0.7	–
ste36-2	[12]	36	181508	181508	181508	0%	5:13	181508	181508	0%	20:39	21.1	–
ste36-3	[12]	36	101643.5	101643.5	101643.5	0%	3:57	101643.5	101643.5	0%	19:18	0.1	–
ste36-4	[12]	36	95805.5	95805.5	95805.5	0%	3:50	95805.5	95805.5	0%	19:08	0.2	–
ste36-5	[12]	36	91651.5	91651.5	91651.5	0%	3:55	91651.5	91651.5	0%	20:00	0.0	–
sko42-1	[12]	42	25525	25525	25525	0%	12:45	25525	25525	0%	55:32	12.6	52:33
sko42-2	[12]	42	216120.5	216120.5	216120.5	0%	8:27	216120.5	216120.5	0%	43:38	0.1	43:38
sko42-3	[12]	42	173267.5	173267.5	173267.5	0%	10:15	173267.5	173267.5	0%	57:36	7.4	48:17
sko42-4	[12]	42	137615	137615	137615	0%	14:45	137615	137615	0%	1:06:28	7.7	48:42
sko42-5	[12]	42	248238.5	248238.5	248238.5	0%	6:15	248238.5	248238.5	0%	28:06	0.1	–
sko49-1	[12]	49	40967	40967	40967	0%	53:49	40967	40967	0%	3:55:43	23:24	1:41:17
sko49-2	[12]	49	416178	416178	416178	0%	23:59	416178	416178	0%	1:49:51	12:35	1:43:34
sko49-3	[12]	49	324512	324512	324512	0%	25:46	324512	324512	0%	1:51:59	33.1	1:37:53
sko49-4	[12]	49	236755.5	236755.5	236755.5	0%	29:27	236755.5	236755.5	0%	1:43:02	8:23	1:39:49
sko49-5	[12]	49	666143	666143	666143	0%	12:52	666143	666143	0%	1:08:22	30.7	1:08:22
sko56-1	[12]	56	64024	64024	64024	0%	1:01:43	64024	64024	0%	3:56:16	2:27:50	3:32:03
sko56-2	[12]	56	496561	496561	496561	0%	50:34	496561	496561	0%	3:57:20	0.3	3:18:41
sko56-3	[12]	56	170449	170385.8504	170449	0.0371%	22:25:20	170449	170449	0%	30:29:39	36:30	4:54:24
sko56-4	[12]	56	313388	313388	313388	0%	1:23:46	313388	313388	0%	5:49:57	30.8	3:47:00
sko56-5	[12]	56	592294.5	592294.5	592294.5	0%	1:21:07	592294.5	592294.5	0%	5:11:56	25:33	3:50:10

Table 6.4: Comparison of (BV) and (TV) for medium-sized SRFLP instances. The column 'Best' lists all best known upper bounds in the literature. 'Time best' denotes the time required by (TV) to find an ordering with such objective value (or better). The last column shows after which time (TV) improves on the lower bound computed by the bundle approach [44]. The running times are given in sec, in min:sec or h:min:sec respectively.

Optimal solutions and significantly reduced gaps for large instances. The family of 'sko' instances introduced in [12] contains yet larger instances with $n \in \{64, 72, 81, 100\}$. Additionally, in [6] a set of 20 randomly generated instances with $n \in \{60, 70, 75, 80\}$ was created. Hungerländer & Rendl [43] restricted the number of function evaluations of the bundle method for these instances to 250. This time, we set some time limits for our approach which depend on the instance size and the used version. For instances with $n \leq 81$, we allowed (BV) to run at most two weeks and (TV) to run at most three weeks. For very large instances with 100 facilities, we actually set the maximum time for (BV) to three weeks and for (TV) to four weeks. Unfortunately, the computations had to be stopped ahead of schedule due to an unforeseen maintenance window. Because of this, we cannot provide full computational results for (TV) on these very large instances, since our data logs were buffered and never flushed. This resulted in the loss of all buffered parts when the forced abandonment occurred. However, we can still provide sufficiently good results for (BV) after almost 15 days of running time.

We can see in Table 6.5 that the bundle approach by Hungerländer & Rendl [44] does not solve a single instance and rarely finds the best known solution for $n \geq 64$. As before, (BV) computes the same lower bounds in much less time without any exception. It also solves 24 of the 40 instances to proven optimality and finds for all except four instances the best known solution. In almost all cases where optimality was proven, the overall running time of (BV) is less than the time required by the bundle approach for performing 250 function evaluations. For all unsolved instances with $n \leq 80$, the gaps are very small and nearly reach the upper bounds in three cases. The 'sko' instances with $n = 81$ seem to be much harder than all other instances. (BV) can only solve one of these within the given time limit of two weeks. However, it significantly reduces the optimality gaps compared to the approach by Hungerländer & Rendl [44]. The results for the very large instances with $n = 100$ are intriguing. Although none of these instances can be solved within 15 days, the gaps are very small, especially for two instances. Note that the best known solutions were not found for the other three instances.

Table 6.6 again compares (BV) and (TV) with respect to the considered large instances. (TV) is capable of solving all but two instances with $n \leq 80$. For the two remaining instances, the gaps compared to (BV) are reduced by factor of 2 or 10 respectively. Moreover, (TV) solves one additional instance with 81 facilities and reduces the gaps for all unsolved ones. We want to remark, that our log files indicate that the instance 'sko-81-4' most likely would have been solved with a higher time limit, since the tightness parameter α had a comparatively large value on termination (see Table 6.7) and the gap still decreased significantly after the last reduction step. As we can see, (TV) did not improve the lower bounds of (BV) for $n = 100$. This is not surprising due to the premature termination, since Table 6.7 will show that α was at that time never smaller than 10^{-4}.

				(SDP_3) with bundle method restricted to 250 function evaluations [43]				(BV)				
Instance	Source	n	Best	LB	UB	Gap	Time	LB	UB	Gap	Time	Improve LB (SDP_3)
AKV-60-1	[6]	60	1477834	1477134	1477834	0.05%	12:38:16	1477831.5577	1477834	0.0002%	5:41:33	1:15:06
AKV-60-2	[6]	60	841776	841472	841776	0.04%	11:08:16	841776	841776	0%	3:25:35	1:43:16
AKV-60-3	[6]	60	648337.5	647031.5	648337.5	0.20%	9:51:06	648337.5	648337.5	0%	2:23:52	1:34:03
AKV-60-4	[6]	60	398406	397951	398406	0.11%	10:49:59	398406	398406	0%	3:05:26	1:45:33
AKV-60-5	[6]	60	318805	318792	318805	0.004%	12:39:37	318805	318805	0%	1:09:07	1:06:36
AKV-70-1	[6]	70	1528537	1526359	1528560	0.14%	26:41:34	1528537	1528537	0%	10:02:44	4:44:51
AKV-70-2	[6]	70	1441028	1439122	1441028	0.13%	26:11:27	1441028	1441028	0%	7:15:10	4:00:01
AKV-70-3	[6]	70	1518993.5	1517803.5	1518993.5	0.08%	26:15:14	1518993.5	1518993.5	0%	6:43:46	3:14:28
AKV-70-4	[6]	70	968796	967316	969150	0.19%	27:28:48	968796	968796	0%	5:59:01	2:53:18
AKV-70-5	[6]	70	4218002.5	4213774.5	4218002.5	0.10%	28:16:05	4217999.4003	4218002.5	0.0001%	16:32:28	2:53:19
AKV-75-1	[6]	75	2393456.5	2387590.5	2393600.5	0.25%	37:57:53	2393456.5	2393456.5	0%	31:42:37	3:18:08
AKV-75-2	[6]	75	4321190	4309185	4322492	0.31%	39:28:38	4320968.7261	4321190	0.0051%	159:23:39	4:29:30
AKV-75-3	[6]	75	1248423	1243136	1249251	0.49%	38:21:06	1248423	1248423	0%	20:25:46	4:05:05
AKV-75-4	[6]	75	3941816.5	3936460.5	3941845.5	0.14%	38:42:58	3941816.5	3941816.5	0%	12:47:31	4:13:13
AKV-75-5	[6]	75	1791408	1786154	1791469	0.30%	41:10:37	1791405.7506	1791408	0.0001%	33:05:38	4:06:35
AKV-80-1	[6]	80	2069097.5	2063346.5	2070391.5	0.34%	58:24:49	2069097.5	2069097.5	0%	72:27:53	11:07:38
AKV-80-2	[6]	80	1921136	1918945	1921202	0.12%	58:47:15	1921136	1921136	0%	72:15:33	22:32:59
AKV-80-3	[6]	80	3251368	3245254	3251413	0.19%	58:17:19	3251368	3251368	0%	26:00:25	8:18:17
AKV-80-4	[6]	80	3746515	3739657	3747829	0.22%	58:50:47	3746515	3746515	0%	85:37:47	5:37:30
AKV-80-5	[6]	80	1588885	1585491	1590847	0.34%	58:30:30	1588885	1588885	0%	37:31:15	10:02:13
sko-64-1	[12]	64	96881	96569	97194	0.65%	13:08:05	96881	96881	0%	5:29:27	1:57:33
sko-64-2	[12]	64	634332.5	633420.5	634332.5	0.14%	14:28:38	634332.5	634332.5	0%	3:00:10	1:30:25
sko-64-3	[12]	64	414323.5	412820.5	414384.5	0.38%	14:04:55	414323.5	414323.5	0%	9:54:09	3:19:02
sko-64-4	[12]	64	297129	295145	298155	1.02%	13:55:45	296925.2018	297129	0.0686%	56:45:08	1:08:25
sko-64-5	[12]	64	501922.5	501059.5	502063.5	0.20%	13:53:04	501922.5	501922.5	0%	3:51:29	2:08:59
sko-72-1	[12]	72	139150	138885	139231	0.25%	29:33:19	139150	139150	0%	10:00:24	4:48:16
sko-72-2	[12]	72	711998	707643	715611	0.11%	29:40:41	711781.134	711998	0.0305%	186:20:45	4:19:36
sko-72-3	[12]	72	1054110.5	1048930.5	1061762.5	0.12%	32:38:47	1054063.7076	1054110.5	0.0044%	111:56:07	2:46:06
sko-72-4	[12]	72	919586.5	916229.5	924019.5	0.85%	33:58:28	919586.5	919586.5	0%	15:42:28	2:56:27
sko-72-5	[12]	72	428226.5	426224.5	430288.5	0.95%	31:39:43	428226.5	428226.5	0%	15:52:28	3:20:55
sko-81-1	[12]	81	205106	203424	207063	1.79%	52:44:10	204461.4123	205108	0.3162%	336:00:06	5:39:25
sko-81-2	[12]	81	521391.5	518711.5	526157.5	1.44%	59:58:08	521391.5	521391.5	0%	37:23:17	8:04:32
sko-81-3	[12]	81	970796	962886	979281	1.70%	58:17:40	969286.8873	970796	0.1557%	336:00:06	5:17:11
sko-81-4	[12]	81	2031803	2019058	2035569	0.82%	57:21:49	2031493.3992	2031803	0.0152%	336:00:07	5:15:15
sko-81-5	[12]	81	1302711	1293905	1311166	1.33%	58:59:28	1302377.7003	1302711	0.0256%	333:49:03	4:26:39
sko-100-1	[12]	100	378234	375999	380562	1.21%	191:47:21	378124.5979	378235	0.0292%	359:23:19	113:10:58
sko-100-2	[12]	100	2076008.5	2056997.5	2084924.5	1.36%	201:46:52	2076001.5413	2076008.5	0.0003%	359:28:11	48:25:07
sko-100-3	[12]	100	16145598.5	15987840.5	16216076.5	1.43%	212:38:54	16140148.2822	16145614.5	0.0339%	359:20:52	69:09:21
sko-100-4	[12]	100	3232522	3200643	3263493	1.96%	204:14:39	3232465.2313	3232522	0.0018%	359:51:51	60:57:46
sko-100-5	[12]	100	1033080.5	1021584.5	1040929.5	1.89%	201:29:27	1031671.6220	1033101.5	0.1386%	359:52:21	81:07:09

Table 6.5: Results of the bundle approach [44] and (BV) for large SRFLP instances. The column 'Best' lists all best known upper bounds in the literature. The last column shows after which time (BV) improves on the lower bound computed by the bundle approach [44]. The running times are given in sec, in min:sec or h:min:sec respectively.

Instance	Source	n	Best	(BV) LB	(BV) UB	(BV) Gap	(BV) Time	(TV) LB	(TV) UB	(TV) Gap	(TV) Time	Improve LB (SDP_3)
AKV-60-1	[6]	60	1477834	1477831.5577	1477834	0.0002%	5:41:33	1477834	1477834	0%	13:14:51	4:48:08
AKV-60-2	[6]	60	841776	841776	841776	0%	3:25:35	841776	841776	0%	13:54:38	9:19:47
AKV-60-3	[6]	60	648337.5	648337.5	648337.5	0%	2:23:52	648337.5	648337.5	0%	8:28:10	4:36:29
AKV-60-4	[6]	60	398406	398406	398406	0%	3:05:26	398406	398406	0%	8:58:31	4:37:14
AKV-60-5	[6]	60	318805	318805	318805	0%	1:09:07	318805	318805	0%	4:41:03	4:00:22
AKV-70-1	[6]	70	1528537	1528537	1528537	0%	10:02:44	1528537	1528537	0%	29:25:16	9:48:23
AKV-70-2	[6]	70	1441028	1441028	1441028	0%	7:15:10	1441028	1441028	0%	46:18:51	17:30:35
AKV-70-3	[6]	70	1518993.5	1518993.5	1518993.5	0%	6:43:46	1518993.5	1518993.5	0%	19:01:52	9:26:05
AKV-70-4	[6]	70	968796	968796	968796	0%	5:59:01	968796	968796	0%	20:17:55	8:42:32
AKV-70-5	[6]	70	4218002.5	4217999.4003	4218002.5	0.0001%	16:32:28	4218002.5	4218002.5	0%	40:52:33	7:06:47
AKV-75-1	[6]	75	2393456.5	2393456.5	2393456.5	0%	31:42:37	2393456.5	2393456.5	0%	70:01:58	11:21:23
AKV-75-2	[6]	75	4321190	4320968.7261	4321190	0.0051%	159:23:39	4321063.2399	4321190	0.0029%	504:00:03	12:56:57
AKV-75-3	[6]	75	1248423	1248423	1248423	0%	20:25:46	1248423	1248423	0%	41:36:25	15:14:41
AKV-75-4	[6]	75	3941816.5	3941816.5	3941816.5	0%	12:47:31	3941816.5	3941816.5	0%	55:03:22	16:12:49
AKV-75-5	[6]	75	1791408	1791405.7506	1791408	0.0001%	33:05:38	1791408	1791408	0%	62:52:51	11:43:20
AKV-80-1	[6]	80	2069097.5	2069097.5	2069097.5	0%	72:27:53	2069097.5	2069097.5	0%	128:10:57	19:44:21
AKV-80-2	[6]	80	1921136	1921136	1921136	0%	72:15:33	1921136	1921136	0%	192:58:59	21:55:04
AKV-80-3	[6]	80	3251368	3251368	3251368	0%	26:00:25	3251368	3251368	0%	50:36:34	19:51:16
AKV-80-4	[6]	80	3746515	3746515	3746515	0%	85:37:47	3746515	3746515	0%	293:58:16	66:27:00
AKV-80-5	[6]	80	1588885	1588885	1588885	0%	37:31:15	1588885	1588885	0%	138:38:47	26:31:04
sko-64-1	[12]	64	96881	96881	96881	0%	5:29:27	96881	96881	0%	20:07:11	7:04:10
sko-64-2	[12]	64	634332.5	634332.5	634332.5	0%	3:00:10	634332.5	634332.5	0%	11:34:38	5:41:09
sko-64-3	[12]	64	414323.5	414323.5	414323.5	0%	9:54:09	414323.5	414323.5	0%	19:46:39	7:30:48
sko-64-4	[12]	64	297129	296925.2018	297129	0.0686%	56:45:08	297113.0325	297129	0.0054%	369:21:11	7:13:20
sko-64-5	[12]	64	501922.5	501922.5	501922.5	0%	3:51:29	501922.5	501922.5	0%	17:29:26	10:26:56
sko-72-1	[12]	72	139150	139150	139150	0%	10:00:24	139150	139150	0%	91:20:37	42:31:27
sko-72-2	[12]	72	711998	711781.134	711998	0.0305%	186:20:45	711998	711998	0%	182:19:56	23:54:00
sko-72-3	[12]	72	1054110.5	1054063.7076	1054110.5	0.0044%	111:56:07	1054110.5	1054110.5	0%	67:45:48	9:57:24
sko-72-4	[12]	72	919586.5	919586.5	919586.5	0%	15:42:28	919586.5	919586.5	0%	55:50:16	15:04:02
sko-72-5	[12]	72	428226.5	428226.5	428226.5	0%	15:52:28	428226.5	428226.5	0%	73:26:48	28:14:14
sko-81-1	[12]	81	205106	204461.4123	205108	0.3162%	336:00:06	204611.2638	205135	0.2560%	504:00:07	36:48:21
sko-81-2	[12]	81	521391.5	521391.5	521391.5	0%	37:23:17	521391.5	521391.5	0%	157:05:34	44:28:26
sko-81-3	[12]	81	970796	969286.8873	970796	0.1557%	336:00:06	969767.2487	970796	0.1061%	504:00:07	38:58:28
sko-81-4	[12]	81	2031803	2031493.3992	2031803	0.0152%	336:00:07	2031625.5296	2031803	0.0087%	504:00:05	67:39:41
sko-81-5	[12]	81	1302711	1302377.7003	1302711	0.0256%	333:49:03	1302711	1302711	0%	344:01:47	19:24:47
sko-100-1	[12]	100	378234	378124.5979	378235	0.0292%	359:23:19	377120.3526	378251	0.2998%	164:18:18	151:59:35
sko-100-2	[12]	100	2076008.5	2076001.5413	2076008.5	0.0003%	359:28:11	2073004.1288	2076008.5	0.1449%	235:17:52	53:19:12
sko-100-3	[12]	100	16145598.5	16140148.2822	16145614.5	0.0339%	359:20:52	16125725.5471	16145614.5	0.1233%	332:41:42	79:40:03
sko-100-4	[12]	100	3232522	3232465.2313	3232522	0.0018%	359:51:51	3231547.3942	3232522	0.0302%	378:42:22	71:27:12
sko-100-5	[12]	100	1033080.5	1031671.6220	1033101.5	0.1386%	359:52:21	1030595.2021	1033102.5	0.2433%	377:11:41	112:16:41

Table 6.6: Comparison of (BV) and (TV) for large SRFLP instances. The column 'Best' lists all best known upper bounds in the literature. The last column shows after which time (TV) improves on the lower bound computed by the bundle approach [44]. The running times are given in sec, in min:sec or h:min:sec respectively.

Analysis of (BV) and (TV). Let us now compare (BV) and (TV) in more detail. First, we want to remark that (BV) seems to be more numerical stable than (TV) when applied to large instances. Whereas it rarely happened in (BV), the L-BFGS-B solver aborted its optimization of the dual variables more often in (TV). This mostly happened when the tightness parameter α took values smaller than 10^{-4} and was more likely with increasing instance size. But we also noticed that this behavior is influenced by our enforced minimum number of function evaluations per iteration in (TV). This can impose too accurate solutions which the L-BFGS-B solver cannot handle with hundreds of thousands of variables. Our implemented safeguard against this failure (see Section 5.3) does a decent job and straightforwardly allows to continue the bounding procedure.

A more comprehensive overview of (BV) and (TV) for large instances is given in Table 6.7. For each instance, it shows different characteristics of both versions at the moment of termination. 'Evals' denotes the total number of function evaluations performed. The columns '$\alpha_{\min}$' show the respective smallest value of the tightness parameter α. Table 6.7 also lists in the columns 'Cuts' how many cuts were involved in the final relaxation. The subsequent columns divide the whole number of cuts into the respective number of triangle, pentagonal, heptagonal and hexagonal inequalities.

We can see in Table 6.7 that the number of function evaluations is the main indicator for higher running times. It stands out that the ratio of evaluations to running time is typically much higher in (BV) than in (TV). However, they are nearly equal for very large instances. This is mainly due to the much more expensive separation routine in (TV) and the different settings of the L-BFGS-B solver in both versions. Recall that the number of memory corrections for (BV) is chosen adaptively with respect to the instance size, whereas a fixed value is used in (TV) (see Section 5.3). On the one hand, this leads, combined with a higher number of cuts, to a higher computational effort compared to (BV), especially since many more nearly inactive inequalities are involved in (TV). On the other hand, we observed that it pays of when other inequalities than triangle and starlike pentagonal inequalities are considered. It yields much more accurate solutions in this case and allows to solve many more instances to optimality. Table 6.7 also shows that a significantly higher number of function evaluations is required as the instance size grows.

Consider now the smallest values of the tightness parameter α in both versions. Actually we cannot compare (BV) and (TV) directly with respect to α, since their reduction schedules of α are different, i.e., α attains different values in both versions. However, we can clearly see that the final value of α is almost always smaller in (BV) when both versions can prove optimality. Besides the number of memory corrections, this is mainly due to a tighter relaxation in (TV) by involving and considering more inequalities. We could reproduce this behavior with equal reduction schedules and found out that (TV) often can prove optimality with higher values of α than (BV). Note that under all cases where optimality was proven in Table 6.7, the smallest value of α was about $6.1 \cdot 10^{-6}$. We want to remark that an even smaller value of $3.8 \cdot 10^{-6}$ was required by (TV) to solve the instance 'Y-60'. Whenever α attained a yet smaller value, the lower bound made tiny progress.

Finally, let us consider the respective number of cuts in Table 6.7. Typically, (TV) involves much more inequalities than (BV). The ratio of triangle to starlike pentagonal inequalities

in (BV) is always about 5 : 1. In contrast, there is no clear ratio of triangle to pentagonal inequalities in (TV), but the numbers are definitely a bit more balanced. There are even some cases where more pentagonal than triangle inequalities are involved, especially for the 'sko' instances. Note that also the number of triangle cuts in (TV) is sometimes smaller than in (BV). We can also see that always a decent number of heptagonal cuts is used in (TV). However, except for two instances, only relatively few hexagonal inequalities are involved. We conclude from this observations with respect to the stronger lower bounds of (TV) compared to (BV) that the heuristic separation of pentagonal and heptagonal inequalities pays off, but the hexagonal inequalities presumably have a little contribution to the lower bound and should be omitted. All in all, (BV) and (TV) both seem to be able to handle different types of inequalities adequately. Although our adjusted bounding procedure admits a huge number of inequalities, it still ensures suitable numbers of the respective types, since triangle inequalities are in general more valuable than pentagonal inequalities and so on.

Instance	(BV) Evals	Gap	$\alpha_{\min}$	Time	Cuts	Tri.	Pent.	(TV) Evals	Gap	$\alpha_{\min}$	Time	Cuts	Tri.	Pent.	Hept.	Hex.
AKV-60-1	80326	0.0002%	1.2e-05	5:41:33	84108	101444	16231	124726	0%	3.1e-05	13:14:51	232553	114798	79919	37835	1
AKV-60-2	41510	0%	9.8e-05	3:25:35	115754	96637	19117	94570	0%	1.2e-04	13:54:38	179463	91530	41796	45984	153
AKV-60-3	28206	0%	9.8e-05	2:23:52	93423	70516	22907	52208	0%	1.2e-04	8:28:10	183866	80147	50213	52586	920
AKV-60-4	33072	0%	6.1e-06	3:05:26	157800	116715	41085	53808	0%	7.6e-06	8:58:31	383685	228415	122036	23222	10012
AKV-60-5	12509	0%	2.0e-04	1:09:07	145666	126032	19634	31150	0%	1.2e-04	4:41:03	172886	88233	44630	35845	4178
AKV-70-1	60545	0%	4.9e-05	10:02:44	95653	75258	20395	128427	0%	6.1e-05	29:25:16	197890	90053	44088	63687	62
AKV-70-2	42413	0%	9.8e-05	7:15:10	137358	106764	30594	109165	0%	2.4e-04	46:18:51	193921	127043	25977	40851	50
AKV-70-3	37044	0%	4.9e-05	6:43:46	218082	174624	43458	76331	0%	6.1e-05	19:01:52	285906	162655	110540	11915	796
AKV-70-4	27135	0%	9.8e-05	5:59:01	186906	152844	34062	73484	0%	1.2e-04	20:17:55	265654	114833	74599	27661	48561
AKV-70-5	108934	0.0001%	1.2e-05	16:32:28	162320	127822	20062	201109	0%	7.6e-06	40:52:33	184249	154849	23483	5888	29
AKV-75-1	127590	0%	2.4e-05	31:42:37	126414	85480	40934	223180	0%	3.1e-05	70:01:58	242481	143002	62586	36843	50
AKV-75-2	504708	0.0051%	1.9e-07	159:23:39	139189	128932	14467	1162556	0.0029%	9.5e-07	504:00:03	277016	216471	48675	11856	14
AKV-75-3	78269	0%	1.2e-05	20:25:46	150655	123512	27143	108426	0%	1.5e-05	41:36:25	422479	269407	96775	56235	62
AKV-75-4	51469	0%	2.0e-04	12:47:31	123258	99258	24000	120215	0%	2.4e-04	55:03:22	307495	188974	106763	10991	767
AKV-75-5	107252	0.0001%	1.2e-05	33:05:38	178638	209084	37516	174094	0%	7.6e-06	62:52:51	459159	316242	95514	46960	443
AKV-80-1	131454	0%	2.4e-05	72:27:53	134902	106536	28366	264625	0%	3.1e-05	128:10:57	301536	174612	112581	14316	27
AKV-80-2	160970	0%	6.1e-06	72:15:33	132421	106914	25507	453580	0%	3.1e-05	192:58:59	221629	120018	90404	11195	12
AKV-80-3	42918	0%	9.8e-05	26:00:25	152944	129536	23408	97784	0%	2.4e-04	50:36:34	319200	186548	100076	26444	6132
AKV-80-4	230653	0%	2.4e-05	85:37:47	74890	62241	12649	531759	0%	1.5e-05	293:58:16	270730	205984	46337	18390	19
AKV-80-5	91576	0%	4.9e-05	37:31:15	160685	121137	39548	234280	0%	6.1e-05	138:38:47	278022	107659	90812	79145	406
sko-64-1	32969	0%	1.2e-05	5:29:27	125593	99429	26164	63905	0%	1.5e-05	20:07:11	483061	197310	214928	61317	9506
sko-64-2	20622	0%	9.8e-05	3:00:10	182186	133096	49090	50928	0%	1.2e-04	11:34:38	335404	133521	133782	67767	334
sko-64-3	30872	0%	4.9e-05	9:54:09	206456	147117	59339	74743	0%	6.1e-05	19:46:39	440187	168974	233367	37339	507
sko-64-4	334113	0.0686%	2.4e-08	56:45:08	244452	205273	30903	1056569	0.0054%	2.4e-07	369:21:11	596234	352960	175327	17852	1285
sko-64-5	27765	0%	9.8e-05	3:51:29	186840	152064	34776	69737	0%	1.2e-04	17:29:26	424657	184198	159735	78270	2454
sko-72-1	37791	0%	1.2e-05	10:00:24	180318	134869	45449	89015	0%	1.5e-05	91:20:37	444319	165137	202254	23309	53619
sko-72-2	579382	0.0305%	2.4e-08	186:20:45	253727	198415	35083	416307	0%	1.5e-05	182:19:56	661809	271000	308449	80020	2340
sko-72-3	443832	0.0044%	9.5e-08	111:56:07	197189	159823	32143	168451	0%	3.1e-05	67:45:48	502923	203007	192994	106334	588
sko-72-4	64617	0%	2.4e-05	15:42:28	205204	144886	60318	138966	0%	1.5e-05	55:50:16	557898	281158	159277	107257	10206
sko-72-5	59029	0%	1.2e-05	15:52:28	187710	156181	31529	161422	0%	6.1e-05	73:26:48	629840	271096	283269	62616	12859
sko-81-1	464943	0.3162%	1.2e-08	336:00:06	378041	321897	56144	535917	0.2560%	9.5e-07	504:00:07	821378	392808	390454	36631	1485
sko-81-2	75099	0%	1.2e-05	37:23:17	244181	203558	40623	228014	0%	1.5e-05	157:05:34	643000	253624	262226	124276	2874
sko-81-3	648293	0.1557%	2.4e-08	336:00:06	315302	247266	68036	644245	0.1061%	6.1e-05	504:00:07	660910	286910	323605	49594	801
sko-81-4	562829	0.0152%	9.5e-08	336:00:07	276071	210943	65128	506882	0.0087%	1.2e-04	504:00:05	582808	245261	260373	77069	105
sko-81-5	654902	0.0256%	2.4e-08	333:49:03	287593	248037	39556	508629	0%	1.5e-05	344:01:47	636893	243657	270331	121017	1888
sko-100-1	240414	0.0292%	7.6e-06	359:23:19	1219998	–	–	102184	0.2998%	1.2e-04	164:18:18	1023053	–	–	–	–
sko-100-2	300477	0.0003%	7.6e-06	359:28:11	751339	–	–	189999	0.1449%	4.9e-04	235:17:52	628212	–	–	–	–
sko-100-3	308796	0.0339%	1.2e-04	359:20:52	776886	–	–	292510	0.1233%	2.0e-03	332:41:42	562712	–	–	–	–
sko-100-4	307233	0.0018%	1.5e-05	359:51:51	638922	–	–	315014	0.0302%	2.4e-04	378:42:22	594726	–	–	–	–
sko-100-5	259381	0.1386%	1.5e-05	359:52:21	1074521	–	–	288202	0.2433%	1.2e-04	377:11:41	790059	–	–	–	–

Table 6.7: Detailed comparison of (BV) and (TV) on large instances. 'Evals' denotes the number of function evaluations performed and $\alpha_{\min}$ the smallest value of the tightness parameter α. The column 'Cuts' shows the number of involved inequalities when the computation was stopped, followed by the number of cuts of each particular type. A '–' sign indicates that no data is available for the respective instance.

Chapter 7

Conclusions and Future Research

In this thesis, we evaluated several lower bounding and exact approaches for the single row facility layout problem (SRFLP). Moreover, we proposed a new semidefinite approach for SRFLP that involves much tighter relaxations and a more suitable algorithmic treatment of these relaxations than all prior approaches. We demonstrated that existing approaches allow substantial improvement in the sense that stronger and optimal lower bounds, even for large instances, can be computed in significantly less time.

Review. In Chapter 2, we presented and discussed several LP-based approaches for SRFLP. We argued that these approaches are only reliable for instances with 40 facilities or less, and hence, that stronger semidefinite bounds are the most promising solution approach for large SRFLP instances. Several well-known semidefinite relaxations for SRFLP were deduced in Chapter 3. We proved that these relaxations are indeed stronger than the corresponding basic linear relaxations. However, we also revealed many practical issues that arise when these semidefinite relaxations are solved by standard interior-point methods.

We addressed these complications in Chapter 4, where we presented two algorithmic approaches based on Lagrangian duality. The first approach by Hungerländer & Rendl [44] led to a non-smooth optimization problem that is solved by a bundle method. In contrast, we proposed to use a different approach involving nonstandard semidefinite bounds, leading to a smooth optimization problem that can be solved efficiently by a quasi-Newton method. We illustrated that the latter approach is much more efficient from a theoretical point of view and allows to consider further constraint classes more straightforwardly.

To exploit our proposed algorithmic method, we tightened in Chapter 5 the existing semidefinite relaxations with a subset of the so-called pentagonal inequalities that is particularly suited for SRFLP. Additionally, we suggested two heuristic separation routines for further, more generic valid inequalities, and some primal heuristics that produce good feasible layouts. We then presented a semidefinite bounding procedure for SRFLP in more detail that appropriately combines the used algorithmic approach with our new semidefinite relaxations. We exploited the flexibility of this bounding procedure and established two different versions which esteem accuracy and computational effort differently. This flexibility is particularly interesting for a branch-and-bound approach for which we provided preliminary insights,

including some obstacles, as well as useful tools such as branching rules especially designed for SRFLP.

The computational results in Chapter 6 show that all prior semidefinite approaches were computationally too expensive or produced improvable bounds. We achieved significant progress by tightening the semidefinite relaxations theoretically and by using a more suited algorithmic solution approach. Indeed, our novel approach for SRFLP greatly exceeds our expectations. We increased the maximum size of successfully solved instances from $n = 42$ in the literature to $n = 81$. This huge improvement is not limited to particular instances and only a few benchmark instances remain unsolved. Moreover, we significantly improved all best known duality gaps for large instances with up to 100 facilities and computed the lower bounds of other approaches in significantly less time. However, there are still many possible ways to improve our approach and we are excited to apply some of our ideas to other problems.

Future work. An obvious way to further improve the lower bounds for some unsolved instances is to design a complete branch-and-bound algorithm that uses our novel semidefinite bounds. We already presented some important ingredients of such an approach in Section 5.4 and showed that appropriate fixations of variables can lead to a substantial speedup of the bounding procedure. However, the full implementation is not yet finished and a lot of details still have to be handled.

Moreover, we could further tighten the semidefinite relaxations and make them more consistent by adding suitable subsets of pentagonal or heptagonal inequalities. Analogous to the starlike pentagonal inequalities, the inclusion of starlike heptagonal inequalities seems to be a promising idea. Since there are $\mathcal{O}(n^8)$ of these inequalities, heuristic separation routines like the ones presented in Section 5.1 should then be used.

Beside possible refinements of the bounding procedure, one could also improve the bound computation and optimization itself. For example, the expensive linear algebra computations could be performed on a GPU instead of a CPU. This is especially interesting for very large-scale instances with more than 100 facilities. Finally, the quasi-Newton solver could be adapted more specifically for SRFLP, or a second-order semismooth Newton method could be used instead (see [52]).

We also want to apply the presented approach to other combinatorial problems such as the quadratic ordering problem or one of its other special cases. To use our approach for this kind of problems, we only have to adjust the objective function. It could be the case that a branch-and-bound approach would then be necessary. Especially the weighted betweenness problem might be an interesting problem to consider as it is a generalization of SRFLP. However, it can still be modeled as an integer linear program using betweenness variables [41]. Hence, the starlike pentagonal inequalities would still be a suitable enhancement of existing semidefinite relaxations. Moreover, we are excited to see how the combination of the bounding procedure and heuristically separated pentagonal and heptagonal inequalities performs on more general binary quadratic optimization problems.

Bibliography

[1] André RS Amaral. On the exact solution of a facility layout problem. *European Journal of Operational Research*, 173(2):508–518, 2006.

[2] André RS Amaral. Enhanced local search applied to the single-row facility layout problem. In *Proc. XL Brazilian Symp. Oper. Res.*, pages 1638–1647, 2008.

[3] André RS Amaral. An exact approach to the one-dimensional facility layout problem. *Operations Research*, 56(4):1026–1033, 2008.

[4] André RS Amaral. A new lower bound for the single row facility layout problem. *Discrete Applied Mathematics*, 157(1):183–190, 2009.

[5] André RS Amaral and Adam N Letchford. A polyhedral approach to the single row facility layout problem. *Mathematical programming*, 141(1-2):453–477, 2013.

[6] Miguel F Anjos, Andrew Kennings, and Anthony Vannelli. A semidefinite optimization approach for the single-row layout problem with unequal dimensions. *Discrete Optimization*, 2(2):113–122, 2005.

[7] Miguel F Anjos and Jean B Lasserre. *Handbook on semidefinite, conic and polynomial optimization*, volume 166. Springer Science & Business Media, 2011.

[8] Miguel F Anjos and Anthony Vannelli. Globally optimal solutions for large single-row facility layout problems. Technical report, Citeseer, 2006.

[9] Miguel F Anjos and Anthony Vannelli. On the computational performance of a semidefinite programming approach to single row layout problems. In *Operations Research Proceedings 2005*, pages 277–282. Springer, 2006.

[10] Miguel F Anjos and Anthony Vannelli. Computing globally optimal solutions for single-row layout problems using semidefinite programming and cutting planes. *INFORMS Journal on Computing*, 20(4):611–617, 2008.

[11] Miguel F Anjos and Manuel VC Vieira. Mathematical optimization approaches for facility layout problems: The state-of-the-art and future research directions. *European Journal of Operational Research*, 261(1):1–16, 2017.

[12] Miguel F Anjos and Ginger Yen. Provably near-optimal solutions for very large single-row facility layout problems. *Optimization Methods & Software*, 24(4-5):805–817, 2009.

[13] Soumen Atta and Priya RS Mahapatra. Population-based improvement heuristic with local search for single-row facility layout problem. *Sādhanā*, 44(11):222, 2019.

[14] Alexandre Belloni and Claudia Sagastizábal. Dynamic bundle methods: Application to combinatorial optimization. 2004.

[15] Brian Borchers. CSDP, a C library for semidefinite programming. *Optimization methods and Software*, 11(1-4):613–623, 1999.

[16] Brian Borchers and Joseph G Young. Implementation of a primal–dual method for SDP on a shared memory parallel architecture. *Computational Optimization and Applications*, 37(3):355–369, 2007.

[17] Christoph Buchheim, Angelika Wiegele, and Lanbo Zheng. Exact algorithms for the quadratic linear ordering problem. *INFORMS Journal on Computing*, 22(1):168–177, 2010.

[18] Thomas Christof. *Low-dimensional 0/1-polytopes and branch-and-cut in combinatorial optimization*. Berichte aus der Informatik. Shaker, Aachen, als ms. gedr. edition, 1997.

[19] Camille Coti, Etienne Leclercq, Frédéric Roupin, and Franck Butelle. Solving 0-1 quadratic problems with two-level parallelization of the BiqCrunch solver. In *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 445–452. IEEE, 2017.

[20] Dilip Datta, André RS Amaral, and José R Figueira. Single row facility layout problem using a permutation-based genetic algorithm. *European Journal of Operational Research*, 213(2):388–394, 2011.

[21] Michel M Deza and Monique Laurent. *Geometry of cuts and metrics*, volume 15. Springer, 2009.

[22] Housni Djellab and Michel Gourgand. A new heuristic procedure for the single-row facility layout problem. *International Journal of Computer Integrated Manufacturing*, 14(3):270–280, 2001.

[23] Ilse Fischer, Gerald Gruber, Franz Rendl, and Renata Sotirov. Computational experience with a bundle approach for semidefinite cutting plane relaxations of Max-Cut and equipartition. *Mathematical Programming*, 105(2-3):451–469, 2006.

[24] Michael R Garey, David S Johnson, and Larry Stockmeyer. Some simplified NP-complete problems. In *Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 47–63, 1974.

[25] Carl Geiger and Christian Kanzow. *Theorie und Numerik restringierter Optimierungsaufgaben*. Springer-Verlag, 2013.

[26] Michel X Goemans and David P Williamson. .878-approximation algorithms for max cut and max 2sat. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 422–431, 1994.

[27] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[28] Martin Grötschel. *The sharpest cut: The impact of Manfred Padberg and his work*. SIAM, 2004.

[29] Martin Grötschel, Michael Jünger, and Gerhard Reinelt. Facets of the linear ordering polytope. *Mathematical programming*, 33(1):43–60, 1985.

[30] Jian Guan and Geng Lin. Hybridizing variable neighborhood search with ant colony optimization for solving the single row facility layout problem. *European Journal of Operational Research*, 248(3):899–909, 2016.

[31] Nicolo Gusmeroli and Angelika Wiegele. EXPEDIS: An exact penalty method over discrete sets. *arXiv preprint arXiv:1912.09739*, 2019.

[32] Lawrence H Harper. Optimal assignments of numbers to vertices. *Journal of the Society for Industrial and Applied Mathematics*, 12(1):131–135, 1964.

[33] Christoph Helmberg. Semidefinite programming for combinatorial optimization, 2000.

[34] Christoph Helmberg and Krzysztof C Kiwiel. A spectral bundle method with bounds. *Mathematical Programming*, 93(2):173–194, 2002.

[35] Christoph Helmberg and Franz Rendl. A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization*, 10(3):673–696, 2000.

[36] Christoph Helmberg and Robert Weismantel. Cutting plane algorithms for semidefinite relaxations. *Fields Institute Communications*, 18:197–213, 1998.

[37] Sunderesh S Heragu and Andrew Kusiak. Machine layout problem in flexible manufacturing systems. *Operations research*, 36(2):258–268, 1988.

[38] Sunderesh S Heragu and Andrew Kusiak. Efficient models for the facility layout problem. *European Journal of Operational Research*, 53(1):1–13, 1991.

[39] Nicholas J Higham. Computing a nearest symmetric positive semidefinite matrix. *Linear algebra and its applications*, 103:103–118, 1988.

[40] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex analysis and minimization algorithms I: Fundamentals*, volume 305. Springer science & business media, 2013.

[41] Philipp Hungerländer. *Semidefinite approaches to ordering problems*. PhD thesis, Alpen-Adria-Universitaet Klagenfurt, 2012.

[42] Philipp Hungerländer. Single-row equidistant facility layout as a special case of single-row facility layout. *International Journal of Production Research*, 52(5):1257–1268, 2014.

[43] Philipp Hungerländer and Franz Rendl. A computational study and survey of methods for the single-row facility layout problem. *Computational Optimization and Applications*, 55(1):1–20, 2013.

[44] Philipp Hungerländer and Franz Rendl. Semidefinite relaxations of ordering problems. *Mathematical Programming*, 140(1):77–97, 2013.

[45] Richard M Karp and Michael Held. Finite-state processes and dynamic programming. *SIAM Journal on Applied Mathematics*, 15(3):693–718, 1967.

[46] Ravi Kothari and Diptesh Ghosh. Insertion based Lin–Kernighan heuristic for single row facility layout. *Computers & Operations Research*, 40(1):129–136, 2013.

[47] Ravi Kothari and Diptesh Ghosh. Tabu search for the single row facility layout problem using exhaustive 2-opt and insertion neighborhoods. *European Journal of Operational Research*, 224(1):93–100, 2013.

[48] Ravi Kothari and Diptesh Ghosh. An efficient genetic algorithm for single row facility layout. *Optimization Letters*, 8(2):679–690, 2014.

[49] Ravi Kothari and Diptesh Ghosh. A scatter search algorithm for the single row facility layout problem. *Journal of Heuristics*, 20(2):125–142, 2014.

[50] Nathan Krislock, Jérôme Malick, and Frédéric Roupin. Improved semidefinite bounding procedure for solving max-cut problems to optimality. *Mathematical Programming*, 143(1-2):61–86, 2014.

[51] Nathan Krislock, Jérôme Malick, and Frédéric Roupin. Computational results of a semidefinite branch-and-bound algorithm for k-cluster. *Computers & Operations Research*, 66:153–159, 2016.

[52] Nathan Krislock, Jérôme Malick, and Frédéric Roupin. Biqcrunch: A semidefinite branch-and-bound method for solving binary quadratic problems. *ACM Transactions on Mathematical Software (TOMS)*, 43(4):1–23, 2017.

[53] K Ravi Kumar, George C Hadjinicola, and Ting-li Lin. A heuristic procedure for the single-row facility layout problem. *European Journal of Operational Research*, 87(1):65–73, 1995.

[54] Lian Kunlei, Zhang Chaoyong, Gaoa Liang, and Shaoa Xinyu. Single row facility layout problem using an imperialist competitive algorithm. In *Proceedings from the 41st International Conference on Computers & Industrial Engineering*, 2011.

[55] Monique Laurent and Svatopluk Poljak. On a positive semidefinite relaxation of the cut polytope. *Linear Algebra and its Applications*, 223(224):439–461, 1995.

[56] Monique Laurent and Svatopluk Poljak. Gap inequalities for the cut polytope. *European Journal of Combinatorics*, 17(2):233 – 254, 1996.

[57] Claude Lemaréchal. Lagrangian relaxation. In *Computational combinatorial optimization*, pages 112–156. Springer, 2001.

[58] Weiguo Liu and Anthony Vannelli. Generating lower bounds for the linear arrangement problem. *Discrete applied mathematics*, 59(2):137–151, 1995.

[59] László Lovász and Alexander Schrijver. Cones of matrices and set-functions and 0–1 optimization. *SIAM journal on optimization*, 1(2):166–190, 1991.

[60] Robert Love and Jsun Wong. On solving a one-dimensional space allocation problem with integer programming. *INFOR: Information Systems and Operational Research*, 14(2):139–143, 1976.

[61] Jérôme Malick. The spherical constraint in Boolean quadratic programs. *Journal of Global Optimization*, 39(4):609–622, 2007.

[62] Jérôme Malick and Frédéric Roupin. On the bridge between combinatorial optimization and nonlinear optimization: a family of semidefinite bounds for 0–1 quadratic problems leading to quasi-newton methods. *Mathematical Programming*, 140(1):99–124, 2013.

[63] José L Morales and Jorge Nocedal. Remark on "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization". *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1–4, 2011.

[64] Christopher E Nugent, Thomas E Vollmann, and John Ruml. An experimental comparison of techniques for the assignment of facilities to locations. *Operations research*, 16(1):150–173, 1968.

[65] Takashi Obata. *Quadratic assignment problem: evaluation of exact and heuristic algorithms*. PhD thesis, 1979.

[66] Marcus Oswald. *Weighted consecutive ones problems*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2003.

[67] Chao Ou-Yang and Amalia Utamima. Hybrid estimation of distribution algorithm for solving single row facility layout problem. *Computers & Industrial Engineering*, 66(1):95–103, 2013.

[68] Gintaras Palubeckis. A branch-and-bound algorithm for the single-row equidistant facility layout problem. *OR spectrum*, 34(1):1–21, 2012.

[69] Gintaras Palubeckis. Fast local search for single row facility layout. *European Journal of Operational Research*, 246(3):800–814, 2015.

[70] Gintaras Palubeckis. Single row facility layout using multi-start simulated annealing. *Computers & Industrial Engineering*, 103:1–16, 2017.

[71] Jean-Claude Picard and Maurice Queyranne. On the one-dimensional space allocation problem. *Operations Research*, 29(2):371–391, 1981.

[72] Franz Rendl, Giovanni Rinaldi, and Angelika Wiegele. A branch and bound algorithm for Max-Cut based on combining semidefinite and polyhedral relaxations. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 295–309. Springer, 2007.

[73] Franz Rendl, Giovanni Rinaldi, and Angelika Wiegele. Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 121(2):307, 2010.

[74] Hamed Samarghandi and Kourosh Eshghi. An efficient tabu algorithm for the single row facility layout problem. *European Journal of Operational Research*, 205(1):98–105, 2010.

[75] Hamed Samarghandi, Pouria Taabayan, and Farzad F Jahantigh. A particle swarm optimization for the single row facility layout problem. *Computers & Industrial Engineering*, 58(4):529–534, 2010.

[76] Sujeevraja Sanjeevi and Kiavash Kianfar. A polyhedral study of triplet formulation for single row facility layout problem. *Discrete Applied Mathematics*, 158(16):1861–1867, 2010.

[77] Bhaba R Sarker. *The amoebic matrix and one-dimensional machine location problems*. ProQuest LLC, Ann Arbor, MI, 1989. Thesis (Ph.D.)–Texas A&M University.

[78] Donald M Simmons. One-dimensional space allocation: an ordering algorithm. *Operations Research*, 17(5):812–826, 1969.

[79] Donald M Simmons. A further note on one-dimensional space allocation. *Operations Research*, 19(1):249–249, 1971.

[80] Maghsud Solimanpur, Prem Vrat, and Ravi Shankar. An ant algorithm for the single row layout problem in flexible manufacturing systems. *Computers & Operations Research*, 32(3):583–598, 2005.

[81] JK Suryanarayanan, Bruce L Golden, and Qi Wang. A new heuristic for the linear placement problem. *Computers & operations research*, 18(3):255–262, 1991.

[82] Albert W Tucker. On directed graphs and integer programs. In *Symposium on Combinatorial Problems, Princeton University*, 1960.

[83] Henry Wolkowicz, Romesh Saigal, and Lieven Vandenberghe. *Handbook of semidefinite programming: theory, algorithms, and applications*, volume 27. Springer Science & Business Media, 2012.

[84] Daniel H Younger. Minimum feedback arc sets for a directed graph. *IEEE Transactions on Circuit Theory*, 10(2):238–245, 1963.

[85] Junfang Yu and Bhaba R Sarker. Directional decomposition heuristic for a linear machine-cell location problem. *European Journal of Operational Research*, 149(1):142–184, 2003.

[86] Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560, 1997.