Philipp Hungerländer

Semidefinite Approaches to Ordering Problems

DISSERTATION

zur Erlangung des akademischen Grades
Doktor der Technischen Wissenschaften

Alpen-Adria-Universität Klagenfurt
Fakultät für Technische Wissenschaften

1. Begutachter: Prof. Dr. Franz Rendl
Institut für Mathematik, Universität Klagenfurt
2. Begutachter: Prof. Dr. Michael Jünger
Institut für Informatik, Universität zu Köln

Jänner 2012

Ehrenwörtliche Erklärung

Ich erkläre ehrenwörtlich, dass ich die vorliegende wissenschaftliche Arbeit selbstständig angefertigt und die mit ihr unmittelbar verbundenen Tätigkeiten selbst erbracht habe. Ich erkläre weiters, dass ich keine anderen als die angegebenen Hilfsmittel benutzt habe. Alle aus gedruckten, ungedruckten oder dem Internet im Wortlaut oder im wesentlichen Inhalt übernommenen Formulierungen und Konzepte sind gemäß den Regeln für wissenschaftliche Arbeiten zitiert und durch Fußnoten bzw. durch andere genaue Quellenangaben gekennzeichnet.

Die während des Arbeitsvorganges gewährte Unterstützung einschließlich signifikanter Betreuungshinweise ist vollständig angegeben.

Die wissenschaftliche Arbeit ist noch keiner anderen Prüfungsbehörde vorgelegt worden. Diese Arbeit wurde in gedruckter und elektronischer Form abgegeben. Ich bestätige, dass der Inhalt der digitalen Version vollständig mit dem der gedruckten Version übereinstimmt.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

(Unterschrift) (Ort, Datum)

Abstract

Combinatorial optimization and semidefinite programming have been two very active research areas over the last decades. Combinatorial optimization uses heuristic, approximation and exact algorithms to find (near-)optimal solutions for many problems of practical interest whose feasible solutions are given by a finite set. Semidefinite programming builds the basis for some of the most advanced approximation results in computer science and is applied to practical problems in control theory, engineering and combinatorial optimization.

Ordering problems are a special class of combinatorial optimization problems, where weights are assigned to each ordering of n objects and the aim is to find an ordering of maximum weight. Even for the simplest case of a linear cost function, ordering problems are known to be NP-hard, i.e. it is extremely unlikely that there exists an efficient (polynomial-time) algorithm for solving ordering problems to optimality.

Ordering problems arise in a large number of applications in such diverse fields as economics, business studies, social choice theory, sociology, archaeology, mathematical psychology, very-large-scale integration and flexible manufacturing systems design, scheduling, graph drawing and computational biology.

In this thesis we use semidefinite optimization for solving ordering problems with up to 100 objects to provable optimality—despite their theoretical difficulty. We present a systematic investigation of semidefinite optimization based relaxations extending and improving existing exact approaches to ordering problems. We consider problems where the cost function is either linear or quadratic in the relative positions of pairs of objects. That includes well-established combinatorial optimization problems like the Linear Ordering Problem, the minimum Linear Arrangement Problem, the Single Row Facility Layout Problem, the weighted Betweenness Problem, the Quadratic Ordering Problem and Multi-level Crossing Minimization.

We provide a theoretical and practical comparison of existing exact approaches based on linear, quadratic or semidefinite relaxations. Up to now there existed quite diverse exact approaches to the various ordering problems. A main goal of this thesis is to highlight their connections and to present a unifying approach by showing that the proposed semidefinite model can be successfully applied to all kinds of ordering problems.

We accomplish a polyhedral study of the various ordering polytopes in small dimensions that helps us to evaluate and further improve the suggested semidefinite relaxations. We also deduce several theoretical results showcasing the polyhedral advantages of the semidefinite approach compared to Branch-and-Cut algorithms based on linear and quadratic relaxations. Additionally we introduce a new drawing paradigm for layered graphs requiring (near-)optimal solutions of an ordering problem with quadratic cost function called Multi-level Verticality Optimization. In this new drawing paradigm we are able to describe the structure of graphs more compactly and therefore obtain (well-)readable drawings of graphs too large for other available methods. We propose several heuristic and exact approaches to solve Multi-level Verticality Optimization problems and design a drawing algorithm to illustrate the (near-)optimal solutions.

For tackling ordering problems of challenging size, we construct an algorithm that uses a method from nonsmooth optimization to approximately solve the proposed semidefinite relaxations and applies a rounding scheme to the approximate solutions to obtain (near-)optimal orderings. We show the efficiency of our algorithm by providing extensive computational results for a large variety of problem classes, solving many instances that have been considered in the literature for years to optimality for the first time. While the algorithm provides improved bounds for several classes of difficult instances with a linear cost function, it is clearly the method of choice for instances with quadratic cost structure (except for some very sparse instances).

Zusammenfassung

Kombinatorische Optimierung und Semidefinite Programmierung waren sehr aktive Forschungsbereiche während der letzten 20 Jahre. In der Kombinatorischen Optimierung werden Heuristiken, Approximationsalgorithmen und exakte Algorithmen zum Auffinden von (beinahe) optimalen Lösungen für viele praktisch relevante Probleme, deren zulässige Lösungen durch eine endliche Menge gegeben sind, verwendet. Die Semidefinite Programmierung stellt die Basis für einige der fortschrittlichsten Approximationsresultate in der Theoretischen Informatik dar und besitzt eine Vielzahl von Anwendungen in der Kontrolltheorie, den technischen Wissenschaften und der Kombinatorischen Optimierung.

Ordnungsprobleme gehören zur Klasse der Kombinatorischen Optimierungsprobleme, wobei jeder Anordnung von n Objekten ein Gewicht zugeordnet wird und das Ziel im Auffinden der Anordnung mit maximalem Gewicht besteht. Sogar für den einfachsten Fall einer linearen Kostenfunktion sind Ordnungsprobleme NP-schwierig, d.h. es ist extrem unwahrscheinlich, dass ein effizienter (polynomieller) Algorithmus für das Auffinden der optimalen Lösung von Ordnungsproblemen existiert.

Ordnungsprobleme treten in einer Vielzahl von Anwendungen in solch unterschiedlichen Bereichen wie Volks- und Betriebswirtschaft, Sozialwerttheorie, Soziologie, Archäologie, mathematische Psychologie, VLSI- und FMS-Design, Scheduling, Graphenzeichnen und Bioinformatik auf.

In dieser Arbeit verwenden wir Semidefinite Optimierung für das Finden einer optimalen Lösung von Ordnungsproblemen mit bis zu 100 Objekten—trotz der theoretischen Schwierigkeiten. Wir präsentieren eine systematische Untersuchung von auf Semidefiniter Optimierung basierenden Relaxationen, welche die existierenden exakten Algorithmen für Ordnungsprobleme erweitern und verbessern. Wir betrachten Probleme mit linearen und quadratischen Kosten für die relative Anordnung von Paaren von Objekten. Dies umfasst insbesondere etablierte Kombinatorische Optimierungsprobleme wie das Lineare Ordnungsproblem, das minimale Lineare Anordnungsproblem, das einreihige Anlagenanordnungsproblem, das gewichtete Betweennessproblem, das quadratische Ordnungsproblem und das mehrstufige Kreuzungsminimierungsproblem.

Wir liefern einen theoretischen und praktischen Vergleich existierender exakter Algorithmen, welche auf linearen, quadratischen und semidefiniten Relaxationen basieren. Bis jetzt existierten sehr unterschiedliche exakte Methoden für die verschiedenen Ordnungsprobleme. Es ist ein Hauptziel dieser Arbeit, deren Zusammenhänge aufzuzeigen und einen vereinheitlichenden Ansatz zu präsentieren, indem wir nachweisen, dass das semidefinite Modell erfolgreich auf alle Typen von Ordnungsproblemen angewendet werden kann.

Wir führen eine polyedrische Studie einiger Ordnungspolytope mit kleiner Dimension durch. Dies hilft uns, die vorgeschlagenen Relaxationen zu evaluieren und zu verbessern. Wir leiten auch einige theoretische Resultate ab, welche die polyedrischen Vorteile der semidefiniten Methode im Vergleich zu auf linearen und quadratischen Relaxationen basierenden Branch-and-Cut Algorithmen aufzeigen. Zusätzlich führen wir ein neues Zeichenparadigma für geschichtete Graphen ein, welches (beinahe) optimale Lösungen eines Ordnungsproblems mit quadratischer Kostenfunktion, das den Namen mehrstufige Vertikalitätsoptimierung trägt, benötigt. In diesem neuen Zeichenparadigma ist es uns möglich die Struktur von Graphen kompakter zu beschreiben und dadurch (gut) lesbare Zeichnungen von Graphen zu erhalten, welche zu groß für die anderen verfügbaren Methoden sind. Wir schlagen einige Heuristiken und exakte Ansätze zur Lösung der mehrstufigen Vertikalitätsoptimierung vor und entwerfen einen Zeichenalgorithmus zur Illustration der (beinahe) optimalen Lösungen.

Um Ordnungsprobleme anspruchsvoller Größe zu lösen, konstruieren wir einen Algorithmus, welcher eine Methode der nicht-glatten Optimierung verwendet, um die vorgeschlagenen semidefiniten Relaxationen approximativ zu lösen. Außerdem wendet er ein Rundungschema auf die appoximativen Lösungen an, um (beinahe) optimale Anordnungen zu erhalten. Wir zeigen die Effizienz unseres Algorithmus,

indem wir umfassende Resultate für eine große Vielfalt von Problemklassen liefern. Dabei lösen wir viele Instanzen, die in der Literatur seit Jahren betrachtet wurden, zum ersten Mal optimal. Während der Algorithmus verbesserte Schranken für einige Klassen schwieriger Instanzen mit linearer Kostenfunktion liefert, ist er eindeutig die Methode der Wahl für Instanzen mit quadratischer Kostenstruktur (außer für einige sehr dünnbesetzte Instanzen).

Acknowledgements

I am grateful to a number of people who have supported me in the development of this thesis and it is my pleasure to highlight them here.

I want to thank my supervisor Franz Rendl for giving me the opportunity to do research in a wonderful scientific environment, for offering me the possibilities and freedom to pursue multiple research areas and most of all, for his enthusiasm about discussing mathematical issues and the large amount of time he devoted to my concerns. His ideas and advice led me into active research and substantiated my thesis.

My thanks go to all members of the Mathematics Department at the Alpen-Adria-Universität Klagenfurt for providing me excellent working conditions. I would like to thank especially our secretary Anita Wachter for taking a lot of organizational stuff from my shoulders, our chair Winfried Müller for his unconfined support for young researchers, my colleague Angelika Wiegele for always supporting me regarding research and teaching and Albrecht Gebhard for helping me many times in the most patient and friendly way with all sorts of computer problems.

I also want to thank the co-authors of my papers for sharing their experience and knowledge, in particular Markus Chimani, for his enthusiasm for our common research. I would like to thank Petra Mutzel and Michael Jünger for introducing me to graph drawing, for inviting me to Vienna and Köln and for their open-hearted hospitality. I want thank Gerhard Reinelt for many fruitful discussions and practical hints regarding the polyhedral approach to the Linear Ordering Problem, for inviting me two times to Heidelberg and for his open-hearted hospitality. I want to thank Miguel Anjos for inviting me to Montreal, for many fruitful discussions regarding layout problems and polynomial programming and for his open-hearted hospitality.

I am grateful to Marcus Oswald for introducing me to the Betweenness Problem, the Target Visitation Problem and PORTA and for many joint working days and nights. I would like to thank Thorsten Bonato, Achim Hildenbrandt and Marcus Oswald for making my stays in Heidelberg not only fruitful but also very enjoyable. I want to thank Andreas Schmutzer for making my stays in Köln very enjoyable.

I want to thank my friends Davide, Markus and Michael for sharing a part of their lives with me and for being there whenever I need help.

Above all, my thanks go to my parents and my sister, for supporting me in all conceivable ways from when I was young, up to now.

Notation

This is a short description of the symbols used throughout this thesis. We also give the abbreviations of the discussed combinatorial optimization problems.

$\mathbb{R}^+$	space of real positive numbers
$\mathbb{R}^n$	space of real n-dimensional vectors
$\mathcal{S}_\setminus$	space of $n \times n$ symmetric matrices
$\mathcal{S}_n^+$	space of $n \times n$ positive semidefinite matrices
$\mathcal{S}_n^{++}$	space of $n \times n$ positive definite matrices
$A \succ (\succcurlyeq) B$	Löwner partial order, positive (semi)definiteness of matrix $A - B$
min	minimum, minimize
max	maximum, maximize
inf	infimum
sup	supremum
$\mathbf{Tr}(A)$	trace of matrix A
$\langle A, B \rangle$	$\langle A, B \rangle := \mathbf{Tr}(A)$
I	identity matrix of appropriate dimension
e	vector of all ones of appropriate dimension
$\mathrm{diag}(A)$	vector formed from the main diagonal of matrix A
$\mathcal{E}$	elliptope
$\mathcal{N}$	set of n objects
$\phi \in \Pi$	set of permutations
$G(V, E)$	graph G with vertex set V and edge set E
$\overline{G}$	complement graph of graph G
$\lvert V \rvert$	number of elements contained in the vertex set V
$\omega(G)$	clique number of graph G
$\chi(G)$	chromatic number of graph G
$\mathfrak{d}(e)$	verticality of a straight-line edge e
ω'	width of the widest level of a (proper) level graph
$\dot{<}$	fixed total order on V (e.g. based on indices)
LED	long-edge dummy node
PD	positional dummy node
$\mathcal{P}_C$	cut polytope
$\mathcal{P}_{LOP}$	linear ordering polytope
$\mathcal{P}_{BTW}$	betweenness polytope
$\mathcal{P}_{TRI}$	triple polytope
$\mathcal{P}_{QO}$	quadratic ordering polytope
$\mathcal{P}_{LQO}$	linear-quadratic ordering polytope
$\mathcal{P}_{MQO}$	multi-level quadratic ordering polytope
$\mathcal{P}_{CR}$	crossing polytope
`(MC)`	Max-Cut Problem
`(LOP)`	Linear Ordering Problem
`(minLA)`	minimum Linear Arrangement Problem
`(SRFLP)`	Single-row Facility Layout Problem

(PMP)	Physical Mapping Problem with End Probes
(COP)	Consecutive Ones Problem
(wBP)	weighted Betweenness Problem
(QOP)	Quadratic Ordering Problem
(BCM)	Bipartite Crossing Minimization
(MLCM)	Multi-level Crossing Minimization
(TCM)	Tanglegram Crossing Minimization
(MLVO)	Multi-level Verticality Optimization
(MLP)	Multi-level Planarization
(MQOP)	Multi-level Quadratic Ordering Problem

Contents

Abstract i
Acknowledgements v
Notation vii

I Introduction **1**

1 Introduction **3**

2 Preliminaries: Semidefinite Programming **7**
2.1 Introduction 7
2.2 Basic Theoretical Properties 7
2.3 Applications 9

3 Preliminaries: On Solving Semidefinite Programs **11**
3.1 Introduction 11
3.2 Interior-Point Methods 12
3.3 A Dynamic Version of the Bundle Method 13

II Theory & Algorithms **17**

4 The Linear Ordering Problem **19**
4.1 Introduction 19
4.2 Exact Approaches Based on Linear Programming 20
4.3 Exact Approaches Based on Semidefinite Programming 21

5 The Minimum Linear Arrangement Problem **27**
5.1 Introduction 27
5.2 Exact Approaches Based on Linear Programming 27
5.3 Exact Approaches Based on Semidefinite Programming 29

6 The Single Row Facility Layout Problem **31**
6.1 Introduction 31
6.2 Exact Approaches Based on Linear Programming 32
6.3 Exact Approaches Based on Semidefinite Programming 33

7 The Quadratic Ordering Problem **37**
7.1 Introduction 37
7.2 Ordering Polytopes in Small Dimensions 38
7.3 Heuristic Constraint Selection 40

8 Multi-level Crossing Minimization **45**
8.1 Introduction . . . 45
8.2 Exact Approaches Based on Linear Programming . . . 46
8.3 Exact Approaches Based on Semidefinite Programming . . . 47
8.4 Some Polyhedral Results . . . 49

9 Multi-level Verticality Optimization **55**
9.1 Introduction . . . 55
9.2 Verticality and Proper Drawings . . . 56
9.3 Non-Proper Drawing Scheme . . . 57
9.4 Basic Heuristic Approaches . . . 60
9.5 Exact Approaches Based on Quadratic Programming . . . 61
9.6 Exact Approaches Based on Linear Programming . . . 64
9.7 Exact Approaches Based on Semidefinite Programming . . . 64
9.8 Some Polyhedral Results . . . 65
9.9 Extensions . . . 67
9.10 Some Complexity Results . . . 69
9.11 Applications Beyond Graph Drawing . . . 70

III Experiments & Outlook **73**

10 The Linear Ordering Problem **75**
10.1 Small Facets . . . 75
10.2 Medium and Large Instances . . . 75
10.3 Speeding up the Linear Relaxation . . . 79

11 The Minimum Linear Arrangement Problem **81**
11.1 The Cartesian Cube . . . 81
11.2 Medium and Large Instances . . . 82

12 The Single Row Facility Layout Problem **85**
12.1 Comparison of Globally Optimal Methods for Small and Medium Instances . . . 85
12.2 Heuristics Based on Semidefinite Optimization . . . 86
12.3 Comparison of Globally Optimal Methods for Large Instances . . . 88

13 The General Quadratic Ordering Problem **93**
13.1 Small Facets . . . 93
13.2 Heuristic Constraint Selection . . . 94

14 Multi-level Crossing Minimization **97**
14.1 Introduction . . . 97
14.2 Graphs with Varying Densities . . . 98
14.3 Real-World Graphs . . . 100
14.4 Polytopes and Further Instances From Literature . . . 103
14.5 Quality of Bounds . . . 103
14.6 On Finding all Optimal Solutions . . . 107
14.7 Practical Comparison of Semidefinite Relaxations . . . 108

15 Multi-level Verticality Optimization **111**
15.1 Introduction . . . 111
15.2 Experimental Comparison of Exact Approaches . . . 112
15.3 Polytopes and Further Instances From Literature . . . 112
15.4 Real-World Graphs . . . 114
15.5 A Comparison with Multi-level Crossing Minimization . . . 116
15.6 Visual Results for Different Drawing Strategies . . . 116

16 Conclusion and Outlook **121**

Bibliography **123**

Index **135**

Part I

Introduction

Chapter 1

Introduction

Part I is a general introduction to the thesis' topic and summarizes the necessary preliminaries. In order to make the thesis self-contained, we discuss the basic theoretical properties and several important applications of semidefinite programs in Chapter 2. In Chapter 3 we sketch the most important methods for solving semidefinite programs, namely interior-point methods and Bundle methods. We discuss in some more detail a dynamic version of the bundle method that we use in Part III to approximately solve computationally challenging semidefinite programs.

In Part II we summarize the main features of existing exact approaches for all kinds of ordering problems and compare them theoretically to our newly introduced semidefinite relaxations. In Chapter 4 we show that the Linear Ordering Problem can be formulated as a semidefinite program with integrality conditions on some variables. Omitting the integrality conditions yields a basic semidefinite relaxation that can be further strengthened by adding some classes of tightening constraints. Finally we prove that our deduced semidefinite relaxations are stronger than the standard linear relaxation and contain nearly all facet classes of the linear ordering polytopes in small dimensions. In Chapters 5 and 6 we show that the deduced semidefinite relaxations can also be applied to the minimum Linear Arrangement Problem and the Single Row Facility Layout Problem, respectively, by appropriate adaption of the cost function. In Chapter 6 we also relate another semidefinite model for the Single Row Facility Layout Problem that has been suggested in the literature to our approach, showing that the latter is theoretically stronger than the former. Furthermore we point out several connections of the betweenness and the quadratic ordering polytope. In Chapter 7 we demonstrate that the former problems can be seen as special cases of the Quadratic Ordering Problem and mention some further ordering problems with this characteristic. We argue that the more complex the cost structure of the ordering problem the better semidefinite approaches perform compared to methods based on linear programming. We illustrate the need of further improving the tightness of the semidefinite relaxations for some very difficult types of ordering problems and present two strategies to approach this goal. The first one uses the analysis of the complete outer description of different ordering polytopes in small dimensions to identify improving constraints for the semidefinite relaxations, whereas the second one heuristically selects the most important constraints from a class of inequalities that is too large to be considered as a whole. In Chapter 8 we model Multi-level Crossing Minimization as a semidefinite program over the multi-level quadratic ordering polytope. For Multi-level Crossing Minimization we are given a layered graph and ask for an ordering of the nodes on their levels such that, when drawing the graph with straight lines, the resulting number of crossings is minimized. It is an important combinatorial optimization problem in the area of graph drawing and can be considered as an ordering problem with quadratic cost structure. We show the polyhedral advantages of the semidefinite approach over the linear programming based approaches for this problem. In doing so, we implicitly evaluate and justify the choice of our semidefinite relaxation. We close the theoretical part of the thesis by introducing the problem of ordering vertices of a layered graph such that the edges of the graph are

drawn as vertical as possible in Chapter 9. This Multi-level Vertical Optimization Problem also falls into the class of ordering problems with quadratic cost structure. It is conceptually related to the well-studied problem of multi-level crossing minimization, but offers certain interesting novel properties: we not only have to consider the pure relative ordering of the nodes, but their final absolute positions within the ordered levels. Furthermore, Multi-level Vertical Optimization is a genuine Multi-level Quadratic Ordering Problem and does not only consist of multiple sequentially linked bilevel problems like Multi-level Crossing Minimization. These properties allow us to describe the structure of graphs more compactly and therefore obtain (near-)optimal, (well-)readable drawings of graphs too large for other approaches. We present a motivation, several heuristic and exact methods and sketch further applications in scheduling and ranking problems, where Multi-level Verticality Optimization occurs apart from graph drawing. We compare the proposed linear, quadratic and semidefinite models from the polyhedral point of view, again showing the strong theoretical properties of the applied semidefinite relaxations. Furthermore, we present a new drawing scheme especially suitable for verticality optimization. It works without the traditional subdivision of edges, i.e., edges may span multiple levels, and therefore potentially allows to tackle larger graphs.

In Part III we tailor the dynamic version of the bundle method described in Section 3.3 to approximately solve the semidefinite relaxations for the various ordering problems proposed in Part II. To obtain (near-)optimal orderings we design a semidefinite rounding heuristic that exploits the information generated by the bundle method. We conduct a large number of experiments, both on randomized and on real-world instances, comparing our approach to state-of-the-art exact algorithms for the respective problem types. In Chapter 10 we compare the practical strength of several semidefinite relaxations and apply our algorithm to various difficult benchmark instances with linear cost structure, yielding improved bounds compared to Branch-and-Cut methods based on linear programming. Additionally we design a new algorithm that considerably speeds up the solution of the basic linear programming relaxation for very large Linear Ordering Problems and showcase its practical performance on selected instances. In Chapter 11 we demonstrate that our algorithm is capable of solving many minimum Linear Arrangement instances to optimality that remained unsolved for the last ten years. However, if the instances are very large, there exist (like for the Linear Ordering Problem) other exact methods that outperform our algorithm. In contrast our algorithm is clearly the method of choice for the Single Row Facility Layout Problem. On the one hand, our approach provides optimal solutions for larger instances than any other exact method available and on the other hand it yields the strongest lower and upper bounds for large-scale instances. We summarize the extensive numerical experiments on all available benchmark instances from the literature in Chapter 12, in fact demonstrating that our exact approach is the best one in terms of running time and bound quality for all medium and large instances. In Chapter 13 we present preliminary computational results for various tightening strategies for semidefinite relaxations. In particular applying the heuristic constraint selection to further tighten the standard semidefinite relaxation of the Max-Cut problem, another well-known combinatorial optimization problem, yields convincing improvements. For Multi-level Crossing Minimization we compare our algorithm to a newly written Branch-and-Cut method building on linear relaxations on a large variety of synthetic and real-world instances. While the semidefinite approach clearly dominates for denser graphs, the linear approach is usually faster for sparse instances. However, even for such sparse graphs, our algorithm solves more instances to optimality than the Branch-and-Cut method. In fact, there is no single instance the linear approach solved, which we did not. We summarize the numerical results in Chapter 14, where we also demonstrate why our algorithm is more appropriate to provide strong lower and upper bounds for instances that cannot be solved to optimality by both methods. In Chapter 15 we experimentally show that for Multi-level Verticality Optimization exact approaches based on linear and quadratic relaxations are inapplicable, even for small sparse graphs, while our semidefinite approach works surprisingly well in practice. We apply our algorithm and several heuristics to a large collection of synthetic and real-world graphs, demonstrating the practical benefits of our proposed new drawing scheme. We also computationally and visually compare Multi-level Verticality

Optimization to the closely related Multi-level Crossing Minimization, showcasing the relative merits of the different optimization goals. Finally in Chapter 16 we give a conclusion and point out several research questions and plans.

The content of this thesis is largely based on the following original papers

- "Semidefinite Relaxations of Ordering Problems" [115] (Chapters 4, 5, 10 and 11),
- "A Computational Study for the Single-Row Facility Layout Problem" [116] (Chapters 6 and 12),
- "An SDP Approach to Multi-level Crossing Minimization" [44, 45] (Chapters 8 and 14),
- "Exact Approaches to Multi-level Vertical Orderings" [42] (Sections 9.1, 9.2, 9.5–9.8, 9.11, 15.2, 15.3, 15.5),
- "Multi-level Verticality Optimization: Concept, Strategies, and Drawing Scheme" [43] (Sections 9.1–9.4, 9.9, 9.10 and 15.4–15.6).

Furthermore, this thesis contains yet unpublished results regarding

- some new strategies for tightening semidefinite relaxations and the according computational experiments (Chapters 7 and 13),
- the speeding up of the solution of large-scale linear programs with a particular high number of constraints (Section 10.3).

Chapter 2

Preliminaries: Semidefinite Programming

2.1 Introduction

Semidefinite programming (SDP) is an extension of linear programming (LP), with vector variables replaced by matrix variables and nonnegativity elementwise replaced by positive semidefiniteness. Thus a (primal) SDP can be expressed as the following optimization problem

$$\begin{aligned} p^* &:= \inf_X \{ \langle C, X\rangle \ : \ X \in \mathcal{P}\}, \\ \mathcal{P} &:= \left\{ \ X \ \middle| \ \langle A_i, X\rangle = b_i, \ i \in \{1,\ldots,m\}, \ X \in \mathcal{S}_n^+ \ \right\}, \end{aligned} \tag{P}$$

where the data matrices $A_i,\ i \in \{1,\ldots,m\},\ C$ and the variable matrix X are in $\mathcal{S}_n$, the space of symmetric $n \times n$ matrices. The essential difference between LP and SDP is that the nonnegative orthant $\mathbb{R}^n_+$ is replaced by the convex, self-dual cone of positive semidefinite matrices $\mathcal{S}_n^+$, which has a nonlinear boundary. Thus (P) is a nonlinear convex programming problem. If the matrix X is restricted to be diagonal, (P) reduces to a linear program. In the following two sections we give a short overview of the basic theoretical properties and main application areas of SDP. We refer the reader to the handbooks [11, 207] for a thorough coverage of the theory, algorithms and software in this area, as well as a discussion of many application areas where semidefinite programming has had a major impact.

2.2 Basic Theoretical Properties

SDP is a relatively new area of optimization as most papers on SDP were written since 1990. The roots of SDP can be traced back to the sixties, when Bellman and Fan [20] derived the first duality theorem. They recognized that much of the duality theory for LP can be extended to SDP by using slightly stronger assumptions. In the following we briefly summarize the basic theoretical properties of semidefinite programs, a comprehensive discussion of this topic can be found for instance in Nesterov and Nemirovskii [165] or De Klerk [52]. Let the Lagrangian dual problem of (P) be given by

$$\begin{aligned} d^* &:= \sup_{y,S} \ \left\{ \ b^\top y \ : \ (y,S) \in \mathcal{D} \right\}, \\ \mathcal{D} &:= \left\{ \ (y,S) \ \middle| \ \sum_{i=1}^m y_i A_i + S = C, \ S \in \mathcal{S}_n^+, \ y \in \mathbb{R}^m \ \right\}. \end{aligned} \tag{D}$$

Then weak duality $\langle C, X\rangle \geq b^\top y$ holds for all $X \in \mathcal{P}$ and $(y, S) \in \mathcal{D}$ due to the Minimax inequality

$$\inf_{x\in S_1} \sup_{y\in S_2} f(x, y) \geq \sup_{y\in S_2} \inf_{x\in S_1} f(x, y), \tag{2.1}$$

that is valid for any function $f : S_1 \times S_2 \to \mathbb{R}$ where S_1 and S_2 are arbitrary sets. Contrary to linear programming, strong duality ($p^* = d^*$) does not hold for SDP in general (see Vandenberghe and Boyd [205] for a standard counterexample). To guarantee strong duality, we additionally have to ask for strict feasibility of (P) (or (D)), i.e. we have to ask for the existence of a $X \in \mathcal{P} \cap \mathcal{S}_n^{++}$ ($S \in \mathcal{D} \cap \mathcal{S}_n^{++}$).

Theorem 2.1 *Assume that $d^* < \infty$ (resp. $p^* > -\infty$). Further assume that (D) (resp. (P)) is strictly feasible. Then $p^* = d^*$ and this value is attained for (P) (resp. (D)).*

A proof of this theorem can be found for instance in Duffin [67], Rockafellar [185], Nesterov and Nemirovskii [165] or De Klerk [52] An example where the primal optimal solution is not attained, is e.g. provided by Helmberg [99]. If strong duality holds, it is easy to deduce the following necessary and sufficient optimality conditions

$$X \in \mathcal{P},\ (y, S) \in \mathcal{D},\ XS = 0, \tag{2.2}$$

where we again want to point out a difference compared to linear programming. For SDP strict complementarity , i.e. the existence of an optimal solution (X^*, S^*, y) such that $X^* + S^* \in \mathcal{S}_n^{++}$, does not hold in general (see e.g. the counterexample given by Alizadeh et al. [4]). Except linear programming, SDP has some further interesting special cases. To deduce them, we use the well-known Schur complement theorem [26, Appendix A.5.5].

Theorem 2.2 *Let $M = \begin{pmatrix} A & B \\ B^\top & C \end{pmatrix}$ with $A \in \mathcal{S}_n^{++}$ and $C \in \mathcal{S}_n$ be given. Then the Schur complement of A in M is given by $C - B^\top A^{-1} B$ and it holds $M \in \mathcal{S}_n^+$, iff $C - B^\top A^{-1} B \in \mathcal{S}_n^+$.*

Proof. M can be transformed to a block diagonal matrix, using the following similarity transformation

$$\begin{pmatrix} I & 0 \\ -A^{-1}B & I \end{pmatrix} \begin{pmatrix} A & B \\ B^\top & C \end{pmatrix} \begin{pmatrix} I & -A^{-1}B \\ 0 & I \end{pmatrix} = \begin{pmatrix} A & 0 \\ 0 & C - B^\top A^{-1} B \end{pmatrix}.$$

Since a block diagonal matrix is positive semidefinite, iff its diagonal blocks are positive semidefinite, the proof is complete. □

Applying the Schur complement theorem we can rewrite the quadratic constraint

$$(Ax + b)^\top (Ax + b) - (c^\top x + d) \leq 0,\ x \in \mathbb{R}^n,$$

and the second order cone constraint

$$\left\{ (t, x) \;\middle|\; t \geq \sqrt{\sum_{i=1}^n x_i^2} \right\},$$

as semidefinite constraints

$$\begin{pmatrix} I & Ax + b \\ (Ax + b)^\top & c^\top x + d \end{pmatrix} \succcurlyeq 0,\ x \in \mathbb{R}^n,$$

$$\begin{pmatrix} tI & x \\ x^\top & t \end{pmatrix} \succcurlyeq 0.$$

Thus several optimization problems with quadratic or cone constraints, including the well-known convex quadratic programming (QP) problem , are special cases of SDP. Another interesting special case is the nonlinear problem

$$\min_x \left\{ \frac{(c^\top x)^2}{d^\top x} \;\middle|\; Ax \geq b \right\},$$

where we additionally assume that $d^\top x > 0$ if $Ax \geq b$. Also for this problem we can state an equivalent SDP problem

$$\min_{t,x} \left\{ t \;\middle|\; \begin{pmatrix} t & c^\top x & 0 \\ c^\top x & d^\top x & 0 \\ 0 & 0 & \mathrm{Diag}(Ax-b) \end{pmatrix} \succcurlyeq 0 \right\}.$$

Also the classical problem of finding the largest eigenvalue $\lambda_{\max}(A)$ of a symmetric matrix A can be formulated as an SDP problem

$$\min_t \{ t \mid tI - A \succcurlyeq 0,\ t \in \mathbb{R} \}.$$

A list of further eigenvalue or matrix norm minimization problems that can be stated as SDP's is provided by Vandenberghe and Boyd [205].

2.3 Applications

There exist important applications of SDP in approximation theory (e.g. non-convex quadratic optimization [164] and nonnegative polynomials [171])), system and control theory [17, 27], and mechanical and electrical engineering (VLSI transistor sizing and pattern recognition [205] and structural design [22]). But in this thesis we are most interested in applications of SDP in combinatorial optimization. An instance of a combinatorial optimization problem is given by a pair (L, f), where L is a countable set of all feasible solutions and f is a function $f : L \to \mathbb{R}$ that assigns an objective value to each element of L. Now the aim is to find an element $i \in L$ with minimal ($f(i) \leq f(u), \forall u \in L$) or maximal ($f(i) \geq f(u), \forall u \in L$) objective value. Thus ordering problems fall into the area of combinatorial optimization. In the following we give a short review of two other important applications of SDP in combinatorial optimization.

The probably most celebrated application is the Lovász θ-function [147] , that maps an undirected graph $G = (V, E)$ to $\mathbb{R}^+$ and is given as the optimal value of the following SDP problem

$$\theta(\overline{G}) := \max_X \left\{ \langle ee^\top, X \rangle : x_{ij} = 0, (i,j) \notin E, \mathbf{Tr}(X) = 1, X \in \mathcal{S}_n^+ \right\}, \tag{2.3}$$

where $e \in \mathbb{R}^{|V|}$ denotes the vector of all-ones, and $\overline{G}$ the complement graph of G, i.e. $\overline{G} = (V, K \setminus E)$, where K consists of all 2-element subsets of V. The θ-function fulfills the following relation known as "sandwich theorem"

$$\omega(G) \leq \theta(\overline{G}) \leq \chi(G),$$

where $\omega(G)$ denotes the clique number of G, and $\chi(G)$ the chromatic number. The sandwich theorem gives a polynomial-time approximation to both $\omega(G)$ and $\chi(G)$ that cannot be off by more than a factor $|V|$. In-approximability results by Håstad [94] and Feige and Kilian [71] show that neither $\omega(G)$ nor $\chi(G)$ can be approximated within a factor $|V|^{1-\varepsilon}$ for any $\varepsilon > 0$. Thus the sandwich theorem yields a very strong approximation guarantee.

Another famous application of SDP to combinatorial optimization is the NP-hard (see Karp [124]) Max-Cut Problem `(MC)` . Let $G = (V, E)$ be an undirected graph with edge weights $w_{ij} \geq 0$ $(i \neq j)$. Then `(MC)` consists in finding a partition (S, T) of V with $T = V \setminus S$ such that the weight of the edges in the

S-T-cut $\sum_{i\in S, j\in T} w_{ij}$ is maximized. To deduce an SDP relaxation of (MC), we rewrite it as a Boolean quadratic optimization problem by introducing the bivalent variables

$$y_i = \begin{cases} 1 & \text{if vertex } i \in S, \\ -1 & \text{if vertex } i \in T, \end{cases} \quad \forall i \in V.$$

Thus for a given edge $(i, j) \in E$ we have

$$y_i y_j = \begin{cases} -1 & (i, j) \text{ lies in the } S\text{-}T\text{-cut}, \\ 1 & \text{otherwise.} \end{cases}$$

The weight of the maximum cut is therefore given by

$$\max_{y\in\{-1,1\}^{|V|}} \left\{ \frac{1}{2} \sum_{i<j} w_{ij}(1 - y_i y_j) \right\} = \max_{y\in\{-1,1\}^{|V|}} \frac{1}{4} y^\top L y, \tag{MC}$$

where $L = -W + \mathrm{Diag}(We)$ and W is the matrix with zero diagonal and the (nonnegative) edge weights as off-diagonal entries. Now we use the matrix $Y := yy^\top$ to rewrite (MC)

$$\max \left\{ \frac{1}{4} \langle L, Y \rangle : \mathrm{diag}(Y) = e, Y \succcurlyeq 0, \mathrm{rank}(Y) = 1 \right\}. \tag{MC}$$

Dropping the rank one condition on Y yields the SDP relaxation

$$\max \left\{ \frac{1}{4} \langle L, Y \rangle : \mathrm{diag}(Y) = e, Y \succcurlyeq 0 \right\}. \tag{MC$_1$}$$

In their celebrated paper [86] Goemans and Williamson devised a randomized rounding scheme that uses (MC$_1$) to generate cuts in the graph. They can prove that one of these cuts gives a 0.878...-polynomial-time-approximation of (MC). Håstad [95] two years later showed that it is NP-complete to approximate (MC) within a factor $\frac{16}{17}$.

Rendl et al. [184] approximately solve (MC$_1$), strengthened by triangle inequalities (for a definition see equation (4.12)), with the help of a dynamic version of the bundle method (for details see Section 3.3) and use the obtained upper bounds in a Branch-and-Bound setting for finding exact solutions of (MC). Their approach nearly always outperforms all other approaches for (MC) and works particular well for dense graphs, where linear programming-based methods fail. In this thesis we apply a similar algorithmic approach to tackle (quadratic) ordering problems. As the quadratic ordering polytope is a face of the cut polytope, our method solves (MC) as a special case (if we leave out some constraint classes).

For more details on the θ-function and the Max-Cut Problem and for further applications of SDP to combinatorial optimization see the survey articles [85, 139, 183] and the book of De Klerk [52, Part II].

Chapter 3

Preliminaries: On Solving Semidefinite Programs

3.1 Introduction

In the previous chapter we have mentioned several areas of application for SDP. This of course motivates the research for efficient methods to solve SDP. Bearing the connections between LP and SDP in mind, it is not surprising that interior-point methods (IPMs) have been successfully extended from LP to SDP. They are for sure the most common and the most elegant way for efficiently solving SDPs. As for LP, there exist different variants of (IPMs) (e.g. primal and dual logarithmic barrier methods, affine-scaling methods, potential reduction methods) that have different strengths dependent on the structure of the SDP (for a survey on (IPMs) see e.g. the books of Wright [208] and De Klerk [52]). (IPMs) have polynomial worst-case iteration bounds for the computation of ϵ-optimal solutions, i.e. feasible (X, S) with duality gap $\langle S, X\rangle \leq \epsilon$ for a given tolerance $\epsilon > 0$ (for a more precise statement of the complexity results for (IPMs) see the review by Ramana and Pardalos [179]). Although the theoretical analysis of (IPMs) for LP and SDP is quite similar, there exist major differences concerning implementation and practical performance. In particular exploiting sparsity of the data matrices becomes very difficult for (IPMs) applied to SDP and there is still a lot of current research on this topic (see e.g. the survey articles of Fujisawa et al. [76] and of Nemirovski and Todd [163] or the recent research papers [9, 145]). Thus, in general, state-of-the-art (IPMs) are limited to SDPs involving matrices of dimension n=1000 and having a few thousand constraints.

Semidefinite relaxations that give good approximations for combinatorial optimization problems typically have a very large number of constraints m (e.g. $\Theta(n^2)$ over even $\Theta(n^3)$) and therefore motivated the research on new methods for solving SDP. The Boundary Point Method [150, 177], using quadratic regularization of SDP problems, was successfully applied to compute the θ-function (for details see Section 2.3) for large graphs. Also several first-order methods that use only gradient information have been developed lately. Burer and Monteiro [32] use their projected gradient algorithm for solving a nonconvex, nonlinear programming reformulation of the basic semidefinite relaxation (MC_1) for Max-Cut. Davi, Jarre and Rendl [50, 51, 118] developed a hybrid approach that first uses a first-order method (APD-Method) to generate an approximate solution and then switches to a Krylov subspace algorithm (QMR method) to improve this approximation. They successfully apply their approach for computing the θ-function and the doubly nonnegative relaxation of the Max-Stable-Set-Problem for large graphs.

Finally the bundle method can be used for solving SDPs if the matrix dimension n is not too far beyond 1000. The number of constraints m can be significantly larger (even $\Theta(n^3)$). The bundle method is used for nonsmooth optimization and was introduced in the 1970's by Lemaréchal [140, 141]. Helmberg and Oustry [103] survey its applications to eigenvalue optimization and related problems. Helmberg and

Rendl [100, 104] use the spectral bundle method to tackle several combinatorial optimization problems (Max-Cut, θ-function, bisection, frequency assignment problems) and provide a detailed comparison of their approach to other methods available. Fischer et al. [73] describe a dynamic version of the bundle method, where they maintain a basic set of constraints explicitly. They provide strong SDP-based bounds for dense instances of the Max-Cut and Equipartition Problem, which cannot be achieved with any of the other methods mentioned above.

In the following two sections we recall the basic properties and algorithmic machinery of primal-dual path-following `(IPMs)` and the dynamic version of the bundle method. We will use the dynamic version of the bundle method (that applies a primal-dual path-following method for function evaluation) for the practical solution of semidefinite relaxations of different ordering problems in Part III.

3.2 Interior-Point Methods

Nesterov and Nemirovski [165] provided the theoretical background for solving SDPs with `(IPMs)` by studying linear optimization problems over closed convex cones. They showed that theses problems can be solved in polynomial time by sequential minimization techniques, where the conic constraint is discarded and a suitable, self-concordant barrier term is added to the objective. Self-concordant barriers go to infinity as the boundary of the cone is approached and can be minimized efficiently by Newton's method, as they are smooth convex functions with Lipschitz continuous second derivatives. A computable self-concordant barrier for the cone of semidefinite matrices is given by $f_{\text{bar}}(X) = -\log\ \det(X)$. Practical experience indicates that primal-dual path-following methods are best suited for our purposes (the optimization of a linear function over the elliptope (4.9)). These methods minimize the duality gap $\langle C, X\rangle - b^\top y = \langle X, S\rangle$ and use a combined primal-dual barrier function $-\log\ \det(XS)$. We assume that strong duality holds and perturb the necessary and sufficient optimality conditions (2.2) to get the following system of equations

$$X \in \mathcal{P},\ (y, S) \in \mathcal{D},\ XS = \mu I, \tag{3.1}$$

where $\mu \in \mathbb{R}^+$. Clearly, $X \succ 0$ and $S \succ 0$ must be satisfied for solutions of (3.1), as $XS = \mu I$ forces X and S to be nonsingular. In fact, (3.1) has a unique solution (X_μ, S_μ, y_μ), iff `(D)` and `(P)` are both strictly feasible. Furthermore (X_μ, S_μ, y_μ) form an analytic curve (the central path), parametrized by μ. This can be shown by straightforward application of the implicit function theorem (for proofs of the two basic results mentioned above see e.g. [52, Chapter 3]).

Primal-dual path-following methods use (3.1) to obtain search directions $(\Delta X, \Delta S, \Delta y)$ that approximately satisfy the partly nonlinear, overdetermined ($m + 1 + n^2 + \binom{n+1}{2}$ equations, $m + 2\binom{n+1}{2}$ variables) system

$$\begin{aligned} X + \Delta X \succcurlyeq 0,\ S + \Delta S \succcurlyeq 0,& \\ \langle A_i, \Delta X\rangle = 0, \quad i = 1, \ldots, m,& \\ \sum_{i=1}^{m} \Delta y_i A_i + \Delta S = 0,& \\ (X + \Delta X)(S + \Delta S) = \mu I,& \\ \Delta X = \Delta X^\top.& \end{aligned} \tag{3.2}$$

The probably most straightforward way to determine approximate solutions of system (3.2) is linearizing its nonlinear equation and then determining a least squares solution (with the Gauss-Newton method), for details see Kruk et al. [133] and De Klerk et al. [53]. The most popular approach (proposed by Zhang [210]) to get approximate solutions of system (3.2) is to drop $\Delta X = \Delta X^\top$ and replace the nonlinear equation by

$$H_P(\Delta X S + X \Delta S) = \mu I - H_P(XS), \tag{3.3}$$

where H_P is defined by

$$H_P(M) := \frac{1}{2}\left[PMP^{-1} + P^{-\top}M^\top P^\top\right],$$

for any matrix M, where P is an arbitrary, nonsingular matrix. This gives now a square linear system that has a unique solution for several choices of P (for details see [193]). Furthermore, if this system has a solution, then ΔX is symmetric. The most common choices for the scaling matrix P, that of course ensure existence and uniqueness of each of the resulting search directions, are the following

- $P = S^{\frac{1}{2}}$ examined by Monteiro [159], Helmberg et al. [105] and Kojima et al. [131],
- $P = X^{-\frac{1}{2}}$ examined by Monteiro [159] and Kojima et al. [131],
- $P = \left[X^{\frac{1}{2}}\left(X^{\frac{1}{2}}SX^{\frac{1}{2}}\right)^{-\frac{1}{2}}X^{\frac{1}{2}}\right]^{\frac{1}{2}}$ examined by Nesterov and Todd [166].

We refer to Todd [202] for an investigation of the theoretical properties of about 20 different search-directions (including the ones mentioned above) used in primal-dual interior-point methods. There exist many variants of primal-dual path-following methods that essentially differ in how μ is update (reduced), and how system (3.2) is symmetrized and solved. A very popular and practically efficient variant that is used in popular SDP software, like Sedumi by Sturm [199], SDPT3 by Toh et al. [33, 204] and CSDP by Borchers [24, 146], is the predictor-corrector method possessing superlinear convergence properties [176]. A practical comparison of many SDP-solvers on several different data is provided by Mittelmann [158] on his website.

3.3 A Dynamic Version of the Bundle Method

Most combinatorial optimization problems can be formulated as linear or quadratic or semidefinite programs in binary variables. Tractable relaxations are obtained by replacing the integrality conditions with bounds on the variables. Then a polyhedral approach is used to get tight relaxations. If the number of cutting planes in the partial description of the convex hull of integer solutions gets "large"[1], this poses a serious challenge even to state of the art software. In theses cases it can be helpful to work with the Lagrangian dual to handle the cutting planes only indirectly. As the Lagrangian dual functional is nonsmooth, a method for nonsmooth optimization has to be applied to it. These iterative algorithms use function and subgradient evaluations of the dual functional to determine a sequence of trial points. When solving semidefinite relaxations of combinatorial optimization problems, we want to maintain some constraints (e.g. semidefiniteness) explicitly, resulting in nontrivial function evaluations. Thus we are interested in algorithms like the bundle method that work well even with a low (less than one hundred) number of function evaluations.

In the following we describe a dynamic version of the bundle method (for further details see [73]). On the one hand we are going to use this approach to approximately solve the semidefinite relaxations of all the ordering problems discussed in our computational Part III. On the other hand in Section 10.3 we show that this approach is also a valuable tool to considerably speed up the solution of linear programs with a very large number of constraints.

Let us consider an SDP of the form

$$\begin{aligned} p^* &:= \max_X \{\ \langle C, X\rangle\ :\ X \in \mathcal{S} \cap \mathcal{T}\}, \\ \mathcal{S} &:= \left\{\ X\ \middle|\ \langle A_i, X\rangle = a_i\ (i = 1, \ldots, k),\ X \in \mathcal{S}_n^+\ \right\}, \\ \mathcal{T} &:= \{\ X\ |\ \langle B_i, X\rangle \le b_i\ (i = 1, \ldots, l)\ \}. \end{aligned} \tag{3.4}$$

[1] What is "large" of course depends on the number of variables n and the type and structure of the optimization problem.

We assume that maintaining only set $\mathcal{S}$ results in an SDP that is still manageable by `(IPMs)`. But the inclusion of $\mathcal{T}$ would make the SDP computationally far too expensive.[2] Thus we suggest to maintain $X \in \mathcal{S}$ explicitly and put $X \in \mathcal{T}$ into the cost function by taking the partial Lagrangian dual

$$L(X, y) = \langle C, X \rangle + \sum_{i=1}^{l} y_i (b_i - \langle B_i, X \rangle).$$

Now we can rewrite the original problem (3.4) as

$$p^* = \max_{X \in \mathcal{S}} \min_{y \geq 0} L(X, y).$$

Assuming the usual strict feasibility conditions and applying the Minimax inequality (2.1) yields

$$p^* = \min_{y \geq 0} f(y),$$

where $f(y) = \max_{X \in \mathcal{S}} L(X, y)$. Thus to compute p^*, we can minimize f. f is the pointwise maximum of linear functions and therefore continuous and convex but not differentiable at points where the maximum is not unique.

We use the bundle method tailored to our problem (see [110] for a comprehensive survey) to minimize the nonsmooth function f over $y \geq 0$. The bundle method iteratively evalutes f at some trial points and uses subgradient information to obtain new iterates. Evaluating f amounts to solving an SDP over the set $\mathcal{S}$ which we have assumed to be manageable by `(IPMs)`. If we have $f(y^*) = L(X^*, y^*)$, then the maximum of L is attained at $X^* \in \mathcal{S}$ for some given y^*. Setting

$$g_i^* := b_i - \langle B_i, X^* \rangle, \ i = 1, \ldots, l,$$

the inequality

$$f(y) \geq L(X^*, y) = \langle C, X^* \rangle + \sum_{i=1}^{l} y_i (b_i - \langle B_i, X^* \rangle) = f(y^*) + \sum_{i=1}^{l} g_i^* (y_i - y_i^*), \ \forall y \in \mathbb{R}^l, \tag{3.5}$$

defines g^* to be a subgradient of f at y^*.

Now suppose we have evaluated f at $k \geq 1$ feasible points $y_1 \geq 0, \ldots, y_k \geq 0$ with respective maximizers $X_1, \ldots, X_k$ and subgradients $g_1, \ldots, g_k$. We denote the current iterate by $\overline{y} \in \{y_1, \ldots, y_k\}$ and set $f_i := f(y_i)$. The subgradient inequality (3.5) implies

$$f(y) \geq \max_{i \in \{1, \ldots, k\}} \left\{ f_i + g_i^\top (y - y_i) \right\} =: f_m(y), \ \forall y \in \mathbb{R}^l.$$

The minorant $f_m(y)$ is equal to $f(y)$ for $y \in \{y_1, \ldots, y_k\}$ and thus can be used to approximate f in the neighbourhood of the current iterate $\overline{y}$. To simplify the presentation we rewrite the minorant

$$f_m(y) = \max_{\lambda \in \Delta^k} \lambda^\top (H + G^\top y),$$

where $H = (h_1, \ldots, h_k)^\top, \ h_i = f_i - g_i^\top y_i, \ G = (g_1, \ldots, g_k)$ and Δ^k denotes the k-dimensional standard simplex. Additionally we add a regularization term to f_m

$$f_r(y) := f_m(y) + \frac{1}{2t} \|y - \overline{y}\|^2,$$

[2] Exactly this situation occurs for semidefinite relaxations of ordering problems, where $\mathcal{S}$ relates to the elliptope (4.9) and $\mathcal{T}$ consists of different types of inequalities, e.g. the triangle inequalities (4.12).

where $t \in \mathbb{R}^+$ controls how close we stay to $\overline{y}$. Introducing Lagrange multipliers η for the sign constraints on y and applying again the Minimax inequality yields the following dual problem

$$\min_{y \geq 0} f_r(y) = \max_{\lambda \in \Delta^k, \eta \geq 0} \min_y \lambda^\top (H + G^\top y) + \frac{1}{2t}\|y - \overline{y}\|^2 - y^\top \eta.$$

The inner minimization is a strictly convex unconstrained quadratic optimization problem in y, hence we can replace it by asking that the first-order optimality conditions

$$y = \overline{y} + t(\eta - G\lambda),$$

hold. Using this relation yields the following outer maximization problem

$$\max_{\lambda \in \Delta^k, \eta \geq 0} \lambda^\top (H + G^\top \overline{y}) - \frac{t}{2}\|\eta - G\lambda\|^2 - \overline{y}^\top \eta.$$

We solve this convex quadratic optimization problem in λ and η approximately by keeping alternately one set of the variables constant. Keeping η constant results in a convex quadratic problem over the standard simplex Δ^k and keeping λ constant allows to solve η coordinatewise, see e.g. [101]. We iterate this process several times and then use the estimates $\overline{\lambda}$ and $\overline{\eta}$ to get a new feasible dual point

$$y_{k+1} = \overline{y} + t(\overline{\eta} - G\overline{\lambda}).$$

Minimizing $f_r(y)$ additionally yields a new primal matrix $X_{k+1} := \sum_{i=1}^k \lambda_i X_i$. Finally we evaluate f at y_{k+1} and use some standard criteria to decide whether y_{k+1} becomes the new trial point. Under appropriate stopping conditions (X_k, y_k) converge towards an optimal primal-dual solution pair of the original problem (3.4), see e.g. [75, 142, 189].

To further improve efficiency, we only dualize those constraints in every iteration, which are likely to be active at the optimum. So we face a situation well known from active set methods. If we would know the constraints active at the optimum, we would not care about the other constraints any more. But this information is not explicitly available to us. So we use primal feasibility and dual optimality information to identify important constraints. In every iteration we add constraints (strongly) violated for the actual primal iterate X_k and remove constraints associated to components of the dual multiplier y_k that are close to zero. Thus f changes in the course of the algorithm. A convergence analysis of this dynamic version of the bundle method can be found in [21].

Part II

Theory & Algorithms

Chapter 4

The Linear Ordering Problem

4.1 Introduction

In its matrix version the Linear Ordering Problem (LOP) can be defined as follows. Given an $n \times n$ matrix $D = (d_{ij})$ of integers, find a simultaneous permutation ϕ of the rows and columns of D such that

$$\sum_{1 \le i < j \le n} d_{\phi(i),\phi(j)},$$

is maximized. Equivalently, we can interpret d_{ij} as weights of a complete directed graph G with vertex set $\mathcal{N} := \{1, \ldots, n\}$. A tournament consists of a subset of the arcs of G containing for every pair of nodes i and j either arc (i, j) or arc (j, i), but not both. Then (LOP) consists of finding an acyclic tournament, i.e. a tournament without directed cycles, of G of maximum total edge weight

$$z_* := \max \left\{ \sum\nolimits_{i<j} d_{\phi(i),\phi(j)} : \ \ \phi \in \Pi \right\}. \tag{4.1}$$

The permutation ϕ gives the ordering of the vertices N of G and the cost function consists of the sum of all edge weights d_{uv} where u comes before v in this ordering. The set of permutations is denoted by Π.

(LOP) is equivalent to the acyclic subdigraph problem and the feedback arc set problem. It is well known to be NP-hard [82] and it is even NP-hard to approximate (LOP) within the factor $\frac{65}{66}$ [167]. Surprisingly there is not much known about heuristics with approximation guarantees. If all entries of D are nonnegative, a $\frac{1}{2}$-approximation is trivial, but no better polynomial time approximation is known. To narrow the quite large gap $[\frac{1}{2}, \frac{65}{66}]$ is a challenging open problem.

Currently available exact algorithms include a Branch-and-Bound algorithm that uses a linear programming based lower bound by Kaas [122], a Branch-and-Cut algorithm proposed by Grötschel, Jünger and Reinelt [89] and a combined interior-point cutting-plane algorithm by Mitchell and Borchers [156] who explore polyhedral relaxations of the problem and also provide computational results using Branch-and-Cut. The current state-of-the-art Branch-and-Cut algorithm was developed by the working group of Prof. Reinelt in Heidelberg and is based on sophisticated cut generation procedures (for details see [152]). It can solve large instances from specific instance classes with up to 150 objects, while it fails on other much smaller instances with only 50 objects ("RandB" problems). Recently Oswald [169] could solve some "RandB" problems using linear programming relaxations based on betweenness variables that are more expensive to solve but yield tighter lower bounds. For a detailed overview over benchmark instances for (LOP) solved and not yet solved see [153, Appendix].

There exist many heuristics and methaheuristics for (LOP) and some of them are quite good in finding the optimal solution for large instances in reasonable time. For a recent survey and comparison see [152, 153]. In the following sections we focus on the computation of reasonable upper bounds to the optimal

solution which in general is more difficult and time-consuming than the heuristic deduction of reasonable lower bounds (good feasible solutions). We will also combine lower and upper bound computations to specify intervals for the optimal solutions. If we are able to close the gap between lower and upper bound, this provides us with an optimality certificate.

(LOP) arises in a large number of applications in such diverse fields as economy, sociology (determination of ancestry relationships [84]), graph drawing (one sided crossing minimization [120]), archaeology, scheduling [23], assessment of corruption perception [1] and ranking in sports tournaments. In 1959 Kemeny [129] posed the first application of (LOP) (Kemeny's problem) concerning the aggregation of individual orderings to a common one in the best possible way. Later on further problems were proposed in the context of mathematical psychology and the theory of social choice that can be formulated as linear ordering problems [74, 198]. The probably most established application of (LOP) is the triangulation of input-output matrices of an economy. Leontief [143, 144] was awarded the Nobel Prize in 1973 for his research on input-output analysis. Input-output analysis is a field of practical importance in economics as it is used for forecasting the development of industries, for structural planning and for structural comparisons between different countries [39, 111]. Its central component is the input-output table or matrix which contains the transactions between the different branches or sectors of an economy in a certain time period. Now triangulation of an input-output matrix allows a descriptive analysis of the transactions between the sectors as it establishes a hierarchy of all sectors such that the amount of flow incompatible with this hierarchy is as small as possible. For more details on the mentioned applications as well as for a detailed polyhedral analysis of the linear ordering polytope $\mathcal{P}_{LOP}$, we again refer to the recent book [152].

The remainder of this chapter is based on Sections 2 and 3 of the paper "Semidefinite Relaxations of Ordering Problems" [115]. In the next section we review the standard linear programming relaxation for (LOP) based on ordering variables. We will point out features and computational limitations of Branch-And-Cut methods using this relaxation. As an alternative to this standard approach for (LOP) we will investigate a more expensive but tighter relaxation based on semidefinite programming in Section 4.3 that pays off for difficult (e.g. highly symmetric) instances. For an experimental comparison of the practical strength of the proposed relaxations see Chapter 10.

4.2 Exact Approaches Based on Linear Programming

(LOP) has a natural formulation as an integer linear program (ILP) in 0-1 variables. An instance of the problem is defined by the $n \times n$ matrix $D = (d_{ij})$. We introduce binary ordering variables x_{ij} with $x_{ij} = 1$ if object i comes before object j and $x_{ij} = 0$ otherwise. Then it is not hard to show that the following constraints describe linear orderings of the objects in $\mathcal{N}$:

$$x_{ij} + x_{ji} = 1, \quad i \neq j \in \mathcal{N}, \tag{4.2}$$

$$x_{ij} + x_{jk} + x_{ki} \in \{1, 2\}, \quad i, j, k \in \mathcal{N}, \tag{4.3}$$

$$x_{ij} \in \{0, 1\}, \quad i \neq j \in \mathcal{N}. \tag{4.4}$$

The first condition models the fact that either object i is before object j or object j is before object i. The second condition rules out the existence of directed 3-cycles and is sufficient to insure that there is no directed cycle. Hence the feasible solutions for these constraints describe acyclic tournaments of the complete directed graph G with vertex set $\mathcal{N}$ [203, 209]. Maximizing $\sum_{i \neq j} d_{ij} x_{ij}$ over the constraints (4.2)–(4.4) therefore solves (LOP). The equations (4.2) are used to eliminate x_{ji} for $j > i$. This leads to the following formulation of (LOP) as linear program in binary variables (cf. [89])

$$z_* = \max \left\{ \sum\nolimits_{1 \leq i < j \leq n} (d_{ij} - d_{ji}) x_{ij} + d_{ji} : \; x \in \mathcal{P}_{LOP} \right\}, \tag{LOP}$$

where the linear ordering polytope is defined as

$$\mathcal{P}_{LOP} = \text{conv}\{\ x : x \in \{0,1\}^{\binom{n}{2}},\ \ 0 \le x_{ij} + x_{jk} - x_{ik} \le 1,\ \ i < j < k \in \mathcal{N}\ \}.$$

The linear relaxation (LP$_{\text{LOP}}$) is obtained by leaving out the integrality conditions on the variables. This results in a linear program with $\binom{n}{2}$ variables and $2\binom{n}{3}$ three-cycle inequalities. (LP$_{\text{LOP}}$) is the basis for exact methods which use the continuous linear relaxation in an enumerative scheme. (LP$_{\text{LOP}}$) poses a serious challenge to standard LP solvers, once $n \approx 200$. In Section 10.3 we will demonstrate how to use the bundle method (cf. Chapter 3.3) to considerably speed up the solution of (LP$_{\text{LOP}}$) for large instances.

In the following section we will analyze a stronger relaxation for (LOP) based on matrix liftings. For this purpose it is convenient to reformulate (LOP) in variables taking the values -1 and $+1$. The variable transformation

$$y_{ij} = 2x_{ij} - 1 \tag{4.5}$$

leads to the equivalent formulation of (LOP)

$$z_* = \max\ \left\{\ \sum\nolimits_{1\le i<j\le n} (d_{ij} - d_{ji})\frac{y_{ij}+1}{2} + d_{ji}\ :\ y_{ij} \in \{-1,1\},\ |y_{ij} + y_{jk} - y_{ik}| = 1,\ i < j < k \in \mathcal{N}\ \right\}.$$

In [98] it is shown that one can easily switch between the $\{0,1\}$ and $\{-1,1\}$ formulations of bivalent problems so that the resulting bounds remain the same and structural properties (like semidefiniteness constraints in SDPs) are preserved. Relaxing the integrality condition $y_{ij} \in \{-1,1\},\ i < j \in \mathcal{N}$ gives again the linear relaxation (LP$_{\text{LOP}}$)

$$z_{LP} := \max \sum_{1\le i<j\le n} (d_{ij} - d_{ji})\ \frac{y_{ij}+1}{2} + d_{ji}, \tag{4.6a}$$

$$\text{subject to}\quad -1 \le y_{ij} + y_{jk} - y_{ik} \le 1,\ \ i < j < k \in \mathcal{N}, \tag{4.6b}$$

$$-1 \le y_{ij} \le 1,\ \ i < j \in \mathcal{N}. \tag{4.6c}$$

The upper bound z_{LP} may lead to gaps between z_* and z_{LP} which are too large for efficient pruning in Branch-and-Bound enumeration. This happens for example for the highly symmetrical (LOP) instances based on the so-called Paley Graphs, where already an instance with 31 objects cannot be solved to optimality. Thus for such problems it would be desirable to have some tighter approximation available. One possibility is to add further cuts to the separation procedure (e.g. Chvátal-Gomory Cuts, Mod-2 Cuts, Möbius ladders - for details see [152, Chapters 5 and 6]). But this approach gets too expensive for larger instances and also does not succeed for the Paley instances. Oswald [169] proposed to formulate (LOP) via triple variables and thus get a tighter basic relaxation (for details see Section 7.2). This approach (not yet published) is related to LP-based approches used for tackling the Minimum Linear Arrangement Problem and the Single Row Facility Layout Problem (discussed in more detail in Sections 5.2 and 6.2). In the following section we take a closer look at even tighter basic relaxations for (LOP) based on semidefinite optimization.

4.3 Exact Approaches Based on Semidefinite Programming

We investigate in some detail how the linear description of the problem can be 'lifted' into the semidefinite model to yield tight approximations. Even though two basic semidefinite models have been proposed in the literature for specific ordering problems ([13] for the Single Row Facility Layout Problem and [31] for Bipartite Crossing Minimization), we provide the first systematic investigation of the semidefinite model in terms of constraint derivation and areas of application. First of all, we deduce a semidefinite formulation for (LOP), the easiest ordering problem.

The matrix lifting approach takes a vector y and considers the matrix $Y = yy^T$. We are interested in lifting (LOP) into quadratic space and thus consider the linear-quadratic ordering polytope

$$\mathcal{P}_{LQO} := \text{conv}\left\{ \begin{pmatrix}1\\ y\end{pmatrix}\begin{pmatrix}1\\ y\end{pmatrix}^\top \;:\; y \in \{-1,1\}^{\binom{n}{2}},\ y \text{ satisfies (4.6b)} \right\}.$$

The nonconvex equation $Y - yy^T = 0$ is relaxed to the constraint

$$Y - yy^T \succeq 0,$$

which is convex due to (the Schur complement) Theorem 2.2

$$\begin{pmatrix} 1 & y^T \\ y & Y \end{pmatrix} \succeq 0 \Leftrightarrow Y - yy^T \succeq 0.$$

Moreover, the main diagonal entries of Y correspond to y_{ij}^2, hence $\text{diag}(Y) = e$, the vector of all ones. We therefore conclude that any $Y \in \mathcal{P}_{LQO}$ satisfies

$$Y - yy^T \succeq 0, \quad \text{diag}(Y) = e. \tag{4.7}$$

To simplify notation let us introduce

$$Z = Z(y,Y) := \begin{pmatrix} 1 & y^T \\ y & Y \end{pmatrix}, \tag{4.8}$$

where $\dim(Z) = \binom{n}{2} + 1 = \zeta$ and $Z = (z_{ij})$. In this case $Y - yy^T \succeq 0 \Leftrightarrow Z \succeq 0$. Hence, $\mathcal{P}_{LQO}$ is contained in the elliptope

$$\mathcal{E} := \{\, Z :\ \text{diag}(Z) = e, Z \succeq 0 \,\}, \tag{4.9}$$

which is studied in detail by Laurent and Poljak [136, 137]. In order to express constraints on y in terms of Y, they have to be reformulated as quadratic conditions in y. A natural way to do this for $|y_{ij} + y_{jk} - y_{ik}| = 1$ consists in squaring both sides, leading to

$$y_{ij}^2 + y_{jk}^2 + y_{ik}^2 + 2(y_{ij,jk} - y_{ij,ik} - y_{ik,jk}) = 1, \quad i < j < k. \tag{4.10}$$

Since $y_{ij}^2 = 1$, this simplifies to

$$y_{ij,jk} - y_{ij,ik} - y_{ik,jk} = -1, \quad i < j < k. \tag{4.11}$$

In [31] it is shown that these equations formulated in the $\{0,1\}$ model describe the smallest linear subspace that contains $\mathcal{P}_{LQO}$. We now formulate (LOP) as a semidefinite optimization problem in bivalent variables.

Theorem 4.1 *The problem*

$$\max\left\{\sum\nolimits_{1\le i<j\le n}(d_{ij}-d_{ji})\frac{y_{ij}+1}{2} + d_{ji} : Z \text{ partitioned as in (4.8) satisfies (4.11)}, Z \in \mathcal{E}, y \in \{-1,1\}^{\binom{n}{2}}\right\}$$

is equivalent to (LOP).

Proof. Since $y_{ij}^2 = 1$ we have $\text{diag}(Y - yy^T) = 0$, which together with $Y - yy^T \succeq 0$ shows that in fact $Y = yy^T$. The 3-cycle equations (4.11) ensure that $|y_{ij} + y_{jk} - y_{ik}| = 1$ holds and thus y represents a feasible ordering. □

We are now dropping the integrality condition on y and obtain the following basic semidefinite relaxation of (LOP)

$$\max\ \{\ \langle C, Y\rangle + c^\top y + K :\ Z \text{ partitioned as in (4.8) satisfies (4.11)},\ \ Z \in \mathcal{E}\ \}, \tag{SDP$_1$}$$

with C is the zero matrix, $c_{ij} := \frac{d_{ij}-d_{ji}}{2},\ i<j \in \mathcal{N}$ and $K := \sum_{1\le i<j\le n} \frac{d_{ij}+d_{ji}}{2}$.

There are some obvious ways to tighten $(\mathtt{SDP}_1)$. First of all we observe that Y and also Z are actually matrices with $\{-1,1\}$ entries in the original (LOP) formulation. Hence they satisfy the triangle inequalities, defining the metric polytope $\mathcal{M}$

$$\mathcal{M} = \left\{ Z : \begin{pmatrix} -1 & -1 & -1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} z_{ij} \\ z_{jk} \\ z_{ik} \end{pmatrix} \le e,\ \ 1 \le i < j < k \le \zeta, \right\}. \tag{4.12}$$

We note that the metric polytope is defined through $4\binom{\zeta}{3} \approx \frac{1}{12}n^6$ facets. They are used as triangle inequalities of the max-cut polytope in [148, 192, 197]. The basic relaxation $(\mathtt{SDP}_1)$ can therefore be improved by asking in addition that $Z \in \mathcal{M}$ yielding $(\mathtt{SDP}_2)$.

Another generic improvement was suggested by Lovász and Schrijver in [148]. Applied to our problem, their approach suggests to multiply the 3-cycle inequalities (4.6b)

$$1 - y_{ij} - y_{jk} + y_{ik} \ge 0,\ \ 1 + y_{ij} + y_{jk} - y_{ik} \ge 0,$$

by the nonnegative expressions $(1-y_{lm})$ and $(1+y_{lm})$. This results in the following inequalities

$$\begin{aligned} -1 - y_{lm} \le y_{ij} + y_{jk} - y_{ik} + y_{ij,lm} + y_{jk,lm} - y_{ik,lm} \le 1 + y_{lm},\ \ i<j<k,\ l<m, \\ -1 + y_{lm} \le y_{ij} + y_{jk} - y_{ik} - y_{ij,lm} - y_{jk,lm} + y_{ik,lm} \le 1 - y_{lm},\ \ i<j<k,\ l<m. \end{aligned} \tag{4.13}$$

We define the polytope $\mathcal{LS}$

$$\mathcal{LS} := \{\ Z : Z \text{ satisfies (4.13)}\ \}, \tag{4.14}$$

consisting of $4\binom{n}{3}\binom{n}{2} \approx \frac{1}{3}n^5$ constraints. The basic relaxation $(\mathtt{SDP}_1)$ can therefore also be improved by asking in addition that $Z \in \mathcal{LS}$ yielding $(\mathtt{SDP}_3)$.

In summary, we get the following tractable relaxation of $\mathcal{P}_{LQO}$, part of which (without the Lovász-Schrijver cuts (4.14)) has been investigated in [31] for Bipartite Crossing Minimization and in [13] for Single-Row Layout Problems

$$\max\left\{ \langle C, Y\rangle + c^\top y + K \ :\ Z \text{ partitioned as in (4.8) satisfies (4.11)},\ Z \in (\mathcal{E} \cap \mathcal{M} \cap \mathcal{LS}) \right\}. \tag{$\mathtt{SDP}_4$}$$

There are further inequality classes to tighten relaxation $(\mathtt{SDP}_4)$ without making it incomputable. Additionally to the proposed Lovász-Schrijver lifting procedure above, we could also multiply different pairs of 3-cycle inequalities to obtain feasible inequalities for Z. Also facets of the betweenness polytope for 5 objects $\mathcal{P}^5_{\mathcal{BTW}}$ or a heuristically selected subset of pentagonal inequalities could be included in $(\mathtt{SDP}_4)$. We refrain from doing so because we have not yet conducted fair practical experiments with these constraint classes and refer to Sections 7.2 and 7.3 for a detailed theoretical discussion.

We close this section with proving some basic properties for the proposed SDP relaxations. First we relate $(\mathtt{SDP}_1)$ to the linear relaxation $(\mathtt{LP}_{\mathtt{LOP}})$, then we give some classes of inequality constraints that are implicitly included in $(\mathtt{SDP}_4)$ and finally we show that working with the larger original formulation of the ordering problem (4.2)–(4.4) does not improve the strength of the semidefinite relaxation.

Proposition 4.2 *The basic semidefinite relaxation* $(\mathtt{SDP}_1)$ *is at least as strong as the linear relaxation* $(\mathtt{LP}_{\mathtt{LOP}})$.

Proof. First we have to verify that for any Z feasible for $(\mathtt{SDP}_1)$ the vector y in its first column satisfies the 3-cycle inequalities (4.6b) on the layers. This follows from the semidefiniteness of the following submatrix of Z

$$\begin{pmatrix} 1 & y_{ij} & y_{ik} & y_{jk} \\ y_{ij} & 1 & y_{ij,ik} & y_{ij,jk} \\ y_{ik} & y_{ik,ij} & 1 & y_{ik,jk} \\ y_{jk} & y_{jk,ij} & y_{jk,ik} & 1 \end{pmatrix}.$$

because expanding

$$\begin{pmatrix}1\\1\\-1\\1\end{pmatrix}^\top \begin{pmatrix}1 & y_{ij} & y_{ik} & y_{jk}\\ y_{ij} & 1 & y_{ij,ik} & y_{ij,jk}\\ y_{ik} & y_{ik,ij} & 1 & y_{ik,jk}\\ y_{jk} & y_{jk,ij} & y_{jk,ik} & 1\end{pmatrix}\begin{pmatrix}1\\1\\-1\\1\end{pmatrix} \geq 0, \text{ and } \begin{pmatrix}1\\-1\\1\\-1\end{pmatrix}^\top \begin{pmatrix}1 & y_{ij} & y_{ik} & y_{jk}\\ y_{ij} & 1 & y_{ij,ik} & y_{ij,jk}\\ y_{ik} & y_{ik,ij} & 1 & y_{ik,jk}\\ y_{jk} & y_{jk,ij} & y_{jk,ik} & 1\end{pmatrix}\begin{pmatrix}1\\-1\\1\\-1\end{pmatrix} \geq 0,$$

yields

$$4 + 2(y_{ij} + y_{jk} - y_{ik} + y_{ij,jk} - y_{ij,ik} - y_{ik,jk}) \geq 0, \text{ and } 4 + 2(y_{ik} - y_{ij} - y_{jk} + y_{ij,jk} - y_{ij,ik} - y_{ik,jk}) \geq 0.$$

and then applying (4.11) finally gives

$$-1 \leq y_{ij} + y_{jk} - y_{ik} \leq 1.$$

Now using (4.11) gives (4.6b). Additionally the bound constraints (4.6c) on the components of y implicitly follow from $Z \in \mathcal{E}$. Thus (SDP_1) is at least as strong as (LP_{LOP}). □

Next we generalize the above result. Even most of the small facets that are usually used for separation in linear programming based Branch-and-Cut approaches are already implicitly included in (SDP_4).

Theorem 4.3 *The 3-cycle inequalities* (4.6b)*, all facet classes of $\mathcal{P}^n_{LOP}$ for $n \leq 6$ and all but one facet classes of $\mathcal{P}^7_{LOP}$ are implicitly included in* (SDP_4).

Proof. We have already proven in Proposition 4.2 that the 3-cycle inequalities (4.6b) are dominated by (4.11) together with $Z \succcurlyeq 0$. The facet classes for $\mathcal{P}^n_{\text{LOP}}$ for $n \leq 5$ are already included in (LP_{LOP}) and thus by Proposition 4.2 they are also included in (SDP_4). In Section 10.1 we also show by computation that all but one facet classes of $\mathcal{P}^6_{\text{LOP}}$ and $\mathcal{P}^7_{\text{LOP}}$ (e.g. Möbius ladders on 6 nodes) are included in (SDP_4). For details on the computations and the respective involved constraint classes see Table 10.1. □

The original formulation of the ordering problem was done in dimension $2\binom{n}{2}$, as we introduced variables y_{ij} for $i \neq j$. The equations $y_{ij} + y_{ji} = 0$ were then used to eliminate half the variables, leading to a new model in dimension $\binom{n}{2}$. Would we get a stronger semidefinite relaxation by working with matrices of order $2\binom{n}{2}$ instead of $\binom{n}{2}$? It is not difficult to show that this is not the case.

Proposition 4.4 *Let m linear equality constraints $Ay = c$ be given. If there exists some invertible $m \times m$ matrix B, we can partition the linear system in the following way*

$$Ay = \begin{bmatrix}B & C\end{bmatrix}\begin{bmatrix}v\\u\end{bmatrix} = c.. \tag{4.15}$$

Then we do not weaken the relaxation by first moving into the subspace given by the equations, and then lifting the problem to matrix space.

Proof. Solving for v in (4.15) yields $v = B^{-1}(c - Cu)$. Thus

$$\begin{bmatrix}1\\u\\v\end{bmatrix} = \begin{bmatrix}1 & 0\\ 0 & I\\ B^{-1}c & -B^{-1}C\end{bmatrix}\begin{bmatrix}1\\u\end{bmatrix} = D\begin{bmatrix}1\\u\end{bmatrix},$$

defining the full column rank matrix D. From this it is clear that

$$\begin{bmatrix}1\\u\\v\end{bmatrix}\begin{bmatrix}1\\u\\v\end{bmatrix}^T = D\begin{bmatrix}1\\u\end{bmatrix}\begin{bmatrix}1\\u\end{bmatrix}^T D^T.$$

Therefore

$$T := \begin{bmatrix} 1 & u^\top & v^\top \\ u & U & W^\top \\ v & W & V \end{bmatrix} = D \begin{bmatrix} 1 & u^\top \\ u & U \end{bmatrix} D^\top,$$

and thus

$$\begin{bmatrix} 1 & u^\top \\ u & U \end{bmatrix} \succcurlyeq 0 \Leftrightarrow T \succcurlyeq 0.$$

□

Chapter 5

The Minimum Linear Arrangement Problem

5.1 Introduction

The minimum Linear Arrangement Problem (minLA) can be defined as follows. Given an undirected graph $G = (V, E)$ find a permutation $\phi : V \to \{1, \ldots, n\}$ minimizing $\sum_{i,j \in E} |\phi(i) - \phi(j)|$.

$$z_* = \min_{\phi \in \Pi} \sum_{i,j \in E} |\phi(i) - \phi(j)|. \qquad \text{(minLA)}$$

(minLA) belongs to a larger class of combinatorial optimization problems. The so-called graph layout problems ask for a permutation of V that optimizes some function of pairwise vertex distances. (minLA) is NP-hard [83] (even if G is bipartite [81]) and was originally proposed by Harper [92, 93] in 1964 to develop error-correcting codes with minimal average absolute errors. (minLA) has also applications in VLSI design [206], in single machine job scheduling [2, 181] and in computational biology [126, 157]. It is also used for the layout of entity relationship models [38] and data flow diagrams [77].

The theoretically fastest known exact algorithm for general graphs is based on dynamic programming and runs in $O(2^{|V|}|E|)$ time [132]. There exist approximation algorithms for (minLA) with performance guarantee $O(\log n)$ [25, 180] and $O(\sqrt{\log n} \log \log n)$ [37, 72]. Díaz et al. [65] provide a survey on graph layout problems in general as well as on (minLA) in particular. In the next section we recall the main ideas of the very recent, most competitive exact algorithms for (minLA) based on linear programming. For information and references on other exact methods, heuristic algorithms and polynomially-solvable special cases of (minLA), we again refer the reader to [65]. In Section 5.3 we explain how to extend the semidefinite approach for (LOP), introduced in the previous chapter, to (minLA). On the practical side, (minLA) is very challenging. Back in 2009 the best exact method for (minLA) was the one based on dynamic programming mentioned above and thus was restricted to instances of size $n \leq 30$. For an experimental comparison of the linear and semidefinite approaches proposed in the following sections we refer to Chapter 11.

5.2 Exact Approaches Based on Linear Programming

The most natural way to formulate (minLA) as an ILP is by the combined use of position and distance variables. Let the position variable x_{ij} be defined as follows

$$x_{ij} = \begin{cases} 1, & \text{if vertex } i \text{ is placed in position } j \\ 0, & \text{otherwise.} \end{cases}$$

The distance variables are given by $d_e = |\phi(i) - \phi(j)|,\ \forall e = \{i,j\} \in E$. Now we can formulate (minLA) as the following ILP

$$\begin{aligned} \min & \sum_{e \in E} d_e, \\ \text{subject to } & \sum_{j \in V} x_{ij} = 1, \quad i \in V, \\ & \sum_{i \in V} x_{ij} = 1, \quad j \in V, \\ & d_e \geq |p-q|(x_{ip} + x_{jq} - 1), \quad e = \{i,j\} \in E,\ p,q \in V, \\ & x_{ij} \in \{0,1\}, \quad i,j \in V. \end{aligned} \tag{minLA}$$

Replacing the integrality conditions on the position variables by 0-1-bounds yields a linear programming relaxation. Unfortunately, this relaxation is too weak for practical purposes, as it yields 0 independent of underlying graph, i.e. set $x_{ij} = \frac{1}{|V|},\ \forall i,j \in V$ and $d_e = 0,\ \forall e \in E$. Caprara et al. [35] propose to omit the position variables and instead to introduce new constraints (rank inequalities) on the distance variables. Dependent on the number of distance variables in their linear program (introduce d_e only for $e \in E$ versus use all $\binom{n}{2}$ distance variables) they obtain a sparse and a dense linear programming relaxation. They try to combine the advantages of both of these relaxations by taking the dense one and projecting it onto the variable space of the sparse one. For this combined relaxation, they derive facet inducing inequalities of the underlying polyhedron and discuss the associated separation problems.

A similar approach was suggested by Seitz [191]. She suggests to use the binary variables

$$d_{ijk} = \begin{cases} 1, & \text{if } |\phi(i) - \phi(j)| = k \\ 0, & \text{otherwise,} \end{cases}$$

where $i < j \in V$ and $1 \leq k \leq |V| - 1$. These variables are related to the distance variables d_e in the following way

$$d_e = \sum_{k=1}^{n-1} k d_{ijk}, \quad e = \{i,j\} \in E.$$

Thus Seitz uses the above relation to formulate (minLA) in the d_{ijk}-variables. She deduces many valid inequalities that are used in a Branch-and-Cut-and-Price algorithm.

There exists another very competitive ILP formulation of (minLA) that was proposed by Caprara et al. [34] and realized by Schwarz [190]. Let us introduce binary betweenness variables $\xi_{ijk} (\{i,j\} \in E,\ k \in V,\ i \neq k \neq j)$

$$\xi_{ijk} = \begin{cases} 1, & \text{if } \phi(i) < \phi(k) < \phi(j) \text{ or } \phi(j) < \phi(k) < \phi(i) \\ 0, & \text{otherwise.} \end{cases}$$

We can also use these variables to express the distance variables

$$d_e = 1 + \sum_{k \in V \setminus \{i,j\}} \xi_{ijk}, \quad e = \{i,j\} \in E,$$

and thus to (partially) formulate (minLA) and also the strengthening rank inequalities via betweenness variables. Caprara et al. conduct a detailed polyhedral study, explain how to separate different types of rank inequalities and design a Branch-and-Cut algorithm. Their approach is closely connected to the one proposed by Amaral [7] for the single row facility layout problem (SRFLP) (for details see Section 6.2). The main difference of these two approaches is the number of the variables introduced. While Caprara et

al. work with $|E|(n-2)$ variables, Amaral introduces a betweenness variable for every triple of vertices, because (SRFLP) can be interpreted as (minLA) with edge weights on complete graphs. Using a sparse model, Caprara et al. have to check if the binary solution vectors of their partial formulation of (minLA) describe an arrangement on the set of vertices. If this is not the case they add a suitable linear inequality to cut the actual solution vector off. Compared to Caprara et al., Amaral uses a simpler separate procedure, because of the high costs associated to separating complex rank inequalities for instances of reasonable size in the dense model.

5.3 Exact Approaches Based on Semidefinite Programming

Another way to formulate (minLA) is as a quadratic ordering problem. Let y_{ij} satisfy

$$y_{ij} + y_{ji} = 0,\ \forall\ i \neq j \in \mathcal{N}, \tag{5.1}$$

$$y_{ij} + y_{jk} + y_{ki} \in \{-1, 1\},\ \forall\ i, j, k \in \mathcal{N}, \tag{5.2}$$

$$y_{ij} \in \{-1, 1\},\ \forall\ i \neq j \in \mathcal{N}, \tag{5.3}$$

then the products $y_{ik}y_{kj}$ are equal to 1 iff k lies between i and j. Hence the term

$$\frac{|V|-2}{2} + \sum_{k \in V \setminus \{i,j\}} \frac{y_{ik}y_{kj}}{2},$$

gives the number of vertices between i and j in the arrangement ϕ defined by y and (minLA) can be written as

$$z_* = \min \left\{ \frac{|V||E|}{2} + \sum_{i,j \in E} \sum_{k \in E \setminus \{i,j\}} \frac{y_{ik}y_{kj}}{2} : y_{ij} \text{ satisfies (5.1), (5.2) and (5.3)} \right\}. \tag{minLA}$$

Applying the results for (LOP) from Section 4.3, we can formulate (minLA) as the following SDP

$$z_* = \min \left\{ \langle C, Y \rangle + K :\ \ Z \text{ partitioned as in (4.8) satisfies (4.11)},\ Z \in \mathcal{E},\ y \in \{-1,1\}^{\binom{n}{2}} \right\}. \tag{minLA}$$

where $K := \frac{|V||E|}{2}$ and the cost matrix $C = (C_{ij,kl})$, $i < j, k < l \in V$ is defined for every $\{i,j\} \in E,\ i < j$ and $k \in V,\ i \neq k \neq j$ as follows

$$C_{ki,kj} = -\frac{1}{4}, \quad k < i, \qquad C_{ik,kj} = \frac{1}{4}, \quad i < k < j, \qquad C_{ik,jk} = -\frac{1}{4}, \quad j < k,$$

and has zero entries otherwise. The SDP formulations of (LOP) and (minLA) just differ with respect to their cost function. Hence we can also deduce for (minLA) the four SDP relaxations (SDP_1)–(SDP_4) with C and K defined above and c equal to the zero vector. Contrary to (LOP), the more expensive but also stronger relaxation (SDP_4) outperforms the other ones in practice (for details see Section 14.7).

In Chapter 11 we give an extensive practical comparison of the different exact approaches mentioned above. Notice that Buchheim et al. [30] propose further ILP-based Branch-and-Cut algorithms for (minLA). Their ILPs result from the linearization of the above quadratic formulation of (minLA) using ordering variables. But their main goal is the design of general separation routines that can be used as a black box and hence can replace detailed polyhedral studies of the underlying polytope. Their implementations for (minLA) leave many degrees of freedom and a lot of room for improvement by tuning various parameters. Thus their algorithms yield rather weak computational results compared to the other exact approaches discussed above (for details see [30, Table 3]). Hence we do not consider their approach for further computational experiments in Chapter 11.

To provide a theoretical comparison between the methods based on linear and semidefinite programming, we would have to determine whether (SDP_4) is exact for many special types of rank inequalities,

e.g. circles, (double)stars and (bi)cliques. These special types are used as cutting planes by all linear programming approaches and often also define facets of the underlying polytopes. We already conducted computational experiments for circles, stars and (bi)cliques of small dimension. Based on theses experiments we conjecture that it is possible to provide such results showing the strength of the SDP relaxations for `(minLA)` (and also `(SRFLP)`). We shall give theorems in this vein for the Multi-level Crossing Minimization problem in Section 8.4.

Chapter 6

The Single Row Facility Layout Problem

6.1 Introduction

An instance of the Single-Row Facility Layout Problem (SRFLP) consists of n one-dimensional facilities, with given positive lengths $l_1, \ldots, l_n$, and pairwise connectivities c_{ij}. Now the task in (SRFLP) is to find a permutation ϕ of the facilities such that the total weighted sum of the center-to-center distances between all pairs of facilities is minimized

$$\min_{\phi \in \Pi} \sum_{i,j \in \mathcal{N}, i<j} c_{ij} z_{ij}^{\phi}, \tag{6.1}$$

where $\mathcal{N} := \{1, \ldots, n\}$, Π denotes the set of all layouts and z_{ij}^{ϕ} is the center-to-center distance between facilities i and j with respect to ϕ.

Several practical applications of (SRFLP) have been identified in the literature, such as the arrangement of rooms on a corridor in hospitals, supermarkets, or offices [195], the assignment of airplanes to gates in an airport terminal [201], the arrangement of machines in flexible manufacturing systems [107], the arrangement of books on a shelf and the assignment of disk cylinders to files [174].

On the one hand (SRFLP) (also known as one-dimensonal space allocation problem) is a special case of the weighted betweenness problem which is again a special case of the quadratic ordering problem (for details see Chapter 7). On the other hand the NP-hard [82] minimum linear arrangement problem (for details see Chapter 5) is a special case of (SRFLP) where all facilities have the same length and all connectivities are equal to 0 or 1. Hence (SRFLP) is also NP-hard.

Accordingly several heuristic algorithms have been suggested to tackle instances of interesting size of (SRFLP), e.g. [49, 87, 90, 106, 108, 135, 186, 187]. However, these heuristic approaches do not provide any optimality certificate, like an estimate of the distance from optimality, for the solution found.

Several exact approaches to (SRFLP) have also been proposed. Simmons [195] first studied (SRFLP) and suggested a branch-and-bound algorithm. Later on Simmons [196] pointed out the possibility of extending the dynamic programming algorithm of Karp and Held [127] to (SRFLP). This was later on implemented by Picard and Queyranne [174]. A nonlinear model was presented by Heragu and Kusiak [108] and various linear mixed integer programs were proposed by Love and Wong [149] and Amaral [5, 6]. However these models suffer from weak lower bounds and hence have high computation times and memory requirements. But just recently Amaral and Letchford [8] achieved significant progress in that direction through the first polyhedral study of the distance polytope for (SRFLP). They additionally showed that their approach is quite effective for instances with challenging size ($n \geq 30$). Amaral [7] suggested an LP-based cutting plane algorithm using betweenness variables that proved to be highly competitive and solved instances

with up to 35 facilities to optimality. Recently Sanjeevi and Kianfar [188] studied the polyhedral structure of Amaral's betweenness model in more detail and identified several classes of facet defining inequalities.

To obtain tight lower bounds for (SRFLP) without using branch-and-bound, SDP approaches are the best known methods to date. Anjos et al. [10] proposed the first SDP relaxation yielding bounds for instances with up to 80 facilities. Anjos and Vanelli [13] further tightened the SDP relaxation using triangle inequalities as cutting planes and gave optimal solutions for instances with up to 30 facilities that remained unsolved since 1988. Anjos and Yen [14] suggested an alternative SDP relaxation and achieved optimality gaps no greater than 5 % for large instances with up to 100 facilities. Using our strongest relaxation (SDP_4), we can theoretically and practically further improve on the tightness of the above SDP relaxations.

The remainder of this chapter is based on Section 2 of the paper "A Computational Study for the Single-Row Facility Layout Problem" [116]. In the following two sections we describe and compare the most successful modelling approaches to (SRFLP) from a theoretical point of view, pointing out their common connections to the Max-Cut [18, 91, 197] and the Quadratic Ordering Problem [30, 31]. For further details on this subject see also the recent survey of (SRFLP) by Anjos and Liers [12].

6.2 Exact Approaches Based on Linear Programming

The most intuitive modelling approach to (SRFLP) uses $\binom{n}{2}$ distance variables $z^{\phi}_{ij}, i, j \in \mathcal{N}$ and is related to the work of Caprara et al. [35] for (minLA) (for details see Section 5.2). This approach suffers from weak lower bounds of the corresponding linear programming relaxation, which results in large branch-and-bound trees and high computation times and memory requirements. Recently Amaral and Letchford [8] achieved significant progress in that direction by identifying several classes of valid inequalities and using them as cutting planes. The polytope containing the feasible distance variables z_{ij} for n facilities with lengths $l \in \mathcal{Z}^n$ is called distance polytope and defined as

$$\mathcal{P}^n_{Dis} := \text{conv}\left\{ z \in \mathbb{R}^{\binom{n}{2}} : \exists \phi \in \Pi \ : \ z_{ij} = z^{\phi}_{ij},\ i, j \in \mathcal{N}, i < j \right\}.$$

Amaral and Letchford show that the equation

$$\sum_{i,j \in \mathcal{N}, i<j} l_i l_j z_{ij} = \frac{1}{6}\left[\left(\sum_{i \in \mathcal{N}} l_i \right)^3 - \sum_{i \in \mathcal{N}} l_i^3 \right],$$

defines the smallest linear subspace that contains $\mathcal{P}^n_{Dis}$. They prove that clique inequalities, strengthened pure negative type inequalities and special types of hypermetric inequalities induce facets of $\mathcal{P}^n_{Dis}$. They further show the validity of rounded psd inequalities and star inequalities for $\mathcal{P}^n_{Dis}$ and use them together with the facet inducing inequalities as cutting planes in a Branch-and-Cut approach.

Amaral [7] proposed a formulation of (SRFLP) via betweenness variables that is closely related to the model of Caprara et al. [34] for (minLA) (for details see Section 5.2). Let us again introduce the binary variables ξ_{ijk}(now for $i, j, k \in \mathcal{N}, i < j, i \neq k \neq j$)

$$\xi_{ijk} = \begin{cases} 1, & \text{if department } k \text{ lies between departments } i \text{ and } j \\ 0, & \text{otherwise.} \end{cases} \tag{6.2}$$

We collect these betweenness variables in a vector ξ and define the betweenness polytope

$$\mathcal{P}^n_{BTW} := \text{conv}\ \{\xi \ : \ \xi \text{ represents an ordering of the elements of } \mathcal{N}\}.$$

In order to formulate (SRFLP) via ξ we need an appropriate objective function. For that purpose we use the relation

$$z^{\phi}_{ij} = \frac{1}{2}(l_i + l_j) + \sum_{\substack{k \in \mathcal{N}, \\ i \neq k \neq j}} l_k \xi_{ijk}, \quad i, j \in \mathcal{N}, i < j,$$

to rewrite (6.1) in terms of ξ (for details see [7, Propositions 1 and 2])

$$\min_{\xi \in \mathcal{P}^n_{BTW}} \sum_{\substack{i,j,k \in \mathcal{N}, \\ i<j, k<j}} (c_{ij} l_k - c_{ik} l_j)\, \xi_{ijk} + \sum_{\substack{i,j \in \mathcal{N}, \\ i<j}} \left(\frac{c_{ij}}{2}(l_i + l_j) + \sum_{\substack{k \in \mathcal{N}, \\ k>j}} c_{ij} l_k \right). \tag{6.3}$$

If department i comes before department j, department k has to be located mutually exclusive either left of department i, or between departments i and j, or right of department j. Thus the following equations are valid for $\mathcal{P}^n_{BTW}$

$$\xi_{ijk} + \xi_{ikj} + \xi_{jki} = 1, \quad i,j,k \in \mathcal{N}, i < j < k. \tag{6.4}$$

In [188] it is shown that these equations describe the smallest linear subspace that contains $\mathcal{P}^n_{BTW}$. To obtain an LP relaxation of `(SRFLP)`, we replace the integrality conditions on ξ with 0-1 bounds:

$$0 \leq \xi_{ijk} \leq 1, \quad i,j,k \in \mathcal{N}, i < j. \tag{6.5}$$

To further strengthen the relaxation, we can come up with additional valid inequalities. Let a subset $\{i,j,k,d\} \subset \mathcal{N}$ be given. On the one hand department d can not be located between the departments i and j, i and k and j and k at the same time. On the other hand if department d is between departments i and k then it also lies between departments i and j or j and k. Thus the inequalities

$$\xi_{ijd} + \xi_{jkd} + \xi_{ikd} \leq 2, \quad i,j,k,d \in \mathcal{N}, i < j < k \tag{6.6}$$

and

$$-\xi_{ijd} + \xi_{jkd} + \xi_{ikd} \geq 0, \quad \xi_{ijd} - \xi_{jkd} + \xi_{ikd} \geq 0, \quad \xi_{ijd} + \xi_{jkd} - \xi_{ikd} \geq 0, \quad i,j,k,d \in \mathcal{N}, i < j < k, \tag{6.7}$$

are valid for $\mathcal{P}^n_{BTW}$. Sanjeevi and Kianfar [188] showed that (6.7) unlike (6.6) are facet defining for $\mathcal{P}^n_{BTW}$. Amaral further generalizes (6.7) to a more complicated set of inequalities: Let $\beta \leq n$ be an even integer and let $S \subseteq \mathcal{N}$. For each $d \in S$, and for any partition (S_1, S_2) of $S \setminus \{d\}$ such that $|S_1| = \frac{1}{2}\beta$, the inequality

$$\sum_{p,q \in S_1, p<q} \xi_{pqd} + \sum_{p,q \in S_2, p<q} \xi_{pqd} \leq \sum_{p \in S_1, q \in S_2, p<q} \xi_{pqd} \tag{6.8}$$

is valid [7] and also facet-defining [188] for $\mathcal{P}^n_{BTW}$. Notice that (6.7) is a special case of (6.8) with $\beta = 4$. Minimizing (6.3) over (6.4)–(6.7) gives the basic linear relaxation `(LP)`. To construct stronger relaxations from `(LP)` Amaral proposes to use the inequalities $(6.8)_{\beta=6}$ as cutting planes (for further details on the practical realization of this approach see Section 12.1).

6.3 Exact Approaches Based on Semidefinite Programming

Anjos et al. [10] proposed to model `(SRFLP)` as a binary quadratic program using $\binom{n}{2}$ ordering variables. They deduced a semidefinite relaxation yielding tighter bounds but being more expensive to compute than the relaxation of Amaral. Later on further SDP approaches have been suggested to improve on the relaxation strength and/or reduce the computational effort involved [13, 14]. In the following we recall the different SDP approaches suggested by Anjos et al. and highlight their relations to our SDP relaxations (for details see Section 4.3). The main differences between the approaches are that Anjos et al. work with a slightly smaller polytope and use different algorithmic ideas to solve their relaxations in practice (for details on the practical performance of the approaches see Chapter 12).

Matrix-based relaxations for (SRFLP) can be deduced from the betweenness-based approach above by introducing bivalent ordering variables $y_{ij} (i, j \in \mathcal{N}, i < j)$

$$y_{ij} = \begin{cases} 1, & \text{if department } i \text{ lies before department } j \\ -1, & \text{otherwise,} \end{cases} \tag{6.9}$$

and using them to express the betweenness variables ξ

$$\xi_{ijk} = \frac{1 + y_{ik}y_{kj}}{2},\ i < k < j, \qquad \xi_{ijk} = \frac{1 - y_{ki}y_{kj}}{2},\ k < i < j, \qquad \xi_{ijk} = \frac{1 - y_{ik}y_{jk}}{2},\ i < j < k, \tag{6.10}$$

for $i, j, k \in \mathcal{N}$. Using (6.10) we can easily rewrite the objective function (6.3) and equalities (6.4) in terms of ordering variables

$$K - \sum_{\substack{i,j \in \mathcal{N} \\ i<j}} \frac{c_{ij}}{2} \left(\sum_{\substack{k \in \mathcal{N} \\ k<i}} l_k y_{ki} y_{kj} - \sum_{\substack{k \in \mathcal{N} \\ i<k<j}} l_k y_{ik} y_{kj} + \sum_{\substack{k \in \mathcal{N} \\ k>j}} l_k y_{ik} y_{jk} \right), \tag{6.11}$$

$$y_{ij}y_{jk} - y_{ij}y_{ik} - y_{ik}y_{jk} = -1, \quad i, j, k \in \mathcal{N}, i < j < k, \tag{6.12}$$

where $K := \left(\sum_{\substack{i,j \in \mathcal{N} \\ i<j}} \frac{c_{ij}}{2} \right) \left(\sum_{k \in \mathcal{N}} l_k \right)$. We have already deduced equations (6.12) in a different way in Section 4.3. In [31] it is shown that these equalities describe the smallest linear subspace that contains the quadratic ordering polytope

$$\mathcal{P}_{QO} := \text{conv} \{\ yy^\top\ :\ y \in \{-1, 1\}^{\binom{n}{2}},\ |y_{ij} + y_{jk} - y_{ik}| = 1\ \}.$$

To obtain matrix-based relaxations of $\mathcal{P}_{QO}$ we collect the ordering variables in a vector y and consider the matrix $Y = yy^\top$. The main diagonal entries of Y correspond to y_{ij}^2 and hence $\text{diag}(Y) = e$, the vector of all ones. Now we can formulate (SRFLP) as the following optimization problem, first proposed in [10]

$$\min \{\ \langle C, Y \rangle + K :\ Y \text{ satisfies (6.12)},\ \text{diag}(Y) = e,\ \text{rank}(Y) = 1,\ Y \succcurlyeq 0\ \}, \tag{SRFLP}$$

where the cost matrix C is deduced from (6.11). Dropping the rank constraint yields the basic semidefinite relaxation of (SRFLP)

$$\min \{\ \langle C, Y \rangle + K :\ Y \text{ satisfies (6.12)},\ \text{diag}(Y) = e,\ Y \succcurlyeq 0\ \}, \tag{$\mathtt{A}_1$}$$

providing a lower bound on the optimal value of (SRFLP). To be able to tackle larger instances Anjos and Yen [14] proposed to sum up the $O(n^3)$ constraints (6.12) over k yielding the $O(n^2)$ constraints

$$\sum_{\substack{k \in \mathcal{N} \\ i \neq k \neq j}} y_{ij}y_{jk} - \sum_{\substack{k \in \mathcal{N} \\ i \neq k \neq j}} y_{ij}y_{ik} - \sum_{\substack{k \in \mathcal{N} \\ i \neq k \neq j}} y_{ik}y_{jk} = -(n-2), \quad i, j \in \mathcal{N}, i < j. \tag{6.13}$$

They showed that the following optimization problem using (6.13) instead of (6.12)

$$\min \{\ \langle C, Y \rangle + K :\ Y \text{ satisfies (6.13)},\ \text{diag}(Y) = e,\ \text{rank}(Y) = 1,\ Y \succcurlyeq 0\ \},$$

is again an exact formulation of (SRFLP). Dropping the rank-one constraint yields a weaker but also cheaper semidefinite relaxation than ($\mathtt{A}_1$)

$$\min \{\ \langle C, Y \rangle + K :\ Y \text{ satisfies (6.13)},\ \text{diag}(Y) = e,\ Y \succcurlyeq 0\ \}. \tag{$\mathtt{A}_0$}$$

As Y is actually a matrix with $\{-1, 1\}$ entries in the original (SRFLP) formulation, Anjos and Vanelli [13] proposed to further tighten ($\mathtt{A}_1$) by adding the triangle inequalities known to be facet-defining for the cut polytope, see e.g. [62]

$$\left\{ Y : \begin{pmatrix} -1 & -1 & -1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} Y_{i,j} \\ Y_{j,k} \\ Y_{i,k} \end{pmatrix} \leq e,\ 1 \leq i < j < k \leq \binom{n}{2} \right\}. \tag{6.14}$$

Using the transformations (6.10) it is straightforward to show the equivalence of a subset of the triangle inequalities with the betweenness constraints (6.6) and (6.7) from above. Along the same lines inequalities (6.8) can be connected to general clique inequalities. Adding the triangle inequalities to $(\mathtt{A}_1)$, Anjos and Vanelli achieved the following relaxation of (SRFLP)

$$\min \{ \langle C, Y \rangle + K : \ Y \text{ satisfies (6.12) and (6.14)}, \ \operatorname{diag}(Y) = e, \ Y \succcurlyeq 0 \}. \qquad (\mathtt{A}_2)$$

As solving $(\mathtt{A}_2)$ directly with an interior-point solver like CSDP [24, 146] gets far too expensive, they suggest to use the $\approx \frac{1}{12}n^6$ triangle inequalities as cutting planes in their algorithmic framework (for further details on the practical realization of this approach see Section 12.1).

Notice that Anjos et al. work with the quadratic ordering polytope $\mathcal{P}_{QO}$, whereas we formulate our semidefinite programs on the linear-quadratic ordering polytope $\mathcal{P}_{LQO}$. As $Z(y, Y) := \begin{pmatrix} 1 & y^T \\ y & Y \end{pmatrix} \succcurlyeq 0$ is in general a stronger constraint than $Y \succcurlyeq 0$, the SDP relaxations $(\mathtt{A}_1)$ and $(\mathtt{A}_2)$ are slightly weaker than our corresponding relaxations $(\mathtt{SDP}_1)$ and $(\mathtt{SDP}_2)$ with C and K defined above and c equal to the zero vector.

Let us also mention that so far all SDP approaches to (SRFLP) refrained from using other clique inequalities to further tighten the SDP relaxations because of their large number. We will argue in Section 7.2 that using well-designed subsets of larger clique inequalities, like e.g. pentagonal inequalities, which can be connected to the betweenness constraints $(6.8)_{\beta=6}$, could be a promising direction to improve current SDP approaches. Another future research intent is to give further theoretical results concerning the tightness of our SDP relaxations, e.g. by investigating if $(\mathtt{SDP}_4)$ is exact for many special types of rank inequalities used for separation in most linear programming based approaches (for further details see Sections 5.2 and 5.3).

Chapter 7

The Quadratic Ordering Problem

7.1 Introduction

The Quadratic Ordering Problem (QOP) asks to find an ordering of the objects $\mathcal{N} := \{1, \dots, n\}$ with maximum profit, where the profit depends on whether object s comes before object t and object u comes before object v in the ordering. Allowing arbitrary C, c and K in

$$\max \left\{ \langle C, Y \rangle + c^\top y + K : Z \text{ partitioned as in (4.8) satisfies (4.11)}, Z \in \mathcal{E}, y \in \{-1,1\}^{\binom{n}{2}} \right\}$$

models (QOP) (see Theorem 4.1). Thus using semidefinite optimization to solve (QOP) is a very natural approach. The Linear Ordering Problem (LOP) as well as the minimum Linear Arrangement (minLA) and the Single Row Facility Layout Problem (SRFLP) are special cases of (QOP). The weighted Betweenness Problem (wBP) is also a particular (QOP) type, but contains (minLA) and (SRFLP) as special cases. An input to (wBP) consists of n objects, a set $\mathcal{B}$ of betweenness conditions and a set $\overline{\mathcal{B}}$ of non-betweenness conditions ($\mathcal{B} \cap \overline{\mathcal{B}} = \emptyset$). The elements of $\mathcal{B}$ and $\overline{\mathcal{B}}$ are triples (i, j, k) with associated costs w_{ijk} for not placing respectively placing object j between objects i and k. Now the task in (wBP) is to find an ordering of the objects such that the sum of costs is minimized. Thus we can model (wBP) as an ILP in betweenness variables (6.2). Then using relation (6.10) we can formulate (wBP) as an (QOP), where c is the zero vector and the cost matrix C has at most $O(n^3)$ entries. Another special case of the (wBP) is the Physical Mapping Problem with End Probes (PMP) from computational biology (for details on the biological background see e.g. the book of Brown [29]). Christof et al. [47] formulated (PMP) as (wBP) with $\approx \frac{n^2}{2}$ betweenness conditions and designed an ILP Branch-and-Cut algorithm using linear ordering and betweenness variables to solve the problem to optimality. Later on Christof et al. [48] showed that (PMP) can be reformulated as an equivalent Consecutive Ones Problem (COP). By solving (COP) with a Branch-and-Cut algorithm, they obtained the strongest computational results for (PMP) so far (for a detailed polyhedral study and further applications of (COP) see the PhD thesis of Oswald [168]). As the (COP) Branch-and-Cut approach only works well for $n \leq 60$ (see [168, Table 7.1]), it would be interesting to apply our SDP approach also to (PMP), because it may yield computational progress for large-scale instances.

Going from linear to quadratic objective functions usually makes an optimization problem much harder. For example the binary maximization of a linear function over the hypercube, which is trivial, becomes the Max-Cut Problem (MC) [197] and thus NP-hard. In our case (LOP) is already NP-hard, nonetheless the practical hardness of (QOP) is significantly higher and classical approaches used for (LOP) do not work at all for (QOP). While there exist quite diverse ILP approaches for the different ordering problems, for the semidefinite approach the linear and quadratic variants of the problem are essentially equally hard to solve. There also exists an ILP-based Branch-and-Cut algorithm for (QOP) designed by Buchheim et al. [30]. Their ILPs result from the linearization of the above semidefinite formulation of (QOP). While

our semidefinite approach can obtain reasonable bounds for $n \leq 100$ objects, their approach is restricted to problems with $n \leq 16$ objects (for details see [30, Table 2]). Although their approach leaves many degrees of freedom and a lot of room for improvement by tuning various parameters, their restriction in size already showcases that the semidefinite approach is preferable for `(QOP)`. This is also supported by the following observation: in general QOP induces a more complex cost structure compared to its special cases discussed in the previous chapters. Thus `(QOP)` needs more of the implicitly defined SDP variables. But the more SDP variables and therefore SDP structure is needed, the better the SDP performs compared to competing ILP approaches (in the context of ordering problems). And already for `(SRFLP)`, our SDP approach is clearly the method of choice.

The tightness of the bounds obtained by semidefinite relaxations of course differs with respect to the complexity of the cost structure of the particular problem type. While our semidefinite relaxations suffice to provide an optimality certificate for most `(minLA)` (and also Multi-level Crossing Minimization `(MLCM)`) instances with up to 70 objects, they cannot close the gaps for some `(QOP)` instances (and also some Multi-level Verticality Optimization `(MLVO)` instances) with only 20 objects. Hence further improving the tightness of the semidefinite relaxations for the more difficult problem types, of course without making them incomputable, seems to be a worthwhile research direction. In the following two sections we propose several strategies for tightening (SDP_4). In Section 7.2 we analyse the complete outer description of different ordering polytopes in small dimensions to evaluate and improve our semidefinite relaxations. We detect several constraint types with a reasonable total number of constraints ($\leq O(n^6)$) that can be used to tighten (SDP_4). In Section 7.3 we propose a heuristic method to select important inequalities from a constraint set that is too large ($\geq O(n^7)$) to be considered as a whole. In Chapter 13 we provide promising preliminary computational results for our tightening strategies.

7.2 Ordering Polytopes in Small Dimensions

In this section we analyse the facet types of the betweenness polytope $\mathcal{P}^n_{BTW}$ and the linear-quadratic ordering polytope $\mathcal{P}^n_{LQO}$ for small n. Using PORTA [46] it is possible to compute the complete outer description of these polytopes in small dimensions. Analysing the complete outer description is known to be a good strategy to evaluate and also improve the quality of relaxations. Thus we apply this approach to find further strong constraint types to tighten (SDP_4) without making it incomputable.

We start with computing the complete outer description of $\mathcal{P}^3_{LQO}$. This polytope has dimension 6 and is defined by the equations (4.11) and 6 facets which are all of the following type

$$(1 \pm y_{ij})(1 \pm y_{kl}) \geq 0,\ i < j, k < l \in \mathcal{N}, \tag{7.1}$$

and thus included in (4.12). Next let us take a look at $\mathcal{P}^4_{LQO}$. This polytope has dimension 21 and is defined by (4.11) together with 126 facets consisting of 4 types. First there are 48 triangle inequalities (4.12) (composed of 36 constraints of type (7.1) and 12 constraints on Y). The second class contains 48 Lovász-Schrijver-cuts (4.13). The generic approach proposed by Lovász and Schrijver [148] can also be applied to pairs of 3-cycle inequalities (4.6b) yielding the following $\approx \frac{n^6}{9}$ constraints

$$\begin{aligned}
-1 - y_{ij} - y_{jk} + y_{ik} \leq y_{lm} + y_{mo} - y_{lo} + y_{ij,lm} + y_{ij,mo} - y_{ij,lo} + y_{jk,lm} + y_{jk,mo} - y_{jk,lo} - \\
y_{ik,lm} - y_{ik,mo} + y_{ik,lo} \leq 1 + y_{ij} + y_{jk} - y_{ik},\ i < j < k \in \mathcal{N},\ l < m < o \in \mathcal{N}, \\
-1 + y_{ij} + y_{jk} - y_{ik} \leq y_{lm} + y_{mo} - y_{lo} - y_{ij,lm} - y_{ij,mo} + y_{ij,lo} - y_{jk,lm} - y_{jk,mo} + y_{jk,lo} + \\
y_{ik,lm} + y_{ik,mo} - y_{ik,lo} \leq 1 - y_{ij} - y_{jk} + y_{ik},\ i < j < k \in \mathcal{N},\ l < m < o \in \mathcal{N},
\end{aligned} \tag{7.2}$$

where 6 of them are facets of $\mathcal{P}^4_{LQO}$. Finally there is an additional constraint class containing 24 facets of more complicated structure. For $n \geq 5$ the outer description of $\mathcal{P}^n_{LQO}$ cannot be obtained within reasonable time by PORTA.

Using the above deduced constraints (7.2) we can further strengthen (SDP$_4$)

$$\max\left\{\langle C, Y\rangle + c^\top y + K : Z \text{ partitioned as in (4.8) satisfies (4.11) and (7.2)}, Z \in \mathcal{E} \cap \mathcal{M} \cap \mathcal{LS}\right\}. \quad (\text{SDP}_5)$$

The multi-level quadratic ordering polytope $\mathcal{P}_{MQO}$ (a definition can be found in the following chapter) could be analyzed in a similar way. It is an open question whether the constraint types (4.13) and (7.2) (with mutually disjoint indices) are facet defining for any $\mathcal{P}_{LQO}$ with dimension $n \geq 4$. It would also be interesting to examine the 24 more complicated facets of $\mathcal{P}^4_{LQO}$ in more detail and to incorporate them in our SDP approach.

Further strong valid inequalities for $\mathcal{P}_{LQO}$ can be obtained by investigating the betweenness polytope $\mathcal{P}_{BTW}$ for small dimension. Oswald [168] computes the complete outer description of $\mathcal{P}^n_{BTW}$ for $n \in \{3, 4, 5\}$ with the help of PORTA and also gives graph representations of all facets in normal form for illustration. In the following we state these facets by reusing the betweenness variables (6.2) from the previous chapter. Let us further recall that $\mathcal{P}_{BTW}$ lies in the subspace defined by the equations (6.4). For $\mathcal{P}^3_{BTW}$, the only facets are given by $\xi \geq 0$. For $\mathcal{P}^4_{BTW}$, we have the following facets for all permutations of $\{1, 2, 3, 4\}$:

$$\xi_{132} + \xi_{123} + \xi_{241} + \xi_{341} \geq -2. \quad (\text{F1})$$

Note, that the trivial bounds $\xi \geq 0$ are no longer facet-defining. The facets of $\mathcal{P}^5_{BTW}$ consist of the facets of $\mathcal{P}^4_{BTW}$ and the following constraints for all permutations of $\{1, 2, 3, 4, 5\}$:

$$\xi_{123} + \xi_{145} + \xi_{254} + \xi_{351} + \xi_{342} \geq -3 \quad (\text{F2})$$

$$\xi_{152} + \xi_{123} + \xi_{143} + \xi_{153} + \xi_{124} + \xi_{145} + \xi_{231} + \xi_{241} + \xi_{251} + \xi_{234} + \xi_{352} + \xi_{345} \geq -6 \quad (\text{F3})$$

$$\xi_{152} + \xi_{123} + \xi_{134} + \xi_{135} + \xi_{231} + \xi_{241} + \xi_{251} + \xi_{243} + \xi_{234} + \xi_{354} + \xi_{345} + \xi_{452} \geq -6 \quad (\text{F4})$$

$$\xi_{123} + \xi_{143} + \xi_{153} + \xi_{124} + \xi_{134} + \xi_{154} + \xi_{135} + \xi_{145} + \xi_{231} + \xi_{241} + \xi_{352} + \xi_{452} \geq -6 \quad (\text{F5})$$

$$\xi_{132} + \xi_{152} + \xi_{123} + \xi_{134} + \xi_{154} + \xi_{145} + \xi_{241} + \xi_{251} + \xi_{235} + \xi_{341} + \xi_{453} \geq -5 \quad (\text{F6})$$

$$\xi_{152} + \xi_{123} + \xi_{143} + \xi_{153} + \xi_{134} + \xi_{145} + \xi_{231} + \xi_{241} + \xi_{251} + \xi_{234} + \xi_{245} + \xi_{352} + \xi_{354} + \xi_{345} + \xi_{452} \geq -7 \quad (\text{F7})$$

$$\xi_{152} + \xi_{123} + \xi_{143} + \xi_{153} + \xi_{124} + \xi_{134} + \xi_{154} + \xi_{135} + \xi_{251} + \xi_{231} + \xi_{241} + \xi_{235} + \xi_{352} + \xi_{345} + \xi_{452} \geq -7 \quad (\text{F8})$$

$$\xi_{142} + \xi_{152} + \xi_{123} + \xi_{143} + \xi_{134} + \xi_{125} + \xi_{231} + \xi_{241} + \xi_{234} + \xi_{254} + \xi_{351} + \xi_{342} + \xi_{345} + \xi_{453} \geq -6 \quad (\text{F9})$$

$$\xi_{142} + \xi_{152} + \xi_{123} + \xi_{143} + \xi_{124} + \xi_{135} + \xi_{231} + \xi_{251} + \xi_{234} + \xi_{245} + \xi_{341} + \xi_{342} + \xi_{354} + \xi_{453} \geq -6 \quad (\text{F10})$$

$$\xi_{132} + \xi_{152} + \xi_{123} + \xi_{143} + \xi_{124} + \xi_{154} + \xi_{135} + \xi_{145} + \xi_{241} + \xi_{251} + \xi_{253} + \xi_{234} + \xi_{235} + \xi_{245} + \xi_{341} + \xi_{351} + \xi_{342} + \xi_{354} + \xi_{452} + \xi_{453} \geq -8 \quad (\text{F11})$$

Rewriting (F1)–(F11) with the help of (6.10) yields quadratic constraints in bivalent ordering variables (6.9). By computation (for details see Section 13.1) we can show that (SDP$_5$) is exact only for (F1) and (F2). Thus we can strengthen (SDP$_5$) by adding the $9\binom{n}{5}$ inequalities (F3)–(F11) to the relaxation. First computational experiments indicate that adding these constraints pays off for all kinds of betweenness

instances. Adding the $\approx \frac{1}{12}n^6$ betweenness inequalities $(6.8)_{\beta=6}$, which are a subset of the $\frac{1}{240}n^{10}$ pentagonal inequalities (7.4) defined below, seems to be another interesting option for further tightening (SDP$_5$).

Finally we take a look at a further polytope that is closely related to $\mathcal{P}_{BTW}$. Let us introduce the triple variables ψ_{ijk} for $i, j, k \in \mathcal{N}$

$$\psi_{ijk} = \begin{cases} 1, & \text{if object } i \text{ comes before object } j \text{ and object } j \text{ comes before object } k, \\ 0, & \text{otherwise.} \end{cases}$$

We collect the above variables in a vector ψ and define the triple polytope

$$\mathcal{P}_{TRI} := \text{conv}\,\{\psi \ : \ \psi \text{ represents a ordering of the elements of } \mathcal{N}\}.$$

On the one hand the triple variables can be used to express the ordering and the betweenness variables

$$y_{ij} = 2(\psi_{ijk} + \psi_{ikj} + \psi_{kij}) - 1,\ i, j, k \in \mathcal{N},\ i < j, \qquad \xi_{ikj} = \psi_{ijk} + \psi_{kji},\ i, j, k \in \mathcal{N},\ i < k,$$

and on the other hand the triple variables can be expressed via products of ordering variables

$$\begin{aligned} \psi_{ijk} &= \frac{y_{ij}y_{jk} + y_{ij} + y_{jk} + 1}{4},\ i < j < k \in \mathcal{N}, \quad \psi_{ijk} = \frac{-y_{ij}y_{jk} + y_{ij} - y_{jk} + 1}{4},\ i, j, k \in \mathcal{N},\ i < j,\ k < j, \\ \psi_{ijk} &= \frac{y_{ij}y_{jk} - y_{ij} - y_{jk} + 1}{4},\ i > j > k \in \mathcal{N}, \quad \psi_{ijk} = \frac{-y_{ij}y_{jk} - y_{ij} + y_{jk} + 1}{4},\ i, j, k \in \mathcal{N},\ j < i,\ j < k. \end{aligned} \tag{7.3}$$

The complete outer description of $\mathcal{P}^n_{TRI}$ can be computed with PORTA for $n \leq 4$. For $n = 4$ we get the following facets for all permutations of $\{1, 2, 3, 4\}$:

$$\begin{aligned} \psi_{123} - \psi_{124} - \psi_{423} &\leq 1, \\ \psi_{123} - \psi_{124} - \psi_{413} - \psi_{143} &\leq 2, \\ \psi_{123} - \psi_{134} - \psi_{423} - \psi_{143} &\leq 2, \\ \psi_{123} + \psi_{432} - \psi_{134} - \psi_{143} - \psi_{421} - \psi_{412} &\leq 2, \end{aligned}$$

Using (7.3) we can rewrite the facet types of $\mathcal{P}^4_{TRI}$ as quadratic constraints in ordering variables. It is easy to show that (SDP$_4$) is exact for all these facet types, as the first inequality reduces to a triangle inequality when written in ordering variables and the other three inequalities belong to (4.13).

Using the triple variables with all facets of $\mathcal{P}^4_{TRI}$ and a part of the facets of $\mathcal{P}^5_{TRI}$, Oswald [169] obtains the best known results for several linear ordering instances from the literature.

It would be interesting to analyze the merits and computational costs of including the different constraint types discussed in this section in our semidefinite relaxations.

7.3 Heuristic Constraint Selection

In this section we want to propose a heuristic approach to find further strengthening constraints for our semidefinite relaxations. We try to detect the most important pentagonal inequalities (clique inequalities of length 5) dynamically, as separating all of them is computationally far too expensive. Thereby the most promising pentagonal inequalities are those that contain a triple of indices belonging to an (almost) violated triangle inequality (4.6b) at the current point.

The method applies to all problems formulated over the linear-quadratic ordering polytope $\mathcal{P}_{LQO}$ (see previous chapters) as well as for problems associated to the multi-level quadratic ordering polytope $\mathcal{P}_{MQO}$ (see next chapters). The pentagonal inequalities are valid for these polytopes as they are known to be facet defining [58] for the cut polytope (with $\binom{n}{2}$ vertices)

$$\mathcal{P}_C := \text{conv}\,\{\ yy^\top \ : \ y \in \{-1, 1\}^{\binom{n}{2}}\},$$

which contains $\mathcal{P}_{LQO}$ and $\mathcal{P}_{MQO}$ as faces. Thus our heuristic approach can also be used to further improve the standard relaxation for the Max-Cut problem (MC) (for general information and references for (MC) see Section 2.3). For indicating the possible merits and computational costs of the proposed heuristic constraint selection we apply the method to several well-known (MC) instances from the literature in Section 13.2.

Along the lines of Theorem 4.1 we can formulate (MC) as the following semidefinite optimization problem

$$\max \{ \langle C, Z \rangle : Z \text{ satisfies (4.8)}, Z \in \mathcal{E}, y \in \{-1,1\}^{\binom{n}{2}} \}. \tag{MC}$$

Dropping the integrality condition on y, we obtain a basic semidefinite relaxation for (MC)

$$\max \{ \langle C, Z \rangle : Z \text{ satisfies (4.8)}, Z \in \mathcal{E} \}. \tag{MC$_1$}$$

We can further strengthen (MC$_1$) by asking Z to satisfy the triangle inequalities (4.12) that are valid for $\mathcal{P}_C$ (the triangle inequalities 2, 3 and 4 of (4.12) are even facet defining for $\mathcal{P}_C$ [18]). This yields the standard semidefinite relaxation for (MC), first proposed by Poljak and Rendl [175].

$$\max \{ \langle C, Z \rangle : Z \text{ satisfies (4.8)}, Z \in (\mathcal{E} \cap \mathcal{M}) \}. \tag{MC$_2$}$$

Recently Rendl et al. [184] presented a highly competitive SDP Branch-and-Bound algorithm building on (MC$_2$). Their approach uses a dynamic version of the bundle method (for details see Section 3.3) to approximately solve (MC$_2$) and nearly always, especially on dense graphs, outperforms all other approaches for (MC). The main limitation of this approach is the quality of the upper bounds obtained from (MC$_2$). Thus tightening (MC$_2$) without making it incomputable should lead to a stronger exact method for (MC). Hence we consider the $16\binom{\zeta}{5} \approx \frac{1}{240}n^{10}$ pentagonal inequalities known to be valid (and except the first inequality also facet defining [18]) for $\mathcal{P}_C$

$$\mathcal{H} = \{ Z : C_5 z_5 \leq 2e,\ 1 \leq i < j < k < l < m \leq \zeta \}, \tag{7.4}$$

where

$$C_5 = \begin{pmatrix}
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 \\
-1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 \\
-1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 \\
-1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 \\
-1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 \\
1 & -1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\
1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 \\
1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 \\
1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 \\
-1 & 1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\
-1 & 1 & -1 & 1 & 1 & -1 & 1 & 1 & -1 & 1 \\
-1 & -1 & 1 & 1 & -1 & 1 & 1 & 1 & 1 & -1
\end{pmatrix} \quad \text{and} \quad z_5 = \begin{pmatrix} z_{ij} \\ z_{ik} \\ z_{il} \\ z_{im} \\ z_{jk} \\ z_{jl} \\ z_{jm} \\ z_{kl} \\ z_{km} \\ z_{lm} \end{pmatrix}.$$

The pentagonal inequalities can be explained with the help of the complete graph with 5 nodes. The nodes can take the values $+1$ or -1 and the weights of the edges are given as the product of the values of their end nodes. Then the pentagonal inequalities state valid bounds on the sum of the edges weights independent of the node values.

Notice that the set $\mathcal{M} \cap \mathcal{H}$ contains exactly the facets of the cut polytope with 5 nodes (for details see `http://comopt.ifi.uni-heidelberg.de/software/SMAPO/cut/cut.html`).

The standard SDP relaxation (MC$_2$) can therefore be improved by asking in addition that $Z \in \mathcal{H}$.

$$\max \{ \langle C, Z\rangle : Z \text{ satisfies (4.8)}, Z \in (\mathcal{E} \cap \mathcal{M} \cap \mathcal{H}) \}. \qquad (\texttt{MC}_4)$$

Even with the dynamic version of the bundle method (for details see Section 3.3) we cannot solve (MC$_4$) efficiently for large $\zeta \approx 500$. Hence we propose to choose only a subset $\mathcal{H}_t$ of $\mathcal{H}$ of order $0(n^6)$ that contains the most promising constraints. Therefore we employ the following heuristic idea. The most promising inequalities in $\mathcal{H}$ are those that contain a triple of nodes defining an at least (almost) violated triangle inequality (4.6b) at the current iterate. Thus we formally define

$$\mathcal{H}_t = \{ Z : C_5 z_5 \leq 2e,\ (i,j,k) \in \mathcal{M}_t,\ 1 \leq l < m \leq \zeta \}.$$

where

$$\mathcal{M}_t = \left\{ (i,j,k) : \begin{pmatrix} -1 & -1 & -1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} z_{i,j} \\ z_{j,k} \\ z_{i,k} \end{pmatrix} \geq (1-t)e,\ 1 \leq i < j < k \leq \zeta \right\},$$

and $t \in [-2, 4]$. We propose to choose t such that $\mathcal{M}_t$ contains $O(n^3)$ triples and hence $O(n^6)$ pentagonal inequalities are evaluated in every iteration. In summary we get the following semidefinite relaxation for (MC) that is tighter than (MC$_2$) and can be computed more efficiently than (MC$_4$).

$$\max \{ \langle C, Z\rangle : Z \text{ satisfies (4.8)}, Z \in (\mathcal{E} \cap \mathcal{M} \cap \mathcal{H}_t) \}. \qquad (\texttt{MC}_3)$$

For a preliminary experimental study applying (MC$_3$) to well-known instances from the literature see Section 13.2.

A more general class of valid constraints for $\mathcal{P}_C$ are the hypermetric inequalities, introduced by Deza [57] (and later, independently by Kelly [128]) and studied in detail by Deza and Laurent [58, 59]. Let b be an integer vector with $\sigma = \sum_{i=1}^n b_i = 1$, then

$$\sum_{1 \leq i < j \leq n} b_i b_j z_{ij} \leq 0,$$

are called hypermetric inequalities. Allowing $\sigma = \sum_{i=1}^n b_i$ odd, yields the more general non homogeneous hypermetric inequalities

$$\sum_{1 \leq i < j \leq n} b_i b_j z_{ij} \leq \frac{\sigma^2 - 1}{4},$$

containing the triangle, pentagonal and hypermetric inequalities. We plan to generalize the heuristic constraint selection to further constraint types (of small length) of the non homogeneous hypermetric inequalities or even to more general classes of gap or clique-web inequalities (for details on these constraint classes see Deza and Laurent [60, 61]). Of course the method can also be applied to appropriate constraint classes of other polytopes.

Finally we want to provide some further theoretical support for the relaxations defined above. Thus we examine the tightness of the upper bounds provided by (MC$_2$) for complete graphs and cycles. Similar results (but for many more types of graphs) have been deduced (using a different approach) by Delorme and Poljak [55] for relaxation (MC$_1$).

Observation 7.1 *Let C_q be a cycle and K_q be the complete graph. The following inequalities are valid for the cut polytope $\mathcal{P}_C$:*

$$\sum_{(i,j) \in C_q} -Y_{i,j} \leq \begin{cases} q & q \text{ even}, \\ q-2 & q \text{ odd}, \end{cases}, \qquad \sum_{(i,j) \in K_q} -Y_{i,j} \leq \begin{cases} \frac{q}{2} & q \text{ even}, \\ \frac{q-1}{2} & q \text{ odd}. \end{cases} \qquad (7.5), (7.6)$$

(MC$_2$) *is exact for* (7.5) *and for* (7.6) *with $q = 3$ or even. Furthermore* (MC$_2$) *yields a (maximum) violation of $\frac{1}{2}$ for* (7.6) *with $q \geq 5$ and odd.*

Proof. Without loss of generality we assume that there is an edge in C_q iff $j - i \mod q = 1,\ i < j \leq \zeta$. Then $(\mathtt{MC}_2)$ is exact for (7.5) with q even due to $Z \in \mathcal{E}$, as $Z \in \mathcal{E}$ implies $-1 \leq Y_{i,j} \leq 1,\ i < j \leq \zeta$.

$(\mathtt{MC}_2)$ is also exact for (7.5) with q odd due to type 1 and type 2 of (4.12): For $q = 3$, (7.5) is trivially satisfied by type 1 of (4.12). For $q \geq 5$ odd we get (7.5) by summing up the following inequalities of type 1 and 2 of (4.12)

$$\begin{aligned} -Y_{1,2} - Y_{1,3} - Y_{2,3} &\leq 1, \\ -Y_{3,4} - Y_{3,5} - Y_{4,5} &\leq 1, \\ Y_{1,3} - Y_{1,5} + Y_{3,5} &\leq 1, \\ &\vdots \\ -Y_{q-2,q-1} - Y_{q-2,q} - Y_{q-1,q} &\leq 1, \\ Y_{1,q-2} - Y_{1,q} + Y_{q-2,q} &\leq 1. \end{aligned}$$

$(\mathtt{MC}_2)$ satisfies (7.6) for q even and violates (7.6) by $\frac{1}{2}$ for q odd due to $Z \in \mathcal{E}$: Using $e^\top Y e \geq 0$ and $Y_{i,i} = 1,\ 1 \leq i \leq \zeta$ yields

$$2 \sum_{1 \leq i < j \leq q} Y_{i,j} \geq -q,$$

and thus

$$\sum_{(i,j) \in K_q} -Y_{i,j} \leq \frac{q}{2}.$$

Furthermore $(\mathtt{MC}_2)$ is exact for (7.6) with $q = 3$ due to type 1 of (4.12) and $(\mathtt{MC}_4)$ is additionally exact for (7.6) with $q = 5$ due to type 1 of (7.4). □

In [54] Delorme and Poljak also prove that $(\mathtt{MC}_1)$ is never worse than 1.131*$(\mathtt{MC})$ for planar, or more generally, weakly bipartite graphs with nonnegative edge weights and conjecture that this value, which is attained for C_5, might be true for any graph G. This conjecture was "almost" confirmed by the result of Goemans and Williamson [86] who proved that $(\mathtt{MC}_1)$ is never worse than 1.138*$(\mathtt{MC})$ for any graph G and any nonnegative edge weights w.

Poljak et al. [138] conjectured that the worst case of the ratio $(\mathtt{MC}_2)/\ (\mathtt{MC})$ is attained for K_5 with 1.0417. Later on Rendl [182] came up with a Toeplitz graph with the following adjacency matrix

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

that yields a $(\mathtt{MC}_2)/\ (\mathtt{MC})$ ratio of 1.0510. This graph also provides the worst $(\mathtt{MC}_4)/\ (\mathtt{MC})$ ratio that we found, namely 1.0312. Karfloff [123] showed that it is impossible to add valid linear constraints to improve the worst case performance ratio of the algorithm of Goemans and Williamson [86] and there do not exist any other theoretical bounds on $(\mathtt{MC}_2)/\ (\mathtt{MC})$ or $(\mathtt{MC}_4)/\ (\mathtt{MC})$ to date.

It would also be interesting to investigate $(\mathtt{MC}_2)$ and $(\mathtt{MC}_4)$ for further graph types, like e.g. the (generalized) Petersen graphs, (bicycle) wheels, Möbius ladders and also triangles and cycles with different weights.

Chapter 8

Multi-level Crossing Minimization

8.1 Introduction

Multi-level Crossing Minimization (MLCM) is defined as follows. Let us consider a proper level graph $G = (V, E)$, with vertex set $V = \dot{\bigcup}_{r=1}^{p} V_r$ and edge set $E = \dot{\bigcup}_{r=1}^{p-1} E_r$ with $E_r \subseteq V_r \times V_{r+1}$. We ask for an ordering of the vertices (when drawn on their respective level) such that the number of crossings between straight-line edges is minimized.

(MLCM) is an important task in automatic graph drawing. Hierarchical graphs (e.g. graphs representing schedules, UML diagrams and flow charts) are mostly drawn with the framework suggested by Sugiyama [200]. Here, the vertices are assigned to levels (corresponding to horizontal layers), essentially fixing the y-coordinates of the vertices. Then, the graph is transformed into a proper multi-level graph in which edges are subdivided such that each edge connects two vertices on adjacent layers. The aim of the multi-level crossing minimization step is to reorder the vertices within the levels so that the number of crossings is minimized when the edges are drawn as straight lines. Finally, the position of the vertices are assigned while keeping the leveling and the ordering of the vertices. An alternative paradigm to Sugiyama's approach is based on upward planarization [40]. Also in this setting, finding optimal solutions of (MLCM) is of interest (see Section 14.3).

In practice, (MLCM) is often reduced to a sequence of 2-level crossing minimization problems in which one level is fixed. Many heuristics [151, 200] as well as FPT (Fixed Parameter Tractable) algorithms [68] have been suggested for this restricted problem, but so far variants of the simple barycenter and median heuristics belong to the best in practice [15, 120, 151]. General (MLCM) is NP-hard, even in this restricted variant [69]. Jünger and Mutzel [120] have shown that this restricted problem can be reduced to a Linear Ordering Problem that can be solved using an integer linear programming (ILP) approach. Combined with a Branch-and-Bound method, they solved 2-level (MLCM) instances with up to 15 vertices on the smaller level to optimality. An alternative exact approach for solving 2-level (MLCM), based on SDP, has been suggested by Buchheim et al. [31]. They model the problem as a Quadratic Ordering Problem, deduce the SDP relaxation (SDP_2) (see Section 4.3) and suggest new ILP- and SDP-based algorithms. Their experiments show that the SDP-based Branch-and-Bound algorithm outperforms various versions of ILP-based Branch-and-Cut algorithms, and is able to solve 2-level instances with up to 18 vertices per level to optimality within reasonable computing time.

The first ILP formulation for general (MLCM) has been suggested by Jünger et al. [119]. At that time, generic ILP solvers have not been able to solve practically relevant instances. Healy and Kuusik [96] extended the ILP formulation by constraints arising from the so-called vertex-exchange graph. For the first time they have been able to solve some practically relevant instances of (MLCM) to optimality [97].

The remainder of this chapter is based on Sections 2 and 3 of the paper "An SDP Approach to Multi-level Crossing Minimization" [45]. In Section 8.2 we present a binary linear model for (MLCM) and recall the

linear programming based approaches building on that model. Next we present an SDP-based approach for (MLCM) in Section 8.3. Finally in Section 8.4 we show the polyhedral advantages of the SDP approach over the linear programming based approaches. Furthermore, our investigations of the facial structure of the related polyhedron justify the choice of the considered semidefinite relaxation.

8.2 Exact Approaches Based on Linear Programming

(MLCM) has a natural formulation as a quadratic linear program in 0-1 variables [119]. Similar to the approaches from the previous chapters, we can model the node order by introducing binary ordering variables, assuming some fixed total order $\dot{<}$ of the nodes (e.g. based on their indices)

$$x_{uv} \in \{0,1\}, \qquad \forall u,v \in V_i,\ 1 \le i \le p,\ u \dot{<} v. \tag{8.1}$$

The variables shall be 1 if u is left of v and 0 otherwise. For notational simplicity, we also use the shorthand $x_{uv} := 1 - x_{vu}$ for $v \dot{<} u$. We have already argued in Section 4.2 that feasible orderings can be described via 3-cycle inequalities

$$0 \le x_{uv} + x_{vw} - x_{uw} \le 1, \qquad \forall u,v,w \in V_i,\ 1 \le i \le p,\ u \dot{<} v \dot{<} w. \tag{8.2}$$

Let us denote by $N(v)$ the set of vertices on level $i+1$ that are the destinations of edges adjacent to vertex v on level i. Now minimizing

$$\sum_{1 \le i < p} \sum_{\substack{s,t \in V_i,\\ s \dot{<} t}} \sum_{\substack{u \in N(s)\\ v \in N(t)}} (x_{st}x_{vu} + x_{ts}x_{uv}) \tag{8.3}$$

over the constraints (8.1) and (8.2) solves (MLCM). We can linearize the objective function by introducing binary crossing variables

$$c_{st,uv} \in \{0,1\}, \qquad 1 \le i < p,\ (s,u),(t,v) \in E_i, \tag{8.4}$$

that shall be 1 if the edges (s,u) and (t,v) cross and 0 otherwise. To bind the crossing variables with the ordering variables, we need the constraints

$$-c_{st,uv} \le x_{uv} - x_{st} \le c_{st,uv}, \qquad 1 \le i < p,\ (s,u),(t,v) \in E_i,\ u \dot{<} v, \tag{8.5}$$

$$1 - c_{st,uv} \le x_{vu} + x_{st} \le 1 + c_{st,uv}, \qquad 1 \le i < p,\ (s,u),(t,v) \in E_i,\ u \dot{>} v. \tag{8.6}$$

Let x be the vector (of length $s = \sum_{r=1}^{p} \binom{|V_r|}{2}$) collecting the variables x_{uv} and $c \in \mathbb{B}^t$ be the vector of crossing variables, with t the total number of crossing variables of the graph. Then we can formulate (MLCM) as a binary linear program as

$$z_* = \min \left\{ \sum_{1 \le i < p} \sum_{(s,u)(t,v) \in E_i} c_{st,uv} \ :\ (x,c) \in \mathcal{I}_{CR}(G) \right\} \tag{MLCM}$$

where

$$\mathcal{I}_{CR}(G) := \{\ (x,c) \text{ satisfy (8.2), (8.5) and (8.6)},\ (x,c) \in \{0,1\}^{s+t}\ \}.$$

Replacing the integrality conditions with 0-1 bounds gives the linear relaxation denoted by (LP_{MLCM}) with objective value z_{LP}. An exact algorithm using (LP_{MLCM}) has been introduced by [120] and was further extended by [119] and [96]. All these algorithms perform best on sparse instances (edge density ≤ 10 %)—see [31]—as for higher densities the gaps between z_* and the linear programming based bounds become too large for efficient pruning in Branch-and-Bound or Branch-and-Cut algorithms. In fact, by setting all x variables to 0.5, we can always—independently of the instance—obtain a feasible fractional solution with $z_{LP} = 0$. To prevent this, we propose to fix a single ordering variable to 1 or 0, to break the symmetry

without losing all optimal solutions. Yet in practice, the obtained relaxed solution still gives only a weak bound, especially for dense instances (for practical evidence see Chapter 14). Thus it would be desirable to have some tighter approximation available in these cases. In the next section we present one provably tighter approximation based on semidefinite optimization.

8.3 Exact Approaches Based on Semidefinite Programming

In this section we concentrate on the lower bound computation for (MLCM) by analyzing matrix liftings of ordering problems. We use (4.5) to transform the ordering variables x_{uv} into variables y_{uv} taking the values -1 and $+1$ and satisfying

$$-1 \leq y_{uv} + y_{vw} - y_{uw} \leq 1, \quad \forall u, v, w \in V_i, 1 \leq i \leq p,\ u \dot{<} v \dot{<} w. \tag{8.7}$$

Let us collect these variables in the vector y and consider the matrix $Y = yy^T$. We are interested in multi-level quadratic orderings and therefore consider the polytope

$$\mathcal{P}_{MQO} := \operatorname{conv}\left\{ \begin{pmatrix}1\\ y\end{pmatrix}\begin{pmatrix}1\\ y\end{pmatrix}^{\top} : \ y \in \{-1,1\}^s,\ y \text{ satisfies (8.7)} \right\}.$$

Following the ideas of Section 4.3 we define the matrix Z

$$Z = Z(y, Y) := \begin{pmatrix} 1 & y^T \\ y & Y \end{pmatrix}, \tag{8.8}$$

with dimension $\zeta := \dim(Z) = 1 + \sum_{i=1}^{p} \binom{|V_i|}{2}$ and the corresponding elliptope

$$\mathcal{E} := \{\ Z :\ \operatorname{diag}(Z) = e, Z \succcurlyeq 0\ \}. \tag{8.9}$$

Applying the dimension result for Quadratic Ordering Problems from [31] to Multi-level Quadratic Ordering Problems (MQOP), it is easy to deduce that the 3-cycle equations

$$y_{uv}y_{vw} - y_{uv}y_{uw} - y_{uw}y_{vw} = -1, \qquad \forall u, v, w \in V_i,\ 1 \leq i \leq p,\ u \dot{<} v \dot{<} w, \tag{8.10}$$

describe the smallest linear subspace containing $\mathcal{P}_{MQO}$. To formulate (MLCM) as a semidefinite optimization problem in bivalent variables we still need a symmetric matrix C of order $\zeta - 1$ that is assigned to count the number of crossings for any given feasible ordering y. We will now discuss how to compute such a matrix.

For all distinct pairs $s \dot{<} t \in V_i$ and all distinct pairs $u \dot{<} v \in V_{i+1}$, we consider the subgraph induced by these four vertices. We have to distinguish four different situations:

1. Let $(s,v),(t,u),(s,u),(t,v) \in E_r$. Independent of the order of $\{s,t\}$ and $\{u,v\}$, there will be exactly one crossing in the subgraph induced by these vertices. Thus we set $C_{st,uv} := 0$ to reflect that the number of crossings is independent of the ordering of the vertices. Additionally we increase a counter for unavoidable crossings δ by one in this case.

2. Let $|\{(s,v),(t,u)\} \cap E_r| < 2$ and $|\{(s,u),(t,v)\} \cap E_r| < 2$. The given subgraph has no crossing, regardless of the vertex orderings, and hence $C_{st,uv} := 0$.

3. Let $(s,v),(t,u) \in E_r$ and $|\{(s,u),(t,v)\} \cap E_r| < 2$. If the ordering has s before t in layer r and u before v in layer $r+1$, that is $y^r_{st}y^{r+1}_{uv} = 1$, there will be one crossing in the subgraph. If $y^r_{st}y^{r+1}_{uv} = -1$, there is no crossing. Since this is a minimization problem, we set $C_{st,uv} := \frac{1}{4}$.

4. Let $|\{(s,v),(t,u)\} \cap E_r| < 2$ and $(s,u),(t,v) \in E_r$. Up to node relabelling, this situation is identical to the previous case, so we set $C_{st,uv} := -\frac{1}{4}$.

We set all other entries in C to 0. In the general case we will have crossing matrices C_i for each layer $i = 1, \ldots, p-1$. We can therefore model (MLCM) by the cost matrix

$$C = \begin{bmatrix} 0 & C_1 & \ldots & 0 & 0 \\ C_1^\top & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 & C_{p-1} \\ 0 & 0 & \ldots & C_{p-1}^\top & 0 \end{bmatrix} \tag{8.11}$$

where the C_i, $1 \leq i < p$, have dimension $\binom{|V_i|}{2} \times \binom{|V_{i+1}|}{2}$ and are determined by the edge set E_i as described above.

Example 8.1 *We now demonstrate this encoding scheme on a small example (bipartite) graph.*

$$V_1 = \{1, 2, 3\},$$
$$V_2 = \{4, 5, 6, 7\}, \text{ and}$$
$$E_1 = \{(1,4), (1,5), (1,6), (2,4), (2,5), (2,7), (3,4), (3,6)\}.$$

The matrix C_1 is indexed row-wise by the pairs $12, 13, 23$. The columns are indexed by the pairs of elements from V_2, i.e., $45, 46, 47, 56, 57, 67$. We set

$$C_1 = \begin{bmatrix} 0 & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ \frac{1}{4} & -\frac{1}{4} & 0 & -\frac{1}{4} & 0 & 0 \\ \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & 0 & \frac{1}{4} \end{bmatrix}.$$

The entries $C_{12,45} = 0$, $C_{12,46} = \frac{1}{4}$, $C_{12,47} = -\frac{1}{4}$ and $C_{13,47} = 0$ fall into categories 1, 3, 4 and 2 above, respectively.

Now let us formulate (MLCM) as a semidefinite optimization problem in bivalent variables (note that the proofs of Theorem 8.2 and Theorem 4.1 are essentially identical).

Theorem 8.2 (MLCM) *is equivalent to the problem*

$$z_* = \min \{ \langle C, Z \rangle + c^\top y + K : Z \in \mathcal{I}_{MQO} \}, \tag{MLCM}$$

where c is a zero vector, $K := \delta + \sum_{i,j=1}^{\zeta} |C_{i,j}|$ and

$$\mathcal{I}_{MQO} := \{ Z : Z \text{ partitioned as in (8.8) satisfies (8.10)}, Z \in \mathcal{E}, y \in \{-1, 1\}^s \}.$$

Proof. Since $y_{uv}^2 = 1$ we have $\operatorname{diag}(Y - yy^T) = 0$, which together with $Y - yy^T \succeq 0$ shows that in fact $Y = yy^T$. The 3-cycle equations (8.10) ensure that $|y_{uv} + y_{vw} - y_{uw}| = 1$ holds. Therefore any matrix $Z \in \mathcal{I}_{MQO}$ is bivalent in its entries and represents feasible orderings on all layers. Thus by definition of the cost matrix C, the objective value $\langle C, Z \rangle + c^\top y + K$ gives the number of crossings. □

By dropping the integrality conditions on y, we get the following basic semidefinite relaxation for (MLCM)

$$\min \{ \langle C, Z \rangle : Z \text{ partitioned as in (8.8) satisfies (8.10)}, Z \in \mathcal{E} \}. \tag{SDP$_I$}$$

We further tighten (SDP$_I$) by reusing the ideas from Section 4.3. As $Z \in I_{MQO}$ is a matrix with $\{-1, 1\}$ entries, it satisfies the $4\binom{\zeta}{3} = O((\sum_{i=1}^{p} |V_i|^2)^3)$ triangle inequalities, defining the metric polytope

$$\mathcal{M} := \left\{ Z : \begin{pmatrix} -1 & -1 & -1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} z_{ij} \\ z_{jk} \\ z_{ik} \end{pmatrix} \leq e, \ \forall\, 1 \leq i < j < k \leq \zeta \right\}. \tag{8.12}$$

Adding $Z \in \mathcal{M}$ to $(\mathtt{SDP}_I)$ yields $(\mathtt{SDP}_{II})$.

We also apply the approach of Lovász and Schrijver [148] to tighten $(\mathtt{SDP}_I)$. It suggests to multiply the 3-cycle inequalities (8.7)(on level i, say) by the nonnegative expressions $(1-y_{uv})$, $(1+y_{uv})$, $(1-y_{uv}-y_{vw}+y_{uw})$ and $(1+y_{uv}+y_{vw}-y_{uw})$, respectively, where the nodes $u \dot{<} v \dot{<} w$ are on some (probably different) level j. There are $O((\sum_{i=1}^{p}|V_i|^3)^2)$ such LS-cuts. Due to the structure of the cost matrix (8.11), efficiency considerations and the theoretical results proved in the next section, we only work with the following subset of the LS-cuts in our computational experiments

$$\begin{aligned} -1-y_{vw} \le y_{st}+y_{tu}-y_{su}+y_{st,vw}+y_{tu,vw}-y_{su,vw} \le 1+y_{vw}, \\ -1+y_{vw} \le y_{st}+y_{tu}-y_{su}-y_{st,vw}-y_{tu,vw}+y_{su,vw} \le 1-y_{vw}, \\ 1 \le i < p,\ s,t,u \in V_i,\ s \dot{<} t \dot{<} u,\ v,w \in V_{i+1},\ v \dot{<} w. \end{aligned} \tag{8.13}$$

These LS-cuts are closely related to the constraints (4.13). As a future research project, we plan to conduct an experimental study to deduce the practical advantages and disadvantages of including further LS-cuts in the SDP relaxations. Especially for Multi-level Verticality Opimization (for details see the next chapter), where it is more difficult to get tight bounds than for (MLCM) (because of the more complex cost structure), we suppose to achieve practical improvements by adding further constraint types to the SDP relaxations. But for now let us define

$$\mathcal{LS} := \{\ Z : Z \text{ satisfies (8.13)}\ \}.$$

Adding $Z \in \mathcal{LS}$ to $(\mathtt{SDP}_I)$ yields $(\mathtt{SDP}_{III})$.

Overall, we get the following tractable relaxation of $\mathcal{P}_{MQO}$ that is closely related to the strongest relaxation $(\mathtt{SDP}_4)$ (see Section 4.3) for Quadratic Ordering Problems

$$\min\ \{\ \langle C, Z\rangle\ + c^\top y + K :\ Z \text{ partitioned as in (8.8) satisfies (8.10)},\ Z \in (\mathcal{E} \cap \mathcal{M} \cap \mathcal{LS})\ \}. \qquad (\mathtt{SDP}_{IV})$$

In the following section we relate $\mathcal{P}_{MQO}$ and the relaxation $(\mathtt{SDP}_{IV})$ to the crossing polytope $\mathcal{P}_{CR}(G) = \mathrm{conv}(\mathcal{I}_{CR}(G))$ and its facets.

8.4 Some Polyhedral Results

We start our polyhedral study by relating $(\mathtt{SDP}_I)$ (and therefore also $(\mathtt{SDP}_{II})$–$(\mathtt{SDP}_{IV})$) to the linear relaxation $(\mathtt{LP}_{\mathtt{MLCM}})$ of (MLCM).

Theorem 8.3 *The basic semidefinite relaxation $(\mathtt{SDP}_I)$ together with the linearized constraints*

$$(1 \pm y_{st})(1 \pm y_{uv}) \ge 0, \quad s,t \in V_i,\ u,v \in V_{i+1}, 1 \le i < p,\ s \dot{<} t,\ u \dot{<} v,$$

(and therefore also $(\mathtt{SDP}_{II})$ and $(\mathtt{SDP}_{IV})$) is at least as strong as the linear relaxation $(\mathtt{LP}_{\mathtt{MLCM}})$.

Proof. First, it is not hard to verify that any Z feasible for $(\mathtt{SDP}_{IV})$ contains a vector y in its first column that satisfies the 3-cycle inequalities (8.7) on the layers. This follows from the semidefiniteness of the following submatrix of Z

$$\begin{pmatrix} 1 & y_{uv} & y_{uw} & y_{vw} \\ y_{uv} & 1 & y_{uv,uw} & y_{uv,vw} \\ y_{uw} & y_{uw,uv} & 1 & y_{uw,vw} \\ y_{vw} & y_{vw,uv} & y_{vw,uw} & 1 \end{pmatrix}, \quad \forall u,v,w \in V_i, 1 \le i \le p,\ u \dot{<} v \dot{<} w.$$

The crossing variables $c_{st,uv}$ are related to $y_{st,uv}$ in the SDP model via the linear transformation

$$y_{st,uv} = 1 - 2c_{st,uv}. \tag{8.14}$$

The constraints (8.5) and (8.6) of the ILP formulation are thus equivalent to

$$\begin{array}{l} -1 + y_{st,uv} \le y_{uv} - y_{st} \le 1 - y_{st,uv}, \\ -1 - y_{st,uv} \le y_{uv} + y_{st} \le 1 + y_{st,uv}, \end{array} \qquad s,t \in V_i,\ u,v \in V_{i+1}, 1 \le i < p,\ s \dot{<} t,\ u \dot{<} v,$$

on the SDP variables. These latter inequalities are special cases (namely $j = i+1$) of the general lifting

$$(1 \pm y_{st})(1 \pm y_{uv}) \ge 0, \quad s,t \in V_i,\ 1 \le i < p,\ s \dot{<} t,\ u,v \in V_j,\ 1 < j \le p,\ u \dot{<} v,$$

that is embodied in (8.12). □

We know from [119] that $\mathcal{P}_{CR}(G) = \mathrm{conv}(\mathcal{I}_{CR}(G)) \in \mathbb{B}^{(\zeta-1)+r}$ is full dimensional. We relate $\mathcal{P}_{CR}(G)$ to $\mathcal{P}_{MQO}$, then we present various classes of facet-defining inequalities for $\mathcal{P}_{CR}(G)$ and show that $(\mathtt{SDP}_{IV})$ contains them. For the former, we consider the lifting $\mathcal{P}^*_{CR}$ of $\mathcal{P}_{CR}$ by extending the variable vector c to incorporate all possible crossing variables, not only for vertex pairs of adjacent layers with associated costs $\neq 0$. Formally, let $I := \{(s,t,u,v) \mid s \dot{<} t$ and $u \dot{<} v$ and ($s \dot{<} u$ or ($s = u$ and $t \dot{<} v$))$\}$, and consider the constraints

$$c_{st,uv} \le x_{st} + x_{uv}, \quad c_{st,uv} \le 2 - x_{st} - x_{uv}, \quad c_{st,uv} \ge x_{st} - x_{uv}, \quad c_{st,uv} \ge x_{uv} - x_{st}, \tag{8.15}$$

for all $(s,t,u,v) \in I$. Taking the extended variable vector c (of length $\bar{t}$), we define

$$\mathcal{P}^*_{CR} := \mathrm{conv}\{(x,c) \text{ satisfy (8.2) and (8.15)},\ (x,c) \in \{0,1\}^{s+\bar{t}}\ \}.$$

Theorem 8.4 *$\mathcal{P}_{CR}(G)$ contains $\mathcal{P}^*_{CR}$, when projected onto the common variables. $\mathcal{P}^*_{CR}$ and $\mathcal{P}_{MQO}$ describe the same polytope, modulo transformations* (4.5) *and* (8.14).

Proof. After using

$$c_{st,vu} = 1 - c_{st,uv}, s \dot{<} t,\ u \dot{<} v, \tag{8.16}$$

to eliminate $c_{st,vu}$ in $\mathcal{P}_{CR}(G)$, it is obvious by comparison of the constraint sets of the polytopes that $\mathcal{P}_{CR}(G)$ contains $\mathcal{P}^*_{CR}$.

Next let us summarize the constraints of $\mathcal{P}^*_{CR}$

$$0 \le x_{uv} + x_{vw} - x_{uw} \le 1, \qquad \forall u,v,w \in V_i,\ 1 \le i \le p,\ u \dot{<} v \dot{<} w, \tag{8.17}$$

$$c_{ij,kl} \le x_{ij} + x_{kl} \le 2 - c_{ij,kl}, \qquad (i,j,k,l) \in I, \tag{8.18}$$

$$-c_{ij,kl} \le x_{ij} - x_{kl} \le c_{ij,kl}, \qquad (i,j,k,l) \in I, \tag{8.19}$$

$$c_{st,uv} \in \{0,1\}, \qquad (s,t,u,v) \in I, \tag{8.20}$$

$$x_{uv} \in \{0,1\}, \qquad \forall u,v \in V_i,\ 1 \le i \le p,\ u \dot{<} v. \tag{8.21}$$

It follows directly from the definition of $\mathcal{P}_{MQO}$ that (8.17), (8.20) and (8.21) hold for all elements in $\mathcal{P}_{MQO}$ in the $\{-1,1\}$ formulation. Further on (8.18) and (8.19) hold for all elements in $\mathcal{P}_{MQO}$ in the $\{-1,1\}$ formulation due to validness of the triangle inequalities (8.12). For instance we obtain (8.18) by using type 1 and type 4 of (8.12)

$$-y_{ij} - y_{kl} - y_{ij,kl} \le 1, \quad \text{and} \quad y_{ij} + y_{kl} - y_{ij,kl} \le 1,$$

and then applying the linear transformations (4.5) and (8.14) respectively

$$c_{ij,kl} \le x_{ij} + x_{kl} \le 2 - c_{ij,kl}.$$

On the other hand (8.17)–(8.21) in the $\{-1,1\}$ formulation ensure $yy^\top = Y$ and (8.10). Thus any matrix Z that satisfies (8.17)–(8.21) lies in $\mathcal{P}_{MQO}$ □

Corollary 8.5 *Facet-defining inequalities of $\mathcal{P}_{CR}(G)$ are valid inequalities for $\mathcal{P}_{MQO}$.*

Proposition 8.6 ([119]) *Let C be a cycle and W_q a q-claw in G. The following inequalities are valid for $\mathcal{P}_{CR}(G)$:*

$$\sum_{(s,u)(t,v)\in C} c_{st,uv} \geq |C|/2 - 1, \qquad \sum_{(s,u)(t,v)\in W_q} c_{st,uv} \geq \begin{cases} \frac{q}{2}\left(\frac{q}{2}-1\right) & q \text{ even} \\ \left(\frac{q-1}{2}\right)^2 & q \text{ odd} \end{cases} \tag{8.22), (8.23}$$

Let T be the set consisting of all pairs of edges of a W_3 except those pairs of edges that are either both within the lower or upper part of the 3-claw. The following inequalities are facet-defining for $\mathcal{P}_{CR}(G)$:

$$\sum_{(u,v)\in C, u\neq s, v\neq s+1} c_{su,(s+1)v} + c_{s(s+1),(s-1)(s+2)} \geq 1, \qquad \sum_{(s,t),(u,v)\in T} c_{su,tv} \geq 1. \tag{8.24), (8.25}$$

For $s,t,u \in V_i,\ s \dot{<} t \dot{<} u,\ v,w \in V_{i+1},\ 1 \leq i < p$ the following inequalities constructed from dome paths are facet-defining for $\mathcal{P}_{CR}(G)$:

$$\begin{aligned} &x_{st} - 2x_{su} + x_{tu} - c_{st,vw} - c_{tu,vw} \leq 0, && -x_{st} + 2x_{su} - x_{tu} - c_{st,vw} - c_{tu,vw} \leq 0, \\ &2x_{st} - x_{su} + x_{tu} - c_{su,vw} - c_{ut,vw} \leq 1, && -2x_{st} + x_{su} - x_{tu} - c_{su,vw} - c_{ut,vw} \leq -1, \\ &x_{st} - x_{su} + 2x_{tu} - c_{ts,vw} - c_{su,vw} \leq 1, && -x_{st} + x_{su} - 2x_{tu} - c_{ts,vw} - c_{su,vw} \leq -1. \end{aligned} \tag{8.26}$$

Theorem 8.7 (SDP$_{IV}$) *satisfies* (8.22)–(8.26) *except* (8.23) *for $q \geq 5$ and odd.*

We prove this theorem through the following sequence of lemmas that deal with the stated constraints independently:

Lemma 8.8 (SDP$_{IV}$) *satisfies* (8.22) *and* (8.24) *due to* (8.9) *together with types 2,3,4 of* (8.12).

Proof. For a cycle of length 4 the statement is trivial. Therefore let us look at a cycle of length 6. Consider orderings $i \dot{<} j \dot{<} k$ and $l \dot{<} m \dot{<} n$ of the vertices and edges (i,l), (i,m), (j,l), (j,n), (k,m), (k,n). We write out the inequalities of types 2,3, or 4 from (8.12) in pairs

$$\begin{aligned} y_{ij,ik} - y_{ij,lm} + y_{ik,lm} &\leq 1, & -y_{ij,ik} + y_{ij,ln} + y_{ik,ln} &\leq 1, \\ y_{ij,jk} - y_{ij,lm} + y_{jk,lm} &\leq 1, & -y_{ij,jk} + y_{ij,ln} + y_{jk,ln} &\leq 1, \\ y_{ij,ik} - y_{ij,lm} + y_{ik,lm} &\leq 1, & -y_{ij,ik} + y_{ij,mn} + y_{ik,mn} &\leq 1, \\ y_{ik,lm} + y_{ik,mn} - y_{lm,mn} &\leq 1, & y_{jk,lm} - y_{jk,mn} + y_{lm,mn} &\leq 1, \\ y_{ik,ln} + y_{ik,mn} - y_{ln,mn} &\leq 1, & y_{jk,ln} - y_{jk,mn} + y_{ln,mn} &\leq 1, \\ y_{ij,ln} + y_{ij,mn} - y_{ln,mn} &\leq 1, & y_{jk,ln} - y_{jk,mn} + y_{ln,mn} &\leq 1, \end{aligned}$$

and then add the pairs together

$$\begin{aligned} -y_{ij,lm} + y_{ik,lm} + y_{ij,ln} + y_{ik,ln} &\leq 2, \\ -y_{ij,lm} + y_{jk,lm} + y_{ij,ln} + y_{jk,ln} &\leq 2, \\ -y_{ij,lm} + y_{ik,lm} + y_{ij,mn} + y_{ik,mn} &\leq 2, \\ y_{ik,lm} + y_{ik,mn} + y_{jk,lm} - y_{jk,mn} &\leq 2, \\ y_{ik,ln} + y_{ik,mn} + y_{jk,ln} - y_{jk,mn} &\leq 2, \\ y_{ij,ln} + y_{ij,mn} + y_{jk,ln} - y_{jk,mn} &\leq 2. \end{aligned}$$

Applying the linear transformation (8.14) and equation (8.16) yields

$$1 \leq 1 - c_{ij,lm} + c_{ik,lm} + c_{ij,ln} + c_{ik,ln} = \sum_{(u,v)\in C, u\neq i, v\neq l} c_{iu,lv} + c_{ij,ml},$$

$$1 \leq 1 - c_{ij,lm} + c_{jk,lm} + c_{ij,ln} + c_{jk,ln} = \sum_{(u,v)\in C, u\neq j, v\neq l} c_{ju,lv} + c_{ij,ln},$$

$$1 \leq 1 - c_{ij,lm} + c_{ik,lm} + c_{ij,mn} + c_{ik,mn} = \sum_{(u,v)\in C, u\neq i, v\neq m} c_{iu,mv} + c_{ki,ml},$$

$$1 \leq c_{ik,lm} + c_{ik,mn} + c_{jk,lm} + 1 - c_{jk,mn} = \sum_{(u,v)\in C, u\neq k, v\neq m} c_{ku,mv} + c_{ki,nm},$$

$$1 \leq c_{ik,ln} + c_{ik,mn} + c_{jk,ln} + 1 - c_{jk,mn} = \sum_{(u,v)\in C, u\neq k, v\neq n} c_{ku,nv} + c_{jk,nm},$$

$$1 \leq c_{ij,ln} + c_{ij,mn} + c_{jk,ln} + 1 - c_{jk,mn} = \sum_{(u,v)\in C, u\neq j, v\neq n} c_{ju,nv} + c_{jk,ln}.$$

These are exactly the six facets given by (8.24) for the given cycle. Summing over them, adding the trivial inequality

$$c_{ij,mn} + c_{ik,ln} + c_{jk,lm} \geq 0,$$

and dividing by 3 gives (8.22) for the given cycle.

Analogously we can deduce (8.24) for any cycle of any length, as these facets only depend on the relative position of six vertices and can be deduced as the sum of two inequalities of (8.12). Summing over them and adding a trivial inequality yields (8.22) for any given cycle. □

Lemma 8.9 (SDP$_{IV}$) *satisfies* (8.25) *and* (8.23) *for* $q = 3$ *due to* (8.9) *together with* (8.12).

Proof. Without loss of generality suppose that orderings $i \dot{<} j \dot{<} k$ and $l \dot{<} m \dot{<} n \dot{<} o$ of the vertices are given. Considering symmetry we have to examine four different edge configurations:

- In the first case the edges (i,l), (i,m), (j,l), (j,n), (k,l), (k,o) are given. Adding one inequality of type 1 and three inequalities of type 2 of (8.12)

$$-y_{lm,ln} - y_{lm,lo} - y_{ln,lo} \leq 1, \qquad -y_{ij,lm} + y_{ij,ln} + y_{lm,ln} \leq 1,$$
$$-y_{ik,lm} + y_{ik,lo} + y_{lm,mo} \leq 1, \qquad -y_{jk,ln} + y_{jk,lo} + y_{ln,lo} \leq 1,$$

  yields

$$-y_{ij,lm} + y_{ij,ln} - y_{ik,lm} + y_{ik,lo} - y_{jk,ln} + y_{jk,lo} \leq 4.$$

  Applying the linear transformation (8.14) and equation (8.16) gives

$$1 \leq c_{ij.ml} + c_{ij,ln} + c_{ik,ml} + c_{ik,lo} + c_{jk,nl} + c_{jk,lo} = \sum_{(i,j),(k,l)\in T} c_{ik,jl}$$

  This is exactly (8.25) for the given 3-claw. Now adding the trivial inequality

$$c_{ij,mn} + c_{ik,mo} + c_{jk,no} \leq 0,$$

  gives (8.23) for $q = 3$ for the given 3-claw.

- If vertex m has degree 3 we need types 2 and 4 of (8.12); if vertex n has degree 3 we need types 2 and 3 of (8.12); if vertex o has degree 3 we need types 1 and 3 of (8.12).

□

Lemma 8.10 ($\texttt{SDP}_{IV}$) *satisfies* (8.23) *for even* q *due to* (8.9) *together with types 2,3,4 of* (8.12).

Proof. Let us first take a look at 4-claws. Without loss of generality suppose that the orderings $i \dot{<} j \dot{<} k \dot{<} l$ and $m \dot{<} n \dot{<} o \dot{<} t \dot{<} u$ of the vertices are given. Considering symmetry we have to examine five different edge configurations.

- In the first case the edges (i,m), (i,n), (j,m), (j,o), (k,m), (k,t), (l,m), (l,t) are given. Due to semidefiniteness of the submatrix of Z

$$\begin{pmatrix} 1 & y_{mn,mo} & y_{mn,mt} & y_{mn,mu} \\ y_{mo,mn} & 1 & y_{mo,mt} & y_{mo,mu} \\ y_{mt,mn} & y_{mt,mo} & 1 & y_{mt,mu} \\ y_{mu,mn} & y_{mu,mo} & y_{mu,mt} & 1 \end{pmatrix} \tag{8.27}$$

  we get the following inequality by using the vector $[1,1,1,1]$

$$-y_{mn,mo} - y_{mn,mt} - y_{mn,mu} - y_{mo,mt} - y_{mo,mu} - y_{mt,mu} \leq 2.$$

  Adding inequalities of type 2 of (8.12)

$$\begin{aligned} -y_{ij,mn} + y_{ij,mo} + y_{mn,mo} &\leq 1, & -y_{ik,mn} + y_{ik,mt} + y_{mn,mt} &\leq 1, \\ -y_{il,mn} + y_{il,mu} + y_{mn,mu} &\leq 1, & -y_{jk,mo} + y_{jk,mt} + y_{mo,mt} &\leq 1, \\ -y_{jl,mo} + y_{jl,mu} + y_{mo,mu} &\leq 1, & -y_{jk,mt} + y_{jk,mu} + y_{mt,mu} &\leq 1, \end{aligned}$$

  yields

$$\begin{aligned} -y_{ij,mn} + y_{ij,mo} - y_{ik,mn} + y_{ik,mt} - y_{il,mn} + y_{il,mu} - y_{jk,mo} + \\ + y_{jk,mt} - y_{jl,mo} + y_{jl,mu} - y_{kl,mt} + y_{jk,mu} \leq 8. \end{aligned}$$

  Applying the linear transformation (8.14) and equation (8.16) gives

$$\begin{aligned} 2 \leq 6 - c_{ij,mn} + c_{ij,mo} - c_{ik,mn} + c_{ik,mt} - c_{il,mn} + c_{il,mu} - c_{jk,mo} + c_{jk,mt} - \\ c_{jl,mo} + c_{jl,mu} - c_{kl,mt} + c_{jk,mu} = c_{ij,mn} + c_{ij,mo} + c_{ik,mn} + c_{ik,mt} + \\ c_{il,mn} + c_{il,mu} + c_{jk,mo} + c_{jk,mt} + c_{jl,mo} + c_{jl,mu} + c_{kl,mt} + c_{jk,mu}. \end{aligned}$$

  Adding the trivial inequality

$$c_{ij,no} + c_{ik,nt} + c_{il,nu} + c_{jk,ot} + c_{jl,ou} + c_{kl,tu} \geq 0,$$

  finally yields

$$\sum_{(i,k)(j,l) \in W_4} c_{ij,kl} \geq 2.$$

  This is exactly (8.23) for $q = 4$ for the given 4-claw.

- In the second case n has degree 4, (8.27) is used with $[-1,1,1,1]$ and inequalities of types 2 and 4 of (8.12) are added. In the third case o has degree 4, (8.27) is used with $[-1,-1,1,1]$ and inequalities of types 2, 3 and 4 of (8.12) are added. In the fourth case t has degree 4, (8.27) is used with $[-1,-1,-1,1]$ and inequalities of types 3 and 4 of (8.12) are added. In the fifth case u has degree 4, (8.27) is used with $[-1,-1,-1,-1]$ and inequalities of types 3 and 4 of (8.12) are added.

Analogously, (8.23) can be shown for any $q \geq 6$ even. We just build the semidefinite submatrix of the vertices in the layer with $q+1$ vertices and use it with a vector that starts with α -1s, followed by a run of +1s, where $\alpha + 1$ is the position of the vertex of degree q in the layer with $q+1$ vertices. Then just adding the matching inequalities of (8.12) for every pair of vertices from the layer with q vertices, applying equations (8.14) and (8.16) and finally adding the remaining crossing variables in form of a trivial inequality gives the respective q claw. □

Lemma 8.11 (SDP$_{IV}$) *violates* (8.23) *for* $q \geq 5$ *odd by (at most)* $\frac{1}{2}$ *due to* (8.9) *together with types 2,3,4 of* (8.12)*.*

Proof. The argument is analogous to the above one for q even, except that for odd numbers the semidefinite submatrix used with the right vector gives an inequality that is $\frac{1}{2}$ weaker than the associated clique inequality of size q. Converted to the number of crossings this gives a (maximum) violation of $\frac{1}{4}$. To avoid this violation we would have to add the associated clique inequality of size q. □

Lemma 8.12 (SDP$_{IV}$) *satisfies* (8.26) *due to* (8.9) *together with* (8.13) *and types 2,3,4 of* (8.12)*.*

Proof. Taking the right inequality of the first equation of (8.13), setting $s = r$, setting lm respectively to ij, jk, ik, and adding the resulting three inequalities, gives

$$y_{ij} + y_{jk} - 2y_{ik} + y_{ij,jk} \leq 1.$$

Now adding the second inequality of (8.12)

$$-y_{ij,jk} + y_{ij,lm} + y_{jk,lm} \leq 1$$

yields

$$y_{ij} + y_{jk} - 2y_{ik} + y_{ij,lm} + y_{jk,lm} \leq 2.$$

Finally applying the linear transformations (4.5) and (8.14) and equation (8.16) gives

$$x_{ij} + x_{jk} - 2x_{ik} - c_{ij,lm} - c_{jk,lm} \leq 0.$$

This is exactly the first inequality of (8.26). In an analogous way, the other five inequalities of (8.26) can be deduced. In detail, we use the left inequality of the second equation of (8.13) together with the second inequality of (8.12) to deduce the second inequality of (8.26). The right inequality of the first equation of (8.13) together with the fourth inequality of (8.12) gives the third inequality of (8.26). The left inequality of the first equation of (8.13) together with the left inequality of the second equation of (8.13) and the fourth inequality of (8.12) yields the fourth inequality of (8.26). The right inequality of the first equation of (8.13) together with the right inequality of the second equation of (8.13) and the third inequality of (8.12) gives the fifth inequality of (8.26). Finally the left inequality of the first equation of (8.13) together with the third inequality of (8.12) yields the sixth inequality of (8.26). □

Corollary 8.13 (SDP$_{IV}$) *is as least as tight as* (LP$_{\text{MLCM}}$) *together with* (8.23)–(8.26) *except* (8.23) *for* $q \geq 5$ *and odd.*

In summary, all of the inequality types considered in (SDP$_{IV}$) are required to ensure facet-defining inequalities for $\mathcal{P}_{CR}(G)$. On the other hand, if we want an SDP relaxation that ensures more known facets of $\mathcal{P}_{CR}(G)$ than (SDP$_{IV}$), we have to consider additionally clique inequalities of size $q \geq 5$ odd in the relaxation. As separating all of them is far too expensive, this supports our model choice. A promising method for partially avoiding this limitation seems to be heuristic selection of the most important pentagonal inequalities proposed in Section 7.3.

Chapter 9

Multi-level Verticality Optimization

9.1 Introduction

In this chapter, we introduce the Multi-level Vertical Ordering Problem (MLVO). We propose exact and heuristic methods to solve it, discuss it from a polyhedral point of view and point out areas of application.

Generally, the problem can be described as a combination of multiple Linear Ordering Problems (LOP) (each is considered a level - for details on (LOP) see Chapter 4); instead of (only) having costs between elements within a level, the (main) costs arise from the positional differences between elements of distinct levels. For reasons that will become evident below, these latter differences can be considered as non-verticalities. In the following, we will apply (MLVO) in a graph drawing setting, where it arises most naturally and allows probably the simplest introduction into the problem class. Yet, note that (MLVO) by itself is of more general nature; we will showcase some additional applications in Section 9.11. When specifically talking about the graph drawing application, i.e., finding orderings of the nodes on their levels such that the edges are drawn as vertical as possible (see a precise definition below) we use the term Multi-level Verticality Optimization, which, nicely enough, gives also rise to the abbreviation (MLVO).

One of the most common drawing paradigms for hierarchical graphs, known as Sugiyama's framework [200], is composed of the following steps:

1. Place the nodes of a graph on different levels, effectively fixing their y-coordinates. Edges spanning multiple levels are subdivided into chains of (sub)edges such that each (sub)edge only spans one level, resulting in a proper level graph.

2. Fix an ordering of the nodes on their levels such that a certain optimization goal (usually the number of crossings) is minimized.

3. Assign x-coordinates (consistent with the ordering) to the nodes, such that, e.g., the number of bends is minimized or the edges' verticality is maximized. (Sub)edges are thereby always drawn as straight lines.

(MLVO) proposes a somehow inverse approach to the problem of finding a good node order on the levels, focusing on the third step's optimization goal. We observe that when thinking about a drawing where the edges are drawn mostly vertical, we will usually also have a low number of crossings. Furthermore, edges tend to cross only on a very local scale (i.e., edges will usually not cross over a large horizontal distance), increasing the drawing's readability [178]. Hence, perhaps this inverse approach leads to (qualitatively) better drawings than the ones obtained by Sugiyama's framework.

The assumption that high verticality leads to few crossings and good drawings is also supported by the following observation: The barycenter heuristic is one of the earliest, and still probably the most common heuristic to quickly solve the layered crossing minimization problem in practice, especially for large-scale

graphs. Yet, the heuristic does actually not actively try to minimize crossings, but iteratively decides on positions p of vertices on layer i, such that p lies at (or close to) the barycenter of the positions of its adjacent nodes on level $i-1$. So, the heuristic is in fact mainly trying to optimize (MLVO). Its crossing minimization properties arise only in the wake of this optimization goal.

As we will see, (MLVO) is a Multi-level Quadratic Ordering Problem (MQOP) and thus is akin to all the other ordering problems discussed in this thesis. Of course (MLVO) is NP-hard (for details see Section 9.10) and closely related to the traditional problem of multi-level crossing minimization (MLCM), where one seeks node orders such that the number of crossings in multi-level drawings is minimized. (MLCM) has received a lot of attention not only within the graph drawing community, but in combinatorial optimization in general (for more information and references on (MLCM) see Chapter 8). (MLVO) is also related to the problem of Multi-level Planarization (MLP) [78, 161], i.e., find node orders which minimize the number of edges that have to be removed in order to obtain a planar (sub)drawing. (MLP) has been proposed as a possible substitute for (MLCM), suggesting that it can result in more aesthetically pleasing drawings. Finally, problems related to (MLVO) also occur in computational geometry, e.g., when considering "optimal shapes" of towns [56], where n buildings are be placed on a 2-dimensional integer grid, and the aim is to minimize the manhattan distances between any pair of buildings.

The remainder of this chapter is based on Sections 3, 4 and 6 of the paper "Exact Approaches to Multi-level Vertical Orderings" [42] and Sections 2, 3 and 4 of the paper "Multi-level Verticality Optimization: Concept, Strategies, and Drawing Scheme" [43]. In the next section, we will formally discuss the concept of verticality, its application to proper level graphs, and propose the (MLVO) problem as an alternative to the steps 2 and 3 in the traditional Sugiyama framework. Thereafter in Section 9.3, we propose a different new drawing style based on the verticality concept; it does not require to subdivide the edges after layering the graph in Sugiyama's first step. It seems that this is the first approach allowing the direct use of non-proper level graphs within the Sugiyama framework. In Section 9.4 we show how to adopt commonly known (MLCM) paradigms in order to obtain simple heuristics to solve (MLVO) in practice and for large graphs. Several exact approaches based on linear, quadratic or semidefinite relaxations are presented in Sections 9.5–9.7 and theoretically compared in Section 9.8. In Section 9.9 we discuss various extensions to (MLVO) that can be interesting in practice. For example, we can use the SDP approach as an exact quadratic compactor for Sugiyama's third stage, i.e., after minimizing crossings. The theoretical difficulty of (MLVO) is discussed in Section 9.10. Although graph drawing is the main (and most developed) application area, we finally showcase in Section 9.11 that (MLVO) can also be interesting in other, very diverse, application fields like scheduling and multiple ranking.

9.2 Verticality and Proper Drawings

Studying (MLVO) we will always consider the following input: Let $G = (V, E)$ with $V = \dot{\bigcup}_{i=1}^{p} V_i$ be a level graph, where we draw the nodes V_i on the i-th level. The function $\ell : V \to \{1, \dots, p\}$ gives the level on which a node resides. Furthermore, let $G' = (V', E')$ with $V' = \dot{\bigcup}_{i=1}^{p} V'_i$, $E' = \dot{\bigcup}_{i=1}^{p-1} E'_i$, and $E'_i \subseteq V'_i \times V'_{i+1}$ for all $1 \leq i < p$, be the corresponding proper level graph. Thereby, the original edges E are subdivided into segments such that each edge in E' connects nodes of adjacent levels. Clearly, we have $V_i \subseteq V'_i$ for all levels i. The additional nodes created by this operation are called long-edge dummy nodes, or LEDs for short.

(MLCM) and (MLP) are always applied to proper level graphs, as only the introduction of LEDs allows to concisely describe their feasible solutions and objective values. Optimizing these problems means solving $p-1$ dependent, sequentially linked bilevel (QOPs) (one for each pair of adjacent levels). We will see that (MLVO) cannot only be applied in such a setting (resulting in proper drawings), but also directly to non-proper graphs (resulting in non-proper drawings): this gives rise to a "true" (MQOP) as all levels can directly interact with each other. Thus (MLVO) seems to be the first optimization problem considered in

this realm that can naturally and reasonably be applied to non-proper graphs.

This is particularly interesting with respect to semidefinite programs: The theoretical results from Section 8.4 and the experimental studies from Chapter 14 show that SDPs have great potential for `(MLCM)`. Yet, when considering the cost matrix (8.11), we can observe that it is constructed of (non-zero) submatrices along its main diagonal; all other entries of the matrix are 0. Non-proper `(MLVO)` seems to be the first `(MQOP)` using the full SDP structure, resulting in denser cost matrices of smaller dimension. Therefore we can obtain (near-)optimal, (well-)readable drawings of graphs too large for Proper `(MLVO)` or `(MLCM)`. Furthermore, for both `(MLVO)` variants the pure relative node ordering is not sufficient to evaluate the objective function, but we also need to take the resulting absolute position of an element (first, second, etc.) into account. This induces a more complex quadratic cost structure on the involved levels compared to, e.g., crossing minimization, needing more of the implicitly defined SDP variables and therefore leading again to denser SDP cost matrices. But the more SDP variables and therefore SDP structure is needed, the better the SDP performs compared to competing ILP approaches (in the context of ordering problems).

We define the colloquial term verticality via its inverse, non-verticality: The non-verticality $\mathfrak{d}(e)$ of a straight-line edge e is the square of the difference in the horizontal coordinates of its end nodes. Then, $\mathfrak{d}(E) := \sum_{e \in E} \mathfrak{d}(e)$ denotes the overall non-verticality of a solution. Using only this notion, we could arbitrarily optimize a drawing by scaling the horizontal coordinates. Hence we consider grid drawings, i.e., the nodes' positions are mapped to integral coordinates, thereby relating verticality to the drawing's width. Clearly, we only consider adjacent integers for the y-coordinates. It remains to argue why non-verticality has to be a quadratic term: assume we would only consider a linear function, then even a small example such as the one depicted in Fig. 9.1(a) would result in multiple solutions that are equivalent w.r.t. their objective values, even though the bottom one is clearly preferable from the readability point of view. Intuitively, we prefer multiple slightly non-vertical edges, over few very non-vertical edges. In fact, this argument brings our model in line with the argument of observing crossings only on a local scale.

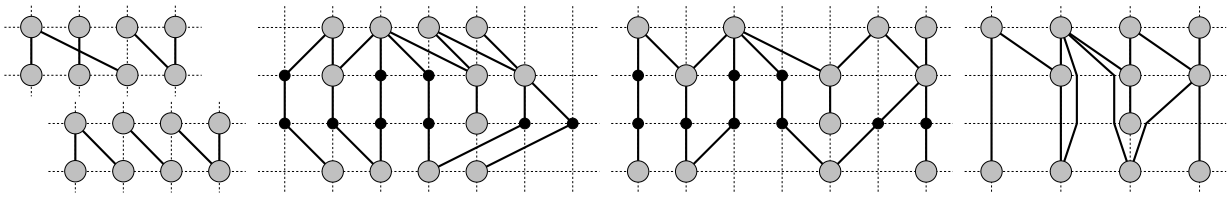
We can consider two distinct alignment schemes, due to the fact that the node partitions V'_i have different cardinalities. Let $\omega' := \max_{1 \leq i \leq p} |V'_i|$ denote the width of the widest level. In the narrow alignment schemes, we require the nodes on the levels to lie on directly adjacent x-coordinates (Fig. 9.1(b)). Then, we would usually like to center the distinct levels w.r.t. each other, i.e, a level i may only use the x-coordinates $\{\delta'_i, \ldots, \delta'_i + |V'_i| - 1\}$, with the level's width offset $\delta'_i := \lfloor(\omega' - |V'_i|)/2\rfloor$. The benefit of this alignment scheme is that a simple linear order of the nodes per layer already fully describes the solution. Yet, note that in most cases such an alignment scheme will not result in aestetically pleasing drawings.

In the wide alignment scheme (Fig. 9.1(c)), nodes are not restricted to lie on horizontally neighboring grid coordinates. In order to model this in our optimization framework, we expand the graph by adding positional dummy nodes (PDs) to each level such that all levels have ω' many nodes. All PDs have degree 0. Since this addition is the only necessary modification to obtain this alignment scheme[1], we will in the following continue to consider any (proper) level graph $G^{(\prime)}$, which may or may not be augmented with PDs.

9.3 Non-Proper Drawing Scheme

On the one hand, disregarding LEDs in our drawing style and thus considering a smaller graph, potentially improves the running times of our algorithm. On the other hand, this idea also has a foundation in graph drawing applications: When looking at typical Sugiyama-style drawings, we often observe that LEDs—even though they are never explicitly drawn—are given too much space: Objectively, it is unreasonable for LEDs to require as much horizontal space as a real node. Therefore, current drawing algorithms try hard to "bundle" multiple long edges into one dense channel (whose width is constant, disregarding the

[1] We can trivially force some predefined relative order of all PDs of a common level by fixing the corresponding ordering variables introduced later. This reduces the symmetry of this expansion and is therefore beneficial for Branch-and-Bound approaches.



(a) Linear Cost (b) Proper, narrow (c) Proper, wide (d) Non-proper, wide

Figure 9.1: Example drawings regarding verticality optimization: (a) equivalent quality with respect to a linear objective function, (b)–(d) different drawing paradigms, cf. text. Original nodes are drawn as large gray circles, LEDs as black small circles, positional dummy nodes (on the empty grid points) are omitted for readability.

number of its elements), to improve overall readability of large, dense graphs; see, e.g., [162]. Yet, such methods usually still use LEDs.

Herein, we show that a drawing scheme can be devised which makes LEDs completely unnecessary, cf. Figs. 9.1(d) and 9.2. We will, however, retain PDs as described above to allow a wide alignment scheme on our grid. A particularly interesting side effect of working without LEDs is that the considered graph stays smaller. Thus, this method allows more involved, time-consuming methods (as, e.g., our exact SDP-approach) to be applicable to larger original graphs.

Consider a non-proper level graph G for which we have computed a solution to (non-proper) `(MLVO)`, i.e., we have an ordering of the nodes on their layers and non-verticality of an edge is measured simply as the square of the horizontal coordinate difference of its end nodes. We now describe how to generate a drawing realizing such an order and verticality.

Hypothetical and shifted routing:

The y- and x-coordinates of the nodes are fixed by the layering (ℓ) and node order per layer, respectively. As a general idea—called the hypothetical routing—we want to draw each edge vertically up to the level directly below the target node. Only there, the edge bends to be drawn as a line with the computed non-verticality.

Clearly, there are problems with this simple concept: Firstly, routing edges strictly vertical may require to draw them through other nodes. Secondly, vertical segments of multiple edges would coincide. To avoid these issues, we have to relax the hypothetical routing such that we route an edge $e = (u, v)$ vertically "close to" the x-coordinate of the source node (i.e., shifted by some small $s(e)$). More formally, the edge starts at the coordinate $(x(u), \ell(u))$, has a first bend point at $(x(u) \pm s(e), \ell(u)+1)$ shifting the edge either to the left or to the right (depending on the edge's overall direction), and is routed vertically until the point $(x(u) \pm s(e), \ell(v) - 1)$ where it bends to go straight to the end point $(x(v), \ell(v))$. For short edges or $s(e) = 0$, some bend points may vanish in the obvious way. When $s(e)$ is assumed small enough, the overall non-verticality of this routing is roughly equivalent to the verticality achieved by the hypothetical routing. Observe that *vertical* edges (i.e., $x(u) = x(v)$) are somehow special as it is not per se clear, whether $s(e)$ should be added or subtracted; we will discuss this uncertainty later. Our overall goal is that the crossings induced by this shifted routing satisfy the following properties:

(P1) adjacent edges (edges with the same starting or end node) do not cross, all other edge pairs cross at most once;

(P2) a vertical edge e_1 *may* only cross another edge e_2 when exactly one end node of e_2 is vertically between the end nodes of e_1; and

(P3) two non-vertical edges cross exactly if their hypothetical routings cross.

In order to achieve these properties, the shift values $s(e)$ for the edges have to be chosen carefully.

Computing Shifts: Let $V^- \subseteq V$ denote the original vertices (in contrast to possible PDs). Larger y (x)

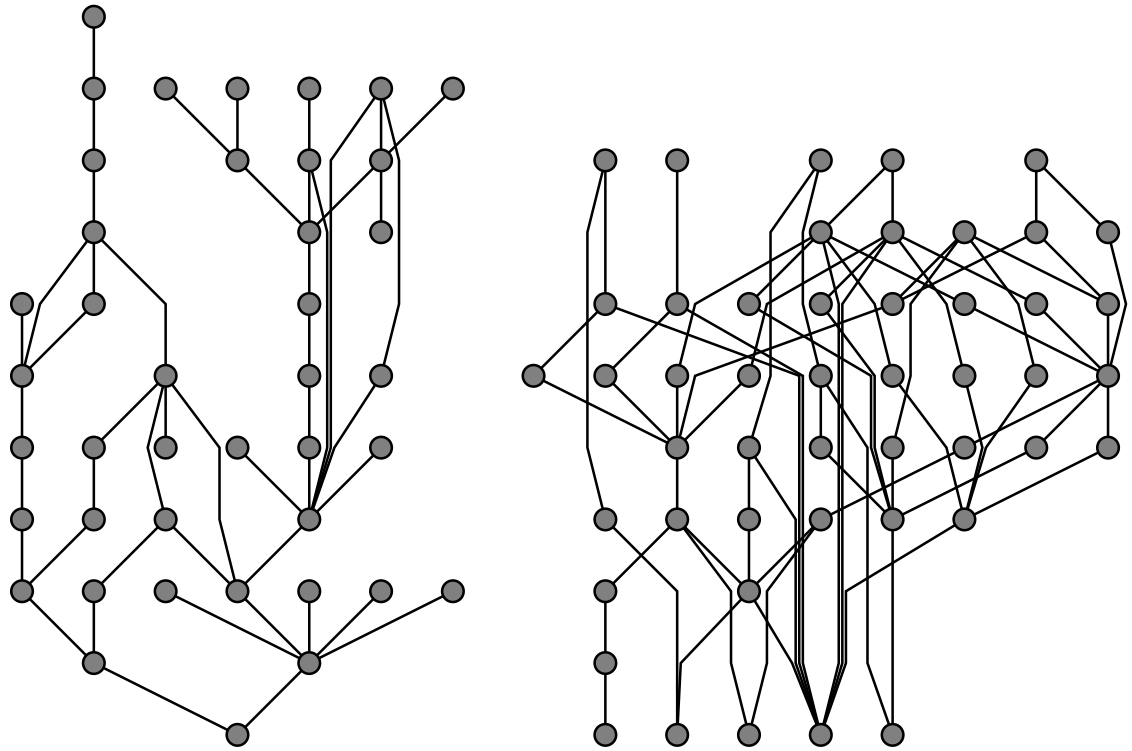


Figure 9.2: Examples of the non-proper drawing scheme with (near-)optimal verticality. Instances: *unix* (left), *world* (right), cf. Sect. 15.3, all with prespecified layering.

coordinates are higher (more right, respectively) in the drawing. We iteratively consider all nodes $v \in V^-$, in decreasing order of their y-coordinate. For all edges $e \in E_v := \{(v,u) \in E : \ell(u) > \ell(v)\}$ that have v as their source node, we will compute an integer label $\sigma(e)$. After labeling all edges of the graph, these labels will be transformed into actual shift values (see below; for now it is sufficient to think about a formula of the type $s(e) = \varepsilon \cdot \sigma(e)$ for some small $\varepsilon > 0$).

To these ends, we further subpartition E_v into $E_v^<, E_v^=, E_v^>$ depending on whether the target node u_e of an edge is left ($x(u_e) < x(v)$), directly above ($x(u_e) = x(v)$), or right ($x(u_e) > x(v)$) of v, respectively. For all nodes $w \in V$ we store the smallest free label $\sigma^l(w), \sigma^r(w)$ to its left and right side, respectively. Initially, these labels are 1 for $w \in V^-$ and 0 otherwise. Now, let $V_e := \{w \in V^- : x(w) = x(v) \wedge \ell(v) < \ell(w) < \ell(u_e)\}$ be the set of original nodes vertically above v, but below the edge's target node. Then, $\sigma^l_{\max}(e) := \max_{v \in V_e} \sigma^l(v)$ denotes the smallest possible label for e. If $V_e = \emptyset$, we set the value to 0 as we do not require any shift for e. Define $\sigma^r_{\max}(e)$ analogously.

First, consider the vertical edges $E_v^=$. Let $E_v^{=,l}, E_v^{=,r}$ be any partition of $E_v^=$—see below for a discussion of a proper choice—into edges that should be shifted to the left or to the right, respectively, if necessary. Now, sort $E_v^{=,l}$ ($E_v^{=,r}$ is analogous) by increasing layer of the target nodes, and iteratively (using the sorted order of the edges) apply integral labels. To label an edge e, set $\sigma(e) := \sigma^l_{max}(e)$ and afterwards $\sigma^l(w) := \sigma(e) + 1$ for all $w \in V_e$. Now consider the set $E_v^<$ ($E_v^>$ is analogous) and sort it by decreasing $\ell(u_e)$, where u_e is the edge's target node; edges within the same equivalence class w.r.t. this measure are sorted by increasing $|x(u_e) - x(v)|$. We can draw all edges that span only one level as straight lines and remove them from $E_v^<$ for the following discussion. Iterating over the edges e in sorted $E_v^<$, we again set $\sigma(e) := \sigma^l_{max}(e)$ and afterwards $\sigma^l(w) := \sigma(e) + 1$ for all $w \in V_e$. Observe that subsequent edges in sorted $E_v^<$ are always labeled 1 larger than their direct predecessor. In this scheme it may happen that the first edges of sorted $E_v^<$ and $E_v^>$ are labeled 0. If this is the case, we increase all labels of the set where the first

edge has the lower target node (breaking ties arbitrarily) by 1, to avoid co-linear lines due to not shifting two edges.

Let $0 < \alpha < \beta < 0.5$ be prespecified parameters describing the distance of minimum and maximum shift. Let σ^* be the largest overall label, then $\delta := (\beta - \alpha)/(\sigma^* - 1)$ denotes the shift difference between two adjacent edges. For any vertex $v \in V^-$ we compute the actual shift value for each emanating edge: If there is an edge e_0 in E_v with label 0, we set $s(e_0) = 0$. If e_0 exists and $e_0 \notin E_v^=$, let $\alpha' := 0$; otherwise $\alpha' := \alpha$. For any other edge $e \in E_v$, we then set $s(e) := (\sigma(e) - 1) \cdot \delta + \alpha'$.

Analysis of the drawing algorithm: By a careful (very technical but mathematically easy) case distinction over the above described algorithm we obtain:

Fact 9.1 *The drawing obtained by the above non-proper drawing algorithm satisfies the properties (P1)–(P3).*

Hence, the pure orderings of the nodes per layer already induce the required number of crossings, up to crossings due to vertical edges. These are decided by the respective *l,r-partitions* of the vertical edges, i.e., the partitions of $E_v^=$ into $E_v^{=,l}, E_v^{=,r}$, for all $v \in V^-$.

Fact 9.2 *Fixing all l,r-partitions, the above drawing algorithms requires the minimum possible number of crossings. Yet, even when given the node orders per level, obtaining l,r-partitions that lead to the overall minimum number of crossings is NP-hard.*

The first part follows again from careful (very technical but mathematically easy) case distinction over the above described algorithm. The NP-hardness follows from the fact that already a single column of vertically arranged vertices resembles the NP-hard fixed linear crossing number problem [154].

In practice, the partition problem is usually not critical: the number of crossings between pairs of vertical edges is usually dominated by the crossings involving non-vertical edges. In fact, in our implementation we settle on a very simple, yet seemingly sufficient, heuristic: during the algorithm, we greedily pick the side where the edge attains the smaller label; we break ties by classifying edges whose source node v is on the left (right) half of the drawing as $E_v^{=,l}$ ($E_v^{=,r}$, resp.). This tie breaking is reasonable, since considering a node v on the left side of the drawing, it will usually have more adjacent nodes to its right than to its left side, and hence the decision usually leads to fewer crossings.

Based on the fact that all sorting is done on integral values, we can conclude:

Theorem 9.3 *The above drawing algorithm generates a non-proper drawing of a level graph $G = (V, E)$ with specified node orders per level in $O(|V| + |E|)$ time. The edges' routings are monotonous in their x- and y-coordinate, as well as strictly monotonous (i.e., in their general direction) and realize the minimal number of crossings (w.r.t. the given node orders and l,r-partitions).*

In Chapter 15 we showcase and compare the visual effects of using different drawing and alignment schemes. There we also study the practical performance of several heuristic and exact solution methods to `(MLVO)`, introduced and theoretically compared in the following sections.

9.4 Basic Heuristic Approaches

Barycenter and Median: We already noted in the introduction that traditional `(MLCM)` heuristics in fact often optimize the drawings' verticality (in a narrow alignment scheme setting). In particular, we can use the traditional approach of computing the barycenter or median for the nodes, by only looking at fixed positions of the nodes one level below/above (in case of proper drawings), or on any level below/above (in case of non-proper drawings), and sort them accordingly. Iterating this procedure for all layers both in the upward and downward direction (i.e., alternatingly consider the levels below or above) until no

more improvement is possible minimizes the number of crossings only indirectly, but the edges' verticality directly.

When considering the wide alignment scheme, we observe that we cannot compute reasonable values for PDs as they have no incident edges. Therefore, we first only compute the barycenter/median for the original nodes—we call these values the desired locations of the nodes—and sort them accordingly. Then, we try to disperse the PDs (necessary to achieve the layer width ω') into this list such that desired locations numerically coincide with their final positions in the list (including the PDs) as well as possible. We do so heuristically by iteratively putting the PDs between two adjacent desired location values d_1, d_2 with largest gaps, and set the PD's location value to $\min\{(d_1 + d_2)/2, d_1 + 1\}$.

Local Optima via 2-Opt and Sifting: Consider an initial node order per level, either by random assignment or by applying the above barycenter or median heuristic. We can apply local optimization strategies to further improve the solution. Herein, we describe two implementation-wise simple, yet promising approaches, which are known from various different optimization problems, including `(MLCM)`.

The first approach is a 2-Opt strategy, i.e., we iteratively pick all possible pairs of nodes v_1, v_2 on a common layer, where at least one node is not a PD. We then exchange their positions and reevaluate the overall verticality. Clearly, we are only interested in the change of the solution value, and therefore it suffices to compute $\Delta\mathfrak{d} := \mathfrak{d}_{\text{before}}(E_1) + \mathfrak{d}_{\text{before}}(E_2) - \mathfrak{d}_{\text{after}}(E_1) - \mathfrak{d}_{\text{after}}(E_2)$, where E_1, E_2 are the edges having v_1, v_2 as one of their end points, respectively. We finally apply this modification only if $\Delta\mathfrak{d}$ is positive. The process stops when no more improving node pair can be found.

Similarly, we can devise a sifting strategy. We pick any two nodes v_1, v_2 (both may be PDs) on a common layer. Let V_{v_1,v_2} be the nodes between these, w.r.t. to the current node order on this level. We then shift all nodes $\{v_1\} \cup V_{v_1,v_2}$ by one position towards the old position of v_2, and move v_2 to the former position of v_1. To decide whether this is an improvement, we have to evaluate the non-verticalities of the edges incident to $\{v_1, v_2\} \cup V_{v_1,v_2}$. Again, we only perform improving steps and the process stops when no more such step is possible.

We refer to Section 15.3 for an experimental study of the practical performance of the proposed heuristics. In the following three sections we present exact methods for `(MLVO)` using linear, quadratic and semidefinite relaxations respectively.

9.5 Exact Approaches Based on Quadratic Programming

Consider the proper level graph G'. Along the lines of the previous chapters we introduce binary variables

$$x'_{uv} \in \{0,1\}, \qquad u, v \in V'_i,\ 1 \le i \le p,\ u \dot{<} v, \tag{9.1}$$

with some fixed total order $\dot{<}$ of the nodes (e.g. based on their indices) to model the node order on the levels. The variables shall be 1 if u is left of v and 0 otherwise. For notational simplicity, we also use the shorthand $x'_{uv} := 1 - x'_{vu}$ for $v \dot{<} u$. We have already argued in Section 4.2 that the following 3-cycle inequalities describe linear orderings on the layers of the given proper level graph

$$0 \le x'_{uv} + x'_{vw} - x'_{uw} \le 1, \qquad u, v, w \in V'_i,\ 1 \le i \le p,\ u \dot{<} v \dot{<} w. \tag{9.2}$$

For each edge $(u, v) \in E'$, we introduce a (conceptually integral) variable $d'_{(u,v)}$ measuring the end-nodes' horizontal distance, i.e., $\sqrt{\mathfrak{d}((u,v))}$: Consider the shorthand $X'(u) := \delta'_{\ell(u)} + \sum_{w \in V'_{\ell(u)}} x'_{wu}$ which gives the number of nodes left of u plus the level's width offset, and therefore the x-coordinate of u. Then the horizontal distance between u and v is $|X'(u) - X'(v)|$. In linear terms we can hence require

$$d'_{(u,v)} \ge X'(u) - X'(v)\ , \quad d'_{(u,v)} \ge X'(v) - X'(u), \qquad (u, v) \in E'. \tag{9.3}$$

This allows us to give a mathematical model with linear constraints but quadratic objective function to solve the Proper (MLVO) problem:

$$v'_* = \min\left\{\sum\nolimits_{e\in E'} \left(d'_e\right)^2, \text{ subject to } (9.1)-(9.3)\right\}. \tag{DM'}$$

Theorem 9.4 *Every optimal solution to* (DM') *induces an optimal solution to* (MLVO) *on proper level graphs, and vice versa.*

By replacing the integrality conditions in (DM') with 0-1 bounds we obtain a quadratic programming relaxation denoted by (cDM').

Let x' be the vector collecting all variables x'_{uv}. We can write $\mathfrak{d}(E')$ as a linear-quadratic function in x', without the explicit need for any d'-variables:

$$\mathfrak{d}(E') = \sum_{(u,v)\in E'} \left(X'(u) - X'(v)\right)^2 \stackrel{!}{=} \begin{pmatrix}1\\ x'\end{pmatrix}^\top D' \begin{pmatrix}1\\ x'\end{pmatrix}, \tag{9.4}$$

for some suitable matrix D'. Therefore we obtain an equivalent formulation to (DM').

Lemma 9.5 (DM') *and its relaxation* (cDM') *give the same values as*

$$\min\left\{\begin{pmatrix}1\\ x'\end{pmatrix}^\top D' \begin{pmatrix}1\\ x'\end{pmatrix}, \textit{ subject to } (9.1) \textit{ and } (9.2)\right\} \tag{OM'}$$

and its relaxation (cOM'), *respectively.*

Although the above models suffice w.r.t. integral solutions, their relaxations can be further strengthened. On the one hand, any polyhedral improvement for the ordering variables directly carries through to (MLVO). On the other hand we can add the following new classes of strengthening inequalities.

Degree Constraints: Consider some node $u \in V'_i$ and all its adjacent nodes N' on some neighboring level j (either above or below). For $\alpha := |N'|$, $\alpha \geq 2$, we can require

$$\sum\nolimits_{v\in N'} d'_{(u,v)} \geq \lfloor \alpha/2 \rfloor \cdot \lceil \alpha/2 \rceil, \tag{9.5}$$

$$\sum\nolimits_{v\in N'} (d'_{(u,v)})^2 \geq \begin{cases}\alpha(\alpha^2-1)/12, & \text{if } \alpha \text{ odd},\\ \alpha(\alpha^2+2)/12, & \text{if } \alpha \text{ even},\end{cases} \tag{9.6}$$

based on the fact that the minimum possible overall non-verticality is achieved when tightly packing N' and centering it above/below u. The second (quadratic) constraint thereby is a strengthening of the former (linear) constraint.

In order to obtain the right hand side of the degree constraints, consider all nodes N' being placed directly next to each other, and u centered below or above these nodes. We can sum the arising horizontal distances as

$$\underbrace{0+1+1+2+2+3+\ldots}_{\alpha \text{ many}} = \sum_{i=1}^{\lfloor \alpha/2 \rfloor} i + \sum_{i=1}^{\lceil \alpha/2 \rceil - 1} i = \lfloor \alpha/2 \rfloor \lceil \alpha/2 \rceil,$$

where the latter function is obtainable via case distinction on whether α is odd or even.

Similarly, we can sum up the arising non-verticalities (i.e., squares of the horizontal distances) as

$$\underbrace{0^2+1^2+1^2+2^2+2^2+3^2+\ldots}_{\alpha \text{ many}} = \sum_{i=1}^{\lfloor \alpha/2 \rfloor} i^2 + \sum_{i=1}^{\lceil \alpha/2 \rceil - 1} i^2,$$

which gives – using $\sum_{i=0}^{n} i^2 = n(n+1)(2n+1)/6$ and a case distinction on whether α is odd or even – the specified right hand side in (9.6).

Complete-Bipartite Constraints: We can generalize the degree constraints by considering complete bipartite subgraphs on consecutive levels. Let $N'_i \subseteq V'_i$ and $N'_{i+1} \subseteq V'_{i+1}$, for some $1 \leq i < p$, be two node sets such that $N'_i \times N'_{i+1} \subseteq E'_i$. Let $\beta := \min\{|N'_i|, |N'_{i+1}|\}$ and $\gamma := \max\{|N'_i|, |N'_{i+1}|\}$. We can require

$$\sum\nolimits_{u \in N'_i} \sum\nolimits_{v \in N'_{i+1}} d'_{(u,v)} \geq \beta \cdot \lfloor \gamma/2 \rfloor \cdot \lceil \gamma/2 \rceil + \begin{cases} \lfloor \beta/2 \rfloor \cdot \lceil \beta/2 \rceil, & \text{if } \gamma \text{ odd,} \\ \lfloor \beta/2 \rfloor \cdot (\lceil \beta/2 \rceil - 1), & \text{if } \gamma \text{ even,} \end{cases} \tag{9.7}$$

where the right hand side gives the minimum possible overall horizontal distances achieved by tightly packing the node sets on their levels, and centering them above each other. Again, we can strengthen (9.7) by considering squared d-variables and the accordingly increased right-hand side instead:

$$\sum_{u \in N_i, v \in N_j} (d'_{(u,v)})^2 \geq \begin{cases} \beta\gamma(\gamma^2 + \beta^2 - 2)/12 & \text{if } \beta \text{ and } \gamma \text{ have the same parity,} \\ \beta\gamma(\gamma^2 + \beta^2 + 1)/12 & \text{otherwise.} \end{cases} \tag{9.8}$$

In order to obtain the right hand side of the complete-bipartite constraints, assume w.l.o.g. that $|N'_i| = \beta$. Placing each node v of N'_i beneath the center of the compactly positioned nodes N'_{i+1}, we would attain $\beta \cdot \lfloor \gamma/2 \rfloor \cdot \lceil \gamma/2 \rceil$ overall horizontal distances, according to the degree constraints' right hand side. Yet, not all nodes N'_i can be placed at the same center positions (or at one of the two center positions for γ even); they have to be tightly grouped around the center. Each shift of a node by one position further away from center position adds 1 to the overall horizontal distances of its incident edges. By case distinction based on the parity of β and γ we obtain the formula. E.g., if both cardinalities are odd, there are exactly two nodes being shifted by i positions, for $1 \leq i \leq \lfloor \beta/2 \rfloor$, resulting in an additional overall sum of horizontal distances of $2 \cdot \frac{1}{2} \cdot \lfloor \beta/2 \rfloor \cdot (\lfloor \beta/2 \rfloor + 1) = \lfloor \beta/2 \rfloor \cdot \lceil \beta/2 \rceil$.

Analogously, we can sum the minimally occurring non-verticalities as

$$\sum_{i=1}^{\beta} \sum_{j=1}^{\gamma} (\lfloor (\gamma - \beta)/2 \rfloor + i - j)^2.$$

After transforming this, using the above formula for the sum of increasing squares and considering a case distinction on the parities of β and γ, this constitutes the right hand side of the quadratic complete-bipartite constraints (9.8).

Next let us point out polyhedral properties of degree and complete-bipartite constraints.

Lemma 9.6 *Degree constraints (even in their weaker form* (9.5)*) strengthen* `(cDM')`*. Complete-bipartite constraints (even in their weaker form* (9.7)*) further strengthen the relaxation, even if it already satisfies all degree constraints* (9.6)*.*

Proof. Observe that the 3-cycle constraints allow to set all x'-variables to 0.5. Hence all horizontal node positions are identical, all d'-variables can be 0, and the relaxation has an optimal solution of 0, as well. As long as there is at least one node with two neighbors on the level above (or below), the corresponding degree constraint forces the associated d'-variables to be non-zero and increase the solution value.

Assume we have a solution feasible w.r.t. `(DM')` and all degree constraints (9.6); again all x'-variables can be 0.5. Let u_1, u_2 be two nodes on a common level, both adjacent to three nodes v_1, v_2, v_3 on an adjacent level. The degree constraints essentially only require as much horizontal distance (or non-verticality) as achieved by centering both u_1, u_2 at the same center position. Applying the complete-bipartite constraint for this structure hence requires larger values for the corresponding d'-variables, raising the objective value. □

Non-Proper `(MLVO)`**.** We can directly rewrite `(DM')` to non-proper level graphs, obtaining the formulation `(DM)`: we order all nodes V level-wise via variables x, and measure the non-verticality of the edges

E via variables d, using $\omega := \max_{1\le i\le p}|V_i|$, $\delta_i := \lfloor(\omega - |V_i|)/2\rfloor$, and $X(u) := \delta_{\ell(u)} + \sum_{v\in V_{\ell(u)}} x_{vu}$:

$$v_* = \min\Big\{ \sum\nolimits_{e\in E} (d_e)^2 : 0 \le x_{uv} + x_{vw} - x_{uw} \le 1,\ X(a) - X(b) \le d_{(a,b)} \ge X(b) - X(a) \\ x_{uv} \in \{0,1\},\quad 1 \le i \le p,\ u,v,w \in V_i,\ u \dot< v \dot< w,\ (a,b) \in E\Big\}. \tag{DM}$$

Analogously to above, replacing the integrality conditions with 0-1 bounds gives (cDM). We can again strengthen (cDM) via corresponding degree and complete-bipartite constraints. Observe that for the former, it suffices that the nodes N lie on *some* common level, not necessarily a neighboring level. Similarly, the node sets for the latter constraints do not have to be on neighboring levels, i.e., we have $N_i \subseteq V_i$, $N_j \subseteq V_j$, for some $1 \le i < j \le p$, and $N_i \times N_j \subseteq E$. Notice that we can also define (OM) and its relaxation (cOM) analogously to above.

9.6 Exact Approaches Based on Linear Programming

Modern mathematical programming software can often already deal with models with linear constraints and quadratic objective functions. Yet, one naturally may try to linearize the models. The ILPs for (MLCM), e.g., can be seen as linearized models from the originally quadratic problem (8.3), and they outperform SDP approaches for sparse graphs with density $\le 10\%$ (for details see Sections 14.2–14.4). Yet, we observe that thereby only a few products (especially for sparse graphs) of two binary variables have to be linearized.

In our first model (DM'), we have squares of arbitrary integers, only bounded by $\omega' - 1$. We can linearize any $(d'_e)^2$ by adding variables $d'_{e,i} \ge 0$ and requiring $d'_{e,i} \ge d'_e - i$, for all $1 \le i < \omega' - 1$. The objective function then becomes $\sum_{e\in E'}(d'_e + 2d'_{e,1} + 2d'_{e,2} + \ldots)$.

In order to obtain an ILP from our second model (OM'), we would have to linearize $\approx \sum_{1\le i<p} \binom{|V'_i|}{2}\binom{|V'_{i+1}|}{2}$ products of two binary variables. This number can be compared to (MLCM) on completely dense graphs, for which, e.g., Table 14.2 shows that the SDP clearly outperforms the ILP.

For non-proper level graphs, the situation turns out to be even worse when considering the linearization of (OM) because the cost matrix D is completely dense. The clearly resulting drawback is also supported by the results in [30, Table 2].

Using the extended d-variables (and letting $d'_{\cdot,0} := d'_{\cdot}$ for notational simplicity), we can linearize the quadratic degree constraints (9.6) as

$$\sum\nolimits_{v\in N'} d'_{(u,v),i} \ge \lfloor\alpha/2 - i\rfloor \cdot \lceil\alpha/2 - i\rceil, \qquad 0 \le i < \lfloor\alpha/2\rfloor. \tag{9.9}$$

To obtain the right hand side we ask for the extended variables $d_{e,i}$ that their sum is greater or equal the sum of distances reduced by i.

Furthermore, we can also linearize the quadratic complete-bipartite constraints as

$$\sum_{u\in N_i, v\in N_j} d'_{(u,v),i} \ge \beta \cdot \lfloor\gamma/2 - i\rfloor \cdot \lceil\gamma/2 - i\rceil + \begin{cases} \lfloor\beta/2\rfloor \cdot \lceil\beta/2\rceil, & \text{if } \gamma \text{ odd}, 0 \le i < \lfloor\gamma/2\rfloor, \\ \lfloor\beta/2\rfloor \cdot (\lceil\beta/2\rceil - 1), & \text{if } \gamma \text{ even}, 0 \le i < \lfloor\gamma/2\rfloor. \end{cases} \tag{9.10}$$

Requiring for the extended variables $d_{e,i}$ that their sum is greater or equal the sum of distances reduced by i and simplifying the resulting expression by using (9.7) gives the right hand side of the above constraints.

9.7 Exact Approaches Based on Semidefinite Programming

Motivated by the strong theoretical properties and practical performance of the SDP approach for (MLCM) (for details see Section 8.4 and Chapter 14 respectively), we also apply the semidefinite relaxations (SDP$_I$)–

(SDP$_{IV}$) (see Section 8.3) for the bound computation of (Non-)Proper (MLVO).[2] Let us write $\mathfrak{d}(E)$ as a linear-quadratic function in y as

$$\mathfrak{d}(E) = \sum_{(u,v)\in E} (X(u) - X(v))^2 = \sum_{(u,v)\in E} \frac{1}{4} \left[\left(\sum_{\substack{t\in V_{\ell(u)} \\ t\neq u}} y_{ut} + g_{\ell(u)} \right) - \left(\sum_{\substack{w\in V_{\ell(v)} \\ w\neq v}} y_{vw} + g_{\ell(v)} \right) \right]^2 \stackrel{!}{=} \langle C, Y\rangle + c^\top y + K,$$

where $g_{\ell(u)} := (\omega - |V_{\ell(u)}|) \mod 2$. Expanding and using $y_{uv}^2 = 1$ yields

$$\mathfrak{d}(E) = \sum_{(u,v)\in E} \frac{1}{4} \left[(g_{\ell(u)} - g_{\ell(v)})^2 + |V_{\ell(u)}| + |V_{\ell(v)}| - 2 + 2 \left((g_{\ell(u)} - g_{\ell(v)}) \sum_{\substack{t\in V_{\ell(u)} \\ t\neq u}} y_{ut} + (g_{\ell(v)} - g_{\ell(u)}) \sum_{\substack{w\in V_{\ell(v)} \\ w\neq v}} y_{vw} + \sum_{\substack{t,w\ \in V_{\ell(u)},\ t\dot{<}w \\ t\neq u,\ w\neq u}} y_{ut}y_{uw} + \sum_{\substack{t,w\ \in V_{\ell(v)},\ t\dot{<}w \\ t\neq v,\ w\neq v}} y_{vt}y_{vw} - \sum_{\substack{t\in V_{\ell(u)},\ t\neq u \\ w\in V_{\ell(v)},\ w\neq v}} y_{ut}y_{vw} \right) \right] \stackrel{!}{=} \langle C, Y\rangle + c^\top y + K. \tag{9.11}$$

Now (9.11) can be directly applied to specify the cost matrix C, the cost vector c and the constant K for (MLVO). Along the lines of Theorem 8.2 we can prove that (MLVO) is equivalent to the problem

$$v_* = \min \{\ \langle C, Y\rangle\ + c^\top y + K :\ Z \in \mathcal{I}_{MQO}\ \}. \tag{MLVO}$$

In the following section we compare the linear, quadratic and semidefinite relaxations regarding their theoretical tightness and furthermore motivate the choice of our SDP relaxation for the practical experiments in Chapter 15.

9.8 Some Polyhedral Results

Let us start by relating the semidefinite relaxation (SDP$_I$) to the quadratic programming relaxation (cDM) incorporating degree and complete-bipartite constraints.

Theorem 9.7 (SDP$_I$) *is at least as strong as* (cDM) *together with the quadratic degree constraints* (9.6) *and quadratic complete-bipartite constraints* (9.8)*.*

Proof. First, it is not hard to verify that any Z feasible for (SDP$_I$) contains a vector y in its first column that satisfies the 3-cycle inequalities (8.7) on the levels. This follows from the semidefiniteness of the following submatrices of Z

$$\begin{pmatrix} 1 & y_{uv} & y_{uw} & y_{vw} \\ y_{uv} & 1 & y_{uv,uw} & y_{uv,vw} \\ y_{uw} & y_{uw,uv} & 1 & y_{uw,vw} \\ y_{vw} & y_{vw,uv} & y_{vw,uw} & 1 \end{pmatrix}, \quad u,v,w \in V,\ u \dot{<} v \dot{<} w.$$

Constraints (9.3) are implicitly ensured by the definition of C, c and K through (9.11). Next let $N_i \subseteq V_i$ and $N_j \subseteq V_j$, for some $1 \leq i < j \leq p$, be two node sets such that $N_i \times N_j \subseteq E$. Applying (9.11) for

[2] Both cases are virtually identical for the SDP approach. For notational simplicity, we will use the variable naming scheme of the non-proper setting.

$\beta := \min\{|N_i|, |N_j|\}$ and $\gamma := \max\{|N_i|, |N_j|\}$ with $\gamma - \beta$ even to the left hand side of (9.8) yields

$$\begin{aligned}
\sum_{\substack{u\in N_i,\\ v\in N_j}} d^2_{(u,v)} &= \frac{1}{4} \sum_{\substack{u\in N_i,\\ v\in N_j}} \left[\beta + \gamma - 2 + 2 \left(\sum_{\substack{t,w\in N_i,\ t\dot{<}w\\ t\neq u,\ w\neq u}} y_{ut}y_{uw} + \sum_{\substack{t,w\in N_j,\ t\dot{<}w\\ t\neq v,\ w\neq v}} y_{vt}y_{vw} - \sum_{\substack{t\in N_i,\ t\neq u\\ w\in N_j,\ w\neq v}} y_{ut}y_{vw} \right) \right] = \\
&\frac{\beta\gamma(\beta+\gamma-2)}{4} + \frac{1}{2} \sum_{\substack{u\in N_i,\\ v\in N_j}} \left(\sum_{\substack{t,w\in N_i,\ t\dot{<}w\\ t\neq u,\ w\neq u}} y_{ut}y_{uw} + \sum_{\substack{t,w\in N_j,\ t\dot{<}w\\ t\neq v,\ w\neq v}} y_{vt}y_{vw} - \sum_{\substack{t\in N_i,\ t\neq u\\ w\in N_j,\ w\neq v}} y_{ut}y_{vw} \right) = \\
&\frac{\beta\gamma(\beta+\gamma-2)}{4} + \frac{1}{2} \sum_{v\in N_j} \left(\sum_{u\dot{<}t\dot{<}w\in N_i} y_{ut}y_{uw} - \sum_{t\dot{<}u\dot{<}w\in N_i} y_{ut}y_{uw} + \sum_{t\dot{<}w\dot{<}u\in N_i} y_{ut}y_{uw} \right) + \\
&\frac{1}{2} \sum_{u\in N_i} \left(\sum_{v\dot{<}t\dot{<}w\in N_j} y_{vt}y_{vw} - \sum_{t\dot{<}v\dot{<}w\in N_j} y_{vt}y_{vw} + \sum_{t\dot{<}w\dot{<}v\in N_j} y_{vt}y_{vw} \right) - \\
&\frac{1}{2} \left(\sum_{\substack{u\dot{<}t\in N_i,\\ v<w\in N_j}} y_{ut}y_{vw} - \sum_{\substack{t\dot{<}u\in N_i,\\ v<w\in N_j}} y_{tu}y_{vw} - \sum_{\substack{u\dot{<}t\in N_i,\\ w<v\in N_j}} y_{ut}y_{wv} + \sum_{\substack{t\dot{<}u\in N_i,\\ w<v\in N_j}} y_{tu}y_{wv} \right).
\end{aligned} \tag{9.12}$$

The terms in the last line of (9.12) cancel each other. Summing up (8.10) for all elements in N_i and N_j, respectively, and applying it to (9.12) gives

$$\frac{\beta\gamma(\beta+\gamma-2)}{4} + \frac{\beta\gamma(\beta-1)(\beta-2)}{12} + \frac{\beta\gamma(\gamma-1)(\gamma-2)}{12} = \frac{\beta\gamma(\gamma^2+\beta^2-2)}{12}.$$

Applying (9.11) for $\gamma - \beta$ odd to the left hand side of (9.8) yields

$$\begin{aligned}
\sum_{u\in N_i, v\in N_j} d^2_{(u,v)} &= \frac{1}{4} \sum_{u\in N_i, v\in N_j} (\beta+\gamma-1) + \frac{1}{2} \sum_{u\in N_i, v\in N_j} \left(\sum_{\substack{t,w\in N_i,\ t\dot{<}w\\ t\neq u,\ w\neq u}} y_{ut}y_{uw} + \sum_{\substack{t,w\in N_j,\ t\dot{<}w\\ t\neq v,\ w\neq v}} y_{vt}y_{vw} \right) + \\
&\frac{1}{2} \sum_{u\in N_i, v\in N_j} \left(\pm \sum_{\substack{t\in N_i\\ t\neq u}} y_{ut} \mp \sum_{\substack{w\in N_j\\ w\neq v}} y_{vw} - \sum_{\substack{t\in N_i,\ t\neq u\\ w\in N_j,\ w\neq v}} y_{ut}y_{vw} \right).
\end{aligned} \tag{9.13}$$

Again the three double sums in the second line of (9.13) give 0. Summing up (8.10) for all elements in N_i and N_j, respectively, and applying it to (9.13) gives

$$\frac{\beta\gamma(\beta+\gamma-1)}{4} + \frac{\beta\gamma(\beta-1)(\beta-2)}{12} + \frac{\beta\gamma(\gamma-1)(\gamma-2)}{12} = \frac{\beta\gamma(\gamma^2+\beta^2+1)}{12}.$$

As the degree constraints are special complete-bipartite constraints with $\beta = 1$, they are also satisfied on any matrix feasible for (SDP_I). □

In summary, (SDP_I) is required to ensure all constraints proposed for the linear and quadratic relaxations. For our practical experiments we work with the stronger semidefinite relaxation (SDP_{IV}). The additional constraint types (8.12) and (8.13) further strengthen the relaxation without making it incomputable. e.g. (8.12) is necessary to solve graphs to optimality that only contain the edges required for a degree constraint with $\alpha = 4$ (or, more generally, graphs that only contain the edges required for complete-bipartite constraints with $\gamma - \beta = 3$), where the smaller level is filled up with PDs. For solving analogous graphs exactly with $\gamma - \beta > 3$ odd, we would have to consider additional clique inequalities of size > 3 odd in the relaxation. As separating them is far too expensive, this supports our model choice. In Section 7.3 we propose an approach that seems to partially avoid this limitation by heuristically selecting the most important pentagonal inequalities. Yet, additional experiments are needed to successfully incorporate this method in our SDP approach.

9.9 Extensions

Edge-weights and different drawing areas: In all the above approaches, including the SDP, it is straight forward to allow edge-weights. These can be used to model edges which are more important to be drawn relatively vertical than others, or to penalize non-verticalities for long edges more than for short ones (or vice versa) in the non-proper drawing scheme.

In practice, it can be interesting to consider other outer shape drawings than the rectangular array dominated by the width of the largest layer. Clearly, it is trivial to allow wider drawings, potentially resulting in less overall non-verticality by adding more PDs to the layers. Similarly, we can approximate any convex shape (e.g. a circlic disc) by adding fewer or more PDs to the layers and shifting the first x-coordinate per layer via an offset, as suitable. We can model more general drawing shapes, including holes, by occupying any forbidden position q with a fixed-position PD u (yet note that edges may still be routed close to these positions) by asking

$$\sum_{\substack{v \in V_{\ell(u)} \\ v \neq u}} y_{uv} = \omega + 1 - 2q + g_{\ell(u)}. \tag{9.14}$$

We can further strengthen the semidefinite relaxation by incorporating the linear-quadratic constraints obtained from multiplying (9.14) with an arbitrary ordering variable $y_{st}, s \dot{<} t \in V_i,\ 1 \leq i \leq p$.

Monotonous drawings: Considering drawings optimal w.r.t. `(MLVO)`, we may want to force an additional monotonicity property. Within the Sugiyama framework, each edge is drawn using only strongly monotonously increasing y-coordinates. We say a drawing is monotonous, if all original edges are weakly monotonous along the x-axis. More formally, let $e = (u, v) \in E$ be any edge in the (non-proper) level graph G, $e_1 = (u = u_0, u_1), e_2 = (u_1, u_2), ..., e_k = (u_{k-1}, u_k = v)$ the corresponding chain of edges in G', and $x : V' \to \mathbb{N}$ the mapping of nodes to x-coordinates in the final drawing. Then a drawing is monotonous, if $x(u) \leq (\geq)\ x(v)$ implies $x(u_i) \leq (\geq)\ x(u_{i+1})$ for all $0 \leq i < k$.

In the non-proper drawing style we already observed that all edges are drawn monotonously along the x-coordinate, but this is not necessarily the case for proper drawings. We may, however, explicitly ask for this property to hold, giving rise to the monotonous `(MLVO)` problem.

While such a requirement is complicated to efficiently implement within our heuristic schemes, it is simple to include in the SDP approach. Conceptually, we require that, for all pairs of consecutive edge segments, their horizontal differences Δ_i, Δ_{i+1} do not have different signs, i.e., $\Delta_i \cdot \Delta_{i+1} \geq 0$.

To be more precise, let $e = (u, v) \in E$ be any original edge in the (non-proper) level graph G spanning k levels with the corresponding edge chain along the nodes $\langle u = u_0, u_1, u_2, \ldots, u_k = v \rangle$ in G'. Monotonicity of the edge is equivalent with feasibility of the following system of inequalities

$$[x(u_{i+1}) - x(u_i)][x(u_{i+2}) - x(u_{i+1})] \geq 0,\ i \in \{0, \ldots, k-2\}.$$

Now using

$$x\left(u_i\right) = -\frac{1}{2} \sum_{\substack{v \in V_{m+i} \\ v \neq u_i}} y_{u_i v} + \frac{\omega + 1}{2} - \frac{g_{\ell(u_i)}}{2},$$

yields the following constraints on Z

$$\begin{aligned}
&\sum_{\substack{v\in V_{\ell(u_{i+1})}\\ v\neq u_{i+1}}}\sum_{\substack{w\in V_{\ell(u_{i+2})}\\ w\neq u_{i+2}}} y_{u_{i+1}v}y_{u_{i+2}w} - \sum_{\substack{v\in V_{\ell(u_{i+1})}\\ v\neq u_{i+1}}}\sum_{\substack{w\in V_{\ell(u_{i+1})}\\ w\neq u_{i+1}}} y_{u_{i+1}v}y_{u_{i+1}w} -\\
&\sum_{\substack{v\in V_{\ell(u_{i})}\\ v\neq u_{i}}}\sum_{\substack{w\in V_{\ell(u_{i+2})}\\ w\neq u_{i+2}}} y_{u_{i}v}y_{u_{i+2}w} + \sum_{\substack{v\in V_{\ell(u_{i})}\\ v\neq u_{i}}}\sum_{\substack{w\in V_{\ell(u_{i+1})}\\ w\neq u_{i+1}}} y_{u_{i}v}y_{u_{i+1}w} +\\
&\big(g_{\ell(u_{i+1})} - g_{\ell(u_i)}\big)\left(\sum_{\substack{v\in V_{\ell(u_{i+2})}\\ v\neq u_{i+2}}} y_{u_{i+2}v} - \sum_{\substack{v\in V_{\ell(u_{i+1})}\\ v\neq u_{i+1}}} y_{u_{i+1}v}\right) +\\
&\big(g_{\ell(u_{i+2})} - g_{\ell(u_{i+1})}\big)\left(\sum_{\substack{v\in V_{\ell(u_{i+1})}\\ v\neq u_{i+1}}} y_{u_{i+1}v} - \sum_{\substack{v\in V_{\ell(u_{i})}\\ v\neq u_{i}}} y_{u_{i}v}\right) +\\
&\big(g_{\ell(u_{i+1})} - g_{\ell(u_i)}\big)\big(g_{\ell(u_{i+2})} - g_{\ell(u_{i+1})}\big) \geq 0,\ i\in\{0,\dots,k-2\}.
\end{aligned} \tag{9.15}$$

We can further strengthen the semidefinite relaxation by additionally generalizing (9.15) for nodes on non-adjacent layers

$$[x(u_i) - x(u_h)][x(u_l) - x(u_j)] \geq 0,\ h < i, j < l \in \{0,\dots,k\}.$$

Node sizes: In many real-world scenarios, it can be interesting to consider nodes of varying size. Before, any node required exactly one grid point; generally, we may introduce nodes requiring $d_x \times d_y$ grid points. A horizontal stretch is easy to incorporate: when considering the absolute grid position of a node we not only compute the number of nodes to its left, but the sum of their horizontal stretches. To incorporate vertical stretches, we copy the node on all its respective layers and connect them from layer to layer with dummy edges. Now, we only generate solutions where these dummy edges are strictly vertical and not crossed, both of which can be achieved in the SDP straight-forwardly.

To incorporate a vertical node u spanning $k+1$ levels we first make sure that the starting and end point of the node u_0 and u_k and according PDs $u_1,\dots,u_{k-1}$ on the layers in between have the same x-coordinate. Second we ensure that no edge (w_0, w_1) is crossing u by the following constraints

$$[x(w_0) - x(u)][x(w_1) - x(u)] \geq 0,\ (w_0,w_1)\in E,\ \ell(u_0) < \ell(w_1) \leq \ell(u_1).$$

`(MLVO)` after `(MLCM)`: Our `(MLVO)` SDP cannot only be used directly after the Sugiyama's first stage, but we can also apply it after a second stage crossing minimization, i.e., after solving `(MLCM)`. By fixing the order of the original nodes (non-PDs), the SDP becomes an exact quadratic compactor for Sugiyama's third stage. Such a fixing can be achieved either by dropping the fixed variables altogether (see Proposition 4.4) and corresponding modifications to the constraint matrix, or by introducing equality constraints on the respective variables. In our experiments, we used the latter approach due to code simplicity. Implementing the reduction strategy would assumingly lead to further improved running times.

To fix the order of the original nodes in the SDP relaxation, we introduce the relative position function $q : V \to \{0,1,\dots,\omega\}$, where $q(u) = 0$ means that the relative position of node u is not fixed. We ask for the following constraint to hold for two nodes $u, v \in V_i,\ u \dot{<} v,\ 1 \leq i \leq p$ with $q(u) > 0, q(v) > 0$

$$y_{uv} = 1,\ \text{if } q(u) < q(v), \qquad y_{uv} = -1,\ \text{if } q(u) > q(v). \tag{9.16}$$

We can further strengthen the semidefinite relaxation by adding linear-quadratic constraints that we get from multiplying (9.16) with an arbitrary ordering variable $y_{st}, s \dot{<} t \in V_i,\ 1 \leq i \leq p$.

9.10 Some Complexity Results

Consider the decision variant of (MLVO), i.e., given some value M we ask whether there exist node orderings such that the obtained non-verticality is at most M.

Theorem 9.8 *The decision variant of* (MLVO) *is NP-complete, already when considering only two levels.*

Proof. We reduce from the NP-complete PARTITION problem, i.e., given a set of n numbers $a_1, \ldots, a_n \in \mathbb{N}$ with $\sum_{1 \leq i \leq n} a_i = 2B$, does there exist a partition of these numbers such that the sum in both subsets is exactly B? For details on the complexity of the PARTITION problem see Subsection 3.1.5 of Garey and Johnson [81] or the original paper of Karp [125].

Consider the following (MLVO) instance with two levels, arising from some PARTITION instance. For each a_i, $1 \leq i \leq n$, we introduce a_i many vertices $U_i = \{v_{i,1}, \ldots, v_{i,a_i}\}$ to V_1 and analogously a_i many vertices $U'_i = \{v'_{i,1}, \ldots, v'_{i,a_i}\}$ to V_2. Then we connect every vertex of U_i with every vertex U'_i. Finally, we add two additional vertices $t \in V_1$ and $t' \in V_2$, and connect all vertices of V_2 with t. We now have $2B + 1$ vertices on each level.

An (MLVO) solution is cleary optimal if it achieves the following two properties:

1. The node t is on position $B + 1$, independent of the ordering of the vertices in V_2: since the width of both levels is equal and odd and t is adjacent to all vertices in V_2 any other position would result in strictly larger non-verticalities.

2. Consider the nodes U_i, U'_i corresponding to some number a_i. Their optimal arrangement is to tightly pack all vertices of U_i (U'_i, respectively) horizontally, and U_i being vertically exactly below U'_i. Any other arrangement would result in strictly larger non-verticalities.

Now, we define M as the non-verticality caused by a solution fulfilling both properties.[3] If a solution with non-verticality M is achievable, then the node t partitions the numbers of the PARTITION instance into two groups with equal sum B: for any a_i, all its nodes are either left or right of t (as they are tightly packed), and there are exactly B vertices to the left and to the right of t. Vice versa, if the PARTITION instance is satisfiable, then an (MLVO) solution with non-verticality M exists. □

Next we show that also a modified variant of (MLVO) is NP-hard.

Theorem 9.9 *The decision variant of bipartite* (MLVO)*, where level 1 is fixed but the vertices on level 2 have arbitrary vertical stretch, is strongly NP-complete.*

Proof. We reduce from the strongly NP-complete 3-PARTITION problem, i.e. given a set A of $3m$ numbers $a_1, \ldots, a_{3m} \in \mathbb{N}$ with $\sum_{1 \leq i \leq 3m} a_i = mB$, do there exist m disjoint subsets $S_1, \ldots, S_m$ of A such that the sum of the numbers of each subset is exactly B? For details on the complexity of the 3-PARTITION problem see Subsection 4.2.2 of Garey and Johnson [81] or the original paper of Garey and Johnson [80].

Consider the following (MLVO) instance with two levels, arising from some 3-PARTITION instance. For each a_i, $1 \leq i \leq n$, we introduce a vertex with horizontal stretch a_i to V_2. Then we add $2m - 2$ further vertices $t_i \in V_1,\ u_i \in V_2,\ 1 \leq i \leq m - 1$, with horizontal stretch 1 and connect t_i with u_i. Finally we introduce mB PDs to V_1 such that the width of both levels is $(B + 1)m - 1$ and fix $t_i,\ 1 \leq i \leq m - 1$, at position $iB + 1$.

An (MLVO) solution is clearly optimal if the nodes $u_i,\ 1 \leq i \leq m - 1$, are located at the positions $iB + 1$ as the non-verticality of such a solution is zero. Hence we set M to zero. Now, if a solution with non-verticality zero is achievable, then the vertices $u_i,\ 1 \leq i \leq m - 1$, partition the numbers of the 3-PARTITION instance into m groups with equal sum B. Vice versa, if the 3-PARTITION instance is satisfiable, then an (MLVO) solution with non-verticality zero exists. □

[3] We can use the quadratic degree and complete-bipartite constraints to state M explicitly.

Finally we show that restricting the horizontal stretch of all vertices to 1 and fixing one of two levels results in a problem that can be solved in polynomial time.

Theorem 9.10 *Bipartite* (MLVO)*, where level 1 is fixed and the horizontal stretch of all vertices is set to 1, can be solved in* $O(n^3)$ *running time with* $n = |V_2|$.

Proof. The non-verticality caused by vertex $v_i \in V_2$ depends only on its position on level 2 and does not depend on the positions of all other vertices in V_2. Thus we can independently compute the non-verticality of vertex i when located at position j for all $1 \leq i, j \leq n$ and store the values as the i-j-entries of an $n \times n$ matrix N. Hence finding an optimal ordering for this variant of (MLVO) is equivalent to solving the Linear Assignment Problem on matrix N. This can be done in polynomial time, e.g. in $O(n^3)$ running time with the Hungarian method [134, 160]. □

Notice that Theorems 9.8–9.10 hold true if we compute the non-verticality as the sum of the *linear* horizontal distances of all edges.

In summary, we have shown that a severely restricted variant of (MLVO) can be solved in polynomial time contrary to the NP-hardness of the crossing minimization problem with the same restrictions (for details see [69]). But if we allow arbitrary horizontal stretches for the vertices on the level not fixed or if we optimize over more than one level, the problem becomes NP-hard.

9.11 Applications Beyond Graph Drawing

We want to conclude with noting that (MLVO) can also be directly applied to other seemingly very different problem classes unrelated to graph drawing: Consider a scheduling problem with multiple machines, where each machine has multiple pre-assigned jobs. The jobs are related to each other in such a way that certain jobs should be finished at similar times. Modeling machines as levels, jobs as nodes, time as horizontal coordinates, and job relations as edges, we directly obtain a Non-proper (MLVO) problem.

Another application, also giving a (Non-)Proper (MLVO) instance can be found in multiple ranking, where we have groups of objects, objects have relationships (e.g., similarities) with objects from other groups, and we want to (linearly) rank the objects within their groups such that related objects are ranked similarly over all groups. This can be seen as a generalization of maximum weight matchings, where the relative positions of all objects are considered in a quadratic cost setting.

In the following, we will discuss some problem variants less abstractly, with the focus on showcasing the problem's versatility: As a tongue-in-cheek example, consider a restaurant that offers a menu which lists food categories (e.g., soup, main dish, side dish, etc.) and allows to choose one or more kinds per category (e.g., the main dish may be steak, turkey, or fish). After the guests have ordered, the following problem arises: Although the restaurant has one cook per food category, each cook wants to prepare all items of the same kind (e.g., all ordered steaks), before preparing a different kind (e.g., before preparing fish). Assume that we do not want the guests to wait long between separate courses, and recognize that, e.g., the main dish should always be accompanied with the side dish at the same time. In which order should the cooks prepare their items (kinds, in fact), such that the guests get their menu with all kinds being reasonably warm/fresh?

This question obviously leads to a weighted (possibly Non-proper) (MLVO) problem (with wide alignment scheme) where the categories are levels, and the kinds are nodes. Notice that weights can be directly added to our ILP/SDP approaches. The quadratic cost structure reflects the preference to accept several small delays rather than some big ones. While this example seems far fetched, or course, it can be seen as a naïve interpretation of the following problem in logistics:

Consider a worker at a storehouse, who has to pack items onto pallets. Each pallet is a separate purchase (we omit the term "order" to avoid confusion) of multiple, prespecified items. Within the storehouse, items are categorized by coarse type (e.g., heavy, small, electronics, etc.) and stored at different locations,

according to this type. Now, we have a conveyor belt (or forklift) for each such storage location serving the worker items of the corresponding type. Whenever an item arrives at the worker, he packs it onto the corresponding pallet. Our goal is that each purchase is packed within a small timeframe, and hence the worker does not have to deal with many started-but-incomplete purchases/pallets simultaneously. By modeling the items as nodes on levels corresponding to their respective item type, we again obtain an `(MLVO)` problem.

Finally we examine team-building, a problem in business studies. Consider a company that wants to build interdisciplinary teams, taking the team members' preferences into account. The different disciplines involved (like cost accounting, financing, or taxation) are the levels, the employees are the nodes, and weighted edges represent the preferences of employees for collaborations. This gives a weighted, Non-proper `(MLVO)` problem with wide alignment scheme. The optimal solution of the multiple ranking can guide the chief executives in their final team-building decisions. For example, employees in the center generally have higher esteem and could be chosen as team leaders; employees far away from each other should not be in the same team. The quadratic cost structure reflects the common notion of fairness, i.e., we prefer to violate multiple preferences slightly, than to violate some very strongly.

Part III

Experiments & Outlook

Chapter 10

The Linear Ordering Problem

In the following we compute the linear and semidefinite programming relaxations introduced in Chapter 4 on well-known benchmark instances for the Linear Ordering Problem (LOP) from the literature. The first two sections of this chapter are based on Sections 5, 6 and 7 of the paper "Semidefinite Relaxations of Ordering Problems" [115]. In the first section we apply the relaxations to facets of the linear ordering polytope $\mathcal{P}_{LOP}$ in small dimensions to get an idea of their "practical" strength. In Section 10.2 we compute approximate SDP bounds for medium and large instances that are notoriously hard for linear relaxations. Finally in Section 10.3 we will discuss a method that allows us to considerably speed up the solution of the basic linear programming relaxation (LP_{LOP}) for very large instances.

10.1 Small Facets

As a first experiment we consider the complete outer description of $\mathcal{P}^n_{LOP}$ for $n \in \{6, 7\}$, and try to recover the correct right hand side of the facet classes (with respect to node permutations). We compare the linear programming relaxation (LP_{LOP}) from Section 4.2 to the semidefinite relaxations (SDP_1)–(SDP_4) from Section 4.3.

In Table 10.1 we examine all nontrivial facet classes of the linear ordering polytopes for 6 and 7 nodes. The facet classes are collected under http://comopt.ifi.uni-heidelberg.de/software/SMAPO/lop/lop.html. We also use the same labeling, see column 1. As usual, n denotes the number of nodes, and "opt" gives the optimal solution. All relaxations are solved to optimality using the standard settings of Sedumi [199]. We also include the combinatorial instances Paley 11 and Paley 19, which are notoriously difficult for linear relaxations.

From Table 10.1 we conclude that the triangle inequalities (4.12) and the Lovász-Schrijver cuts (4.13) are incomparable, as there are instances where (SDP_2) is tighter than (SDP_3) and vice versa. The basic relaxation (SDP_1) improves upon the pure linear model, but does not give the correct facet classes for $n = 6$. Adding either the triangle inequalities or the Lovász-Schrijver cuts gives the correct facet classes for $n = 6$. Furthermore the full model (SDP_4) identifies all except one of the facet classes correctly for $n = 7$.

As a first conclusion we observe that the semidefinite approach provides a substantial improvement over the linear approach in the approximation of $\mathcal{P}_{LOP}$ in small dimensions.

10.2 Medium and Large Instances

Solving (SDP_4) with interior-point methods (IPMs) (for details see Section 3.2) for problems of size $n \geq 20$ gets computationally far too expensive, because of the large number ($O(n^6)$) of inequality constraints.

facet class	n	opt	($\mathrm{LP}_{\mathrm{LOP}}$)	(SDP_1)	(SDP_2)	(SDP_3)	(SDP_4)
FC3	6	7	7.5	7.35	7	7	7
FC4, 5	6	8	8.5	8.35	8	8	8
FC3	7	7	7.5	7.35	7	7	7
FC4, 20	7	8	8.5	8.35	8	8	8
FC5	7	9	9.5	9.37	9	9.09	9
FC6	7	9	9.5	9.37	9	9	9
FC10, 25	7	9	9.5	9.37	9.06	9	9
FC21	7	9	9.5	9.37	9	9.01	9
FC7, 9, 22, 24	7	10	10.5	10.37	10.11	10	10
FC8, 13, 23	7	10	10.5	10.37	10.19	10	10
FC11	7	10	10.5	10.37	10	10	10
FC12	7	10	10.5	10.37	10	10.03	10
FC14	7	10	10.5	10.35	10.35	10.24	10.22
FC15, 16	7	11	11.5	11.37	11.22	11	11
FC26	7	11	11.5	11.37	11.23	11	11
FC17, 27	7	13	13.5	13.40	13	13	13
FC18	7	14	14.5	14.40	14.17	14.04	14
FC19	7	14	14.5	14.40	14.10	14.01	14
Paley	11	35	36.67	36.03	36.03	35.92	35.92
Paley	19	107	114	110.70	110.70	110.50	110.50

Table 10.1: Improvements of various semidefinite relaxations as compared to the linear programming relaxation ($\mathrm{LP}_{\mathrm{LOP}}$) on facet classes of the linear ordering polytope for 6 and 7 nodes and on the notoriously difficult Paley instances. n gives the number of vertices and "opt" denotes the optimal solution. While all semidefinite relaxations for the facet classes can be solved within a few seconds, it already takes about 10 minutes to solve (SDP_4) for the instance Paley 19.

Thus to obtain solutions of (SDP$_4$) also for larger instances, say $n \approx 100$, we apply a dynamic version of the bundle method (for details see Section 3.3) to the partial Lagrangian dual, obtained by dualizing the 3-cycle equalities (4.11), the triangle inequalities (4.12) and the Lovász-Schrijver cuts (4.13). Thus a function evaluation of the bundle method amounts to solving an SDP over the elliptope (4.9). We use standard primal-dual path-following (IPMs) (see e.g. see [105]) to do these function evaluations. In Table 10.2 we summarize the running times for solving such SDPs of different dimensions on an Intel Xeon 5160 processor with 3 GHz and 2 GB RAM.

n	ζ	time
30	436	3
50	1226	40
70	2416	500
100	4951	3000

Table 10.2: Average computation times (in seconds) using (IPMs) to solve an SDP over the elliptope, where n is the number of objects and the primal matrix variable Z is of order ζ.

In fact these function evaluations constitute the computational bottleneck of the dynamic bundle method applied to our applications as they are always responsible for more than 95% of the total required running time.

Let us give some further details on our parameter settings of the bundle method that we maintain for the computations in the following four chapters. After every fifth function evaluation we search for newly violated constraints at the current primal point. We add all constraints with violation > 0.001 to the bundle and additionally remove constraints with relatively speaking small associated Lagrangian multipliers ($\lambda_i < 0.05 \cdot \lambda_{\max}$). A further critical operation is the first-time initialization of the dual variables, where we choose the initial λ_i as "$\frac{\text{initial duality gap}}{\| \text{ total constraint violation } \|^2} \cdot$ violation of constraint i".

From a purely theoretical point of view, it is clear that (SDP$_4$) provides the strongest relaxation. To control the computational effort we stop the bundle method after a preset number of function evaluations and thus provide valid upper bounds of the solutions of the semidefinite relaxations, leaving some room for further incremental improvement. In our preliminary experiments on larger instances, we noticed that it is important to first get 'nearly' feasible with respect to the 3-cycle equations defining (SDP$_1$). Once this is achieved we start adding the triangle inequalities and the Lovász-Schrijver cuts. Since their number is quite large, we analyzed experimentally their effect and noticed that the inclusion of only the most violated triangle inequalities resulted in the quickest improvement of the bound with only a limited number of function evaluations. We therefore concentrate on getting good approximations to (SDP$_2$).

In Table 10.3 we provide substantially improved upper bounds for some hard (LOP) instances for which the optimal solution is not yet known (for details see [153, Tables 10,12,14]). These (LOP) instances can be downloaded from the benchmark library LOLIB http://heur.uv.es/optsicom/LOLIB. The table identifies the instance by its name and size n. We then provide the best known (integer) solution in the column labeled "bks", the linear programming bound (LP$_{\text{LOP}}$) and the semidefinite bound (SDP$_2$) that is (approximately) determined using 250 function evaluations of the bundle method. Finally, we also give the relative gap between the best known feasible solution and the bounds in the columns "LP-gap" and "SDP-gap". The gap (in percent) is computed as $gap = 100\frac{bound-bks}{bks}$.

In average we dualize about 17000 3-cycle equations and 200000 triangle inequalities in every iteration of the bundle method. The instances with 50 objects belong to the RandB problems, whereas the Paley graphs are contained in the Spec problems.

In Table 10.4 we summarize upper bounds for large-scale problems for which again the optimal solution is not yet known. Here we used (SDP$_1$) and allowed 25 function evaluations.

These instances with 100 objects belong to the RandA1, whereas the atp instance is contained in the

graph	n	bks	$(\mathtt{LP}_{\mathtt{LOP}})$	LP-gap	$\mathtt{SDP}_2^{250}$	SDP-gap
pal31	31	285	310	8.77	297	4.21
pal43	43	543	602	10.87	569	4.79
p50-05	50	42907	44196	3.00	43177	0.63
p50-06	50	42325	43765	3.40	42673	0.82
p50-07	50	42640	43977	3.14	42897	0.60
p50-08	50	42666	44655	4.66	43241	1.35
p50-09	50	43711	45183	3.37	43954	0.56
p50-10	50	43575	45346	4.06	44097	1.20
p50-11	50	43527	45132	3.69	43932	0.93
p50-12	50	42808	44671	4.35	43341	1.25
p50-13	50	43169	44872	3.94	43608	1.02
p50-14	50	44519	46272	3.94	44907	0.87
p50-15	50	44866	46479	3.60	45253	0.86
p50-16	50	45310	46693	3.05	45531	0.49
p50-17	50	46011	47751	3.78	46487	1.03
p50-18	50	46897	48152	2.68	47125	0.49
p50-19	50	47212	49162	4.13	47710	1.05
p50-20	50	46779	48155	2.94	47135	0.76
pal55	55	1045	1084	3.73	1049	0.38

Table 10.3: Bounds for some hard medium size (LOP) instances. n gives the number of nodes, "bks" denotes the best known (integer) solution. $(\mathtt{LP}_{\mathtt{LOP}})$ gives the linear programming bound and $\mathtt{SDP}_2^{250}$ denotes the semidefinite bound determined using 250 function evaluations of the bundle method. The relative gap between the best known feasible solution and the bounds are denoted by "LP-gap" and "SDP-gap" respectively. Notice that approximate running times for the SDP approach can be determined by multiplying the number of function evaluations by the effort per function evaluation for n, given in Table 10.2.

graph	n	bks	$(\mathtt{LP}_{\mathtt{LOP}})$	LP-gap	$\mathtt{SDP}_1^{25}$	SDP-gap
t1d100.01	100	106852	114468	7.13	110314	3.24
t1d100.02	100	105947	114077	7.67	110321	4.13
t1d100.03	100	109819	117843	7.31	113926	3.74
atp111	111	1495	1636	9.43	1526	2.07

Table 10.4: Bounds for large (LOP) instances (RandA1 problems). n gives the number of nodes, "bks" denotes the best known (integer) solution. $(\mathtt{LP}_{\mathtt{LOP}})$ gives the linear programming bound and $\mathtt{SDP}_2^{250}$ denotes the semidefinite bound determined using 25 function evaluations of the bundle method. The relative gap between the best known feasible solution and the bounds are denoted by "LP-gap" and "SDP-gap" respectively. Notice that approximate running times for the SDP approach can be determined by multiplying the number of function evaluations by the effort per function evaluation for n, given in Table 10.2.

Spec problems. Notice that for the other 22 RandA1 instances we obtained comparable computational results.

For all instances in Tables 10.3 and 10.4 we are able to substantially improve the best known upper bounds (see [153, Tables 10,12,14])). Thus we could also reduce the gaps between lower and upper bound. This substantial reduction of the gaps will also lead to considerably smaller branching trees in a Branch-and-Bound approach. To illustrate this let us mention that applying a Branch-and-Bound algorithm using the linear programming bounds to the paley graphs 31 and 43 results in bounds still beyond 300 respectively 600 after days of branching. Of course there are also many other problem classes with up to 250 objects where *LP-gap* is already quite small (< 1 %) and thus the current state-of-the-art Branch-and-Cut approach (for details see Section 4.1) yields the optimal solution.

In order to extend the SDP approach to instances with $n > 100$, we propose to apply first-order methods instead of (IPMs) to compute the function evaluations over the elliptope (for details see Chapter 3). We assume that this would result in a loss of accuracy (which is not much of a problem as we deduce approximate solutions anyway) but an eminent gain in running time.

10.3 Speeding up the Linear Relaxation

In this section we will explain two ideas to speed up the solution of (LP_{LOP}) and showcase their effect on selected instances from the benchmark library LOLIB. The linear programming relaxation of (LOP) can be solved directly by state-of-the-art software like CPLEX [117] for $n \leq 100$. The standard approach for larger instances is to alternately solve (LP_{LOP}) on a subset S of the 3-cycle inequalities and add the most violated 3-cycle inequalities to S (hence the set S grows in every CPLEX iteration). These two steps are iterated until all 3-cycle inequalities are satisfied.

First we propose to use a dynamic version of the bundle method (for details see Section 3.3) to speed up this procedure. We apply 50 function evaluations to obtain good approximations of the objective value of (LP_{LOP}) and thus also of the set of constraints that are active at the optimal solution. In the case of linear programming function evaluations of the bundle method consist in maximizing a linear function over a set of bound constraints and hence are trivial. Therefore the application of the bundle method is quite cheap, its computational effort ranges between 30 seconds for instances with 100 objects and 10 minutes for instances with 450 objects. Then we use the set of constraints obtained by the bundle method to initialize S and start with the standard approach explained above. By doing so we substantially reduce the number of CPLEX calls *# cc* as well as the number of 3-cycle constraints $|S_{\text{fin}}|$ used by CPLEX for its final call where it finds the optimal solution of (LP_{LOP}).

Secondly we propose a post-processing of the CPLEX solution in every iteration that yields a further speed up of our method by at least the factor 5 for all considered instances. It allows us to solve (LP_{LOP}) for the large sparse (in the cost matrix D) *atp*-instances for the first time. Linear programming software like CPLEX puts variables associated to cost coefficients equal to zero to their upper or lower bound (e.g. -1 or $+1$ in the $\{-1, +1\}$ formulation of (LP_{LOP}), see (4.6)). We propose to set these variables as close as possible to $\frac{\text{upper bound + lower bound}}{2}$, taking into account the current subset of 3-cycle inequalities S. This post-processing of the CPLEX solution results in fewer violated 3-cycle inequalities (a smaller set S_{fin}) and faster running times and its effects grow with the sparsity of the instance.

In Table 10.5 we summarize upper bounds for some instances for which the optimal solution is not yet known (for details see [153, Tables 10,11,14,15]). These instances can be downloaded from the benchmark library LOLIB `http://heur.uv.es/optsicom/LOLIB`. The table identifies the instance by its name and size n. The linear programming bound resulting from the linear programming relaxation (LP_{LOP}) is given in column "z_{LP}". Then we provide the number of CPLEX calls "# cc" and running time in seconds. Finally we give the number of 3-cycle constraints $|S_{\text{fin}}|$ needed to find the optimal solution of (LP_{LOP}). The experiments were conducted on an Intel Xeon 5160 processor with 3 GHz and 2 GB RAM.

For the first time we could compute the linear programming relaxation (LP_{LOP}) of the very sparse

graph	n	z_{LP}	# cc	time	$\lvert S_{\text{fin}}\rvert$
t1d100.01	100	114468	2	73.2	18004
atp111	111	1514.913	5	63.8	12512
atp134	134	1824.284	6	158.8	19721
t1d150.01	150	261413	2	773	68011
t2d150.01	150	76276.924	3	59.1	10441
be75eec.150	150	3527035.851	5	1149.6	38856
stabu1.150	150	2923697.661	5	914.4	35161
atp163	163	2110.254	9	602.8	33546
t1d200.01	200	464286.000	2	7578.6	240783
t1d200.02	200	460090.667	2	4568.2	162447
t2d200.01	200	148294.764	4	472.6	33616
be75eec.250	250	9150239.673	5	23568.9	121873
stabu1.250	250	8011535.186	5	22186.3	110612
atp452	452	2756.527	15	32955.2	213860

Table 10.5: Computational details on speeding up (LP_{LOP}) by using the bundle method and post-processing of the CPLEX solution. n gives the number of nodes, "z_{LP}" denotes the linear programming bound, "# cc" gives the number of CPLEX calls and $|S_{\text{fin}}|$ denotes the number of 3-cycle constraints needed to find the optimal solution of (LP_{LOP}). The running times are given in seconds.

instances "atp134", "atp163" and "atp452" as well as for the instance "stabu1.250".We also tried to incorporate these bounds as well as the somewhat weaker but also cheaper bounds obtained by using only the bundle method in a Branch-and-Bound framework. But we could not achieve any significant progress because the linear programming bounds of these instances are quite weak and thus the Branch-and-Bound trees are very large. Hence there is not much gain in solving the linear programming relaxation say 100 times faster. Therefore we conclude that for getting essentially better upper bounds for these instances it is more promising to concentrate on methods that yield (approximate) solutions of stronger relaxations, like the ones based on SDP discussed in the previous sections.

Chapter 11

The Minimum Linear Arrangement Problem

This chapter is based on Section 7 of the paper "Semidefinite Relaxations of Ordering Problems" [115]. We compare the most competitive exact approaches for the minimum Linear Arrangement Problem (`minLA`) (for descriptions of these algorithms see Chapter 5) on well-known benchmark instances from the literature. But prior to this we apply the SDP approach to the Cartesian cube and compare the obtained values to other lower bounds based on combinatorial or spectral properties. We again apply a dynamic version of the bundle method to obtain approximate solutions of our SDP relaxations (for details see Sections 3.3 and 10.2). Notice that contrary to Section 10.2, now we work with the strongest SDP relaxation ($\texttt{SDP}_4$) (with C and K defined in Section 5.3 and c equal to the zero vector) as the number of violated Lovász-Schrijver cuts (4.13) stays quite small for all instances considered and thus these constraints do not restrict the overall performance of the algorithm (for detailed computational results on this effect and the gains of working with ($\texttt{SDP}_4$) instead of ($\texttt{SDP}_2$) see Section 14.7).

11.1 The Cartesian Cube

To get a first idea of the tightness of the SDP approach, we solve (`minLA`) on the n-dimensional Cartesian cube Q_n. The optimal value

$$z_*(Q_n) = 2^{n-1}(2^n - 1)$$

was determined by Harper [92]. There exist several combinatorial lower bounds for (`minLA`), e.g. the degree lower bound introduced by Petit [172]

$$z_* \geq \frac{1}{2} \sum_{i \in V} \left\lfloor \frac{(\deg(i)+1)^2}{4} \right\rfloor,$$

where $\deg(i)$ gives the degree of vertex i. We refer the reader to [173] for further such bounds like e.g. the edge or the mesh bound. While combinatorial bounds are tight for special graph classes, they are rather weak in general. For further comparison we also consider two more sophisticated lower bounds based on spectral properties. Juvan and Mohar [121] show that

$$z_* \geq \lambda_2 \frac{|V|^2 + 1}{6},$$

where λ_2 denotes the second smallest eigenvalue of the Laplacian of the graph. Helmberg et al. [102] provide a theoretically stronger bound that is based on a projection technique developed for the quadratic assignment problem and uses the full spectrum of the Laplacian.

The lower bounds obtained by the approaches mentioned above are summarized in Table 11.1. The relaxation (SDP$_4$) correctly identifies $z_*(Q_n)$ for $n \leq 4$ and also provides very strong bounds for larger dimensions.

n	Petit [172]	Juvan and Mohar [121]	Helmberg et al. [102]	(SDP$_4$)	Optimum $z_*(Q_n)$
2	3	5	6	6	6
3	6	21	24	28	28
4	13	85	99	120	120
5	23	341	392	493	496
6	37	1365	1542	2002	2016

Table 11.1: Lower bounds for (minLA) on the hypercube Q_n

11.2 Medium and Large Instances

In this section we computationally compare the exact algorithms theoretically discussed in Chapter 5 on well-known benchmark instances from several sources. We summarize the results of our experiments in Table 11.2. The instances in the first block were first addressed in Caprara and Salazar-González [36] in conjunction with the Minimum Bandwidth Problem and the second block of instances was proposed by Seitz [191]. "gd95c" and "gd96c" were introduced by Díaz et al. [65] and can be downloaded from http://www.lsi.upc.edu/~jpetit/MinLA/Experiments/. All instances from the first and second block are available at the Boeing Sparse Matrix Collection [66]. Table 11.2 starts with the instance name, the number of vertices $|V|$, the density $d = \frac{2|E|}{|V|(|V|-1)}$ of the instance and the upper bound "ub" obtained by a multi-start local search routine. For all approaches, we give their computed lower bounds "lb" and the associated running times in seconds, where we set the time limit to 24 hours. A missing entry indicates that the instance was not considered by the respective approach. Further note that the Branch-and-Cut-and-Price algorithm from [191] does not yield valid lower bounds if stopped because of the time limit.

The computations in [35] were carried out on a PC with processor Intel Core 2 Duo 3.33GHz and 2 GB RAM, Schwarz and Seitz [190, 191] run their algorithms on a 2× Xeon CPU with 2.5GHz and 2GB RAM, whereas for computing the SDP relaxation we use an Intel Xeon 5160 processor with 3 GHz and 2 GB RAM. Notice that we do not take into account the speed of the machines, as it does not differ too much. The machine of Schwarz and Seitz is the quickest and about 2.5 times faster than ours, which is the slowest.[1]

The semidefinite approach proves optimality of the upper bounds for 13 instances for the first time together with [34, 190]. Our SDP algorithm also provides the best known lower bound for two instances ("can_73" and "impcol_b"). As we get our lower bounds from the root node relaxation, we are very optimistic to solve these instances when using our bounds in a Branch-and-Bound approach. The algorithm of Caprara et al. [34], realized by Schwarz [190], is preferable to the SDP approach for small graphs and large, sparse graphs. As the Single Row Facility Layout Problem can be interpreted as (minLA) with edge weights on the complete graph, the computational results of the next chapter support our conjecture that the SDP approach is the most competitive one for graphs with $|V| \geq 35$ and $d \geq 50\%$.

While our approach is restricted to graphs with $|V| \leq 100$ independent of the graph's density, the approach from [34, 190] can be successfully applied to sparse graphs with up to ≈ 200 vertices. For even larger graphs the algorithm proposed in [35] is the method of choice as it can provide reasonable bounds for sparse graphs with up to ≈ 1000 vertices.

[1] For exact numbers of the speed differences see http://www.cpubenchmark.net/.

Instance				Reference [34, 190]		Reference [35]		Reference [191]		SDP model	
name	$\|V\|$	d	ub	lb	time	lb	time	lb	time	lb	time
can_24	24	0.246	210	210	4.7	203	2.8	203.86	1080	210	66.9
fidap005	27	0.358	414	414	4.1	412	4.2			414	124.4
fidapm05	42	0.277	1003	1003	1516.9	998	805.2			1003	6200.1
bcspwr01	39	0.062	106	106	6.5	91	0.7	-	limit	106	1036.2
bcsstk01	48	0.156	1132	1132	40852.8	972	3848.1	-	limit	1130	10744.5
bcspwr02	49	0.050	161	161	16.0	144	1.8	-	limit	161	5237.7
dwt_59	59	0.060	289	289	39.4	258	55.4			289	37925.3
can_61	61	0.135	1137	1137	1371.9	1119	538	-	limit	843	11790.9
can_62	62	0.041	210	210	49.6	203	2.8	-	limit	210	30090.2
dwt_66	66	0.059	192	192	34.2	192	1.7			192	3660.0
dwt_72	72	0.029	167	167	38.4	150	6.7			167	77333.7
can_73	73	0.057	1100	962	limit	971	2016.8			1088	limit
steam3	80	0.134	1416	1416	164.3	1406	limit			1413	limit
dwt_87	87	0.060	932	932	2507.0	897	limit			917	limit
nos4	100	0.049	1031	1031	3990.0	976	59118.0			1014	limit
tub100	100	0.029	246	246	131.0	245	71.5			243	limit
pores_1	30	0.236	383	383	29.9			351.02	15340	383	286.5
ibm32	32	0.181	485	485	1241.3			462.36	30677	485	306.1
curtis54	54	0.086	454	454	69.6			-	limit	357	13851.5
will57	57	0.079	335	335	56.0			-	limit	335	18407.0
impcol_b	59	0.164	2076	2000	limit			-	limit	2074	51082.6
gd95c	62	0.076	506	506	109.7	443	68.3	-	limit	506	36647.7
gd96c	64	0.060	519	519	2368.3	402	218.1	-	limit	516	91582.1

Table 11.2: Comparison of several exact approaches for `(minLA)`. $|V|$ gives the number of vertices, d denotes the density of the instance, "ub" gives a feasible upper bound, "lb" denotes the lower bound and the time limit is set to 24h. A missing entry indicates that the instance was not considered by the respective approach. Further note that the Branch-and-Cut-and-Price algorithm from [191] does not yield valid lower bounds if stopped because of the time limit.

Chapter 12

The Single Row Facility Layout Problem

In this chapter we again use a dynamic version of the bundle method to obtain approximate solutions of the relaxation (SDP$_4$) (with C and K defined in Section 6.3 and c equal to the zero vector - for further details on our algorithmic approach see Sections 3.3 and 10.2) for a broad selection of small, medium and large instances of the Single Row Facility Layout Problem (SRFLP). The chapter is based on Section 3 of the paper "A Computational Study for the Single-Row Facility Layout Problem" [116]. We compare our approach to the leading algorithms for the different instance sizes. Thereby we demonstrate that it clearly dominates all other methods, permitting significant progress for medium as well as large instances. We can give optimal solutions for several medium instances from the literature with up to 42 facilities that remained unsolved so far and reduce all the best known gaps for large scale instances by a factor varying from 2 to 100. Additionally in Section 12.2 we propose a new SDP based rounding heuristic for (SRFLP) and relate it to the SDP heuristic from [10] concerning its computational costs and practical performance.

12.1 Comparison of Globally Optimal Methods for Small and Medium Instances

In Table 12.1 we computationally compare the four most competitive approaches to (SRFLP) for small and medium instances. These are the ILP approaches of Amaral and Letchford [8] and Amaral [7], the SDP approach of Anjos and Vanelli [13] building on relaxation (A$_2$) and our SDP approach building on relaxation (SDP$_4$).

Anjos and Vanelli start with the basic relaxation (A$_1$) and then enhance it with violated triangle inequalities (4.12) in every iteration (using the interior-point solver CSDP version 5.0 [24, 146]) until no more triangle inequalities are violated (for details on the underlying model see Section 6.3).

Amaral and Letchford suggest an ILP Branch-and-Cut algorithm based on the distance variables z_{ij} (for details on the underlying model see Section 6.2). They use a cheap initial LP relaxation with only $O(n^2)$ non-zero coefficients and apply exact separation routines for triangle and special strengthened pure negative type inequalities and heuristic ones for clique, rounded psd and star inequalities. They suggest a specialised branching rule to avoid the use of additional binary variables and use a primal heuristic based on multi-dimensional scaling to obtain feasible layouts.

Amaral proposes an ILP cutting plane algorithm based on the betweenness variables ξ_{ijk} (for details on the underlying model see again Section 6.2) that improves on the results in [13] and [8]. For computational usage of the betweenness model Amaral suggests to alternate between solving (LP) and strengthening (LP) (by searching for cutting planes $(6.8)_{\beta=6}$ violated at the optimal solution of the current (LP) and adding

them to (LP)). Amaral also introduces new instances with 33 and 35 facilities, solves them to optimality and points out that he cannot solve larger instances with his approach as the involved linear programs become too large and too difficult to solve with the currently available LP solvers.

In Table 12.1 we give a full computational comparison of the most successful exact approaches to (SRFLP) on all available instances from the literature, including well-known benchmark instances [5, 6, 7, 108, 195], instances with clearance requirement [107] and random-generated instances [13].[1] The table identifies the instance by its name, source and number of departments n and gives the times required by the four approaches to find a layout and prove its optimality.

The computations in [13] were carried out on a 2.0GHz Dual Opteron with 16 GB RAM, Amaral used an Intel Core Duo, 1.73 GHz PC with 1 GB RAM, in [8] a 2.5 GHz Pentium Dual Core PC with 2 GB RAM was employed, whereas for applying our approach we again use an Intel Xeon 5160 processor with 3 GHz and 2 GB RAM.

For small instances with up to 20 facilities the ILPs are preferable to the SDP approaches whereas our SDP approach outperforms the other approaches on the larger instances. The difference between the approaches strongly grows with the problem size. Note that we do not take into account the speed of the machines, as it does not differ too much and thus does not affect the conclusions drawn above. Our machine is the quickest and about 2.5 times faster than the one in [7], which is the slowest.[2]

This motivates us to tackle new, larger instances with our approach. We summarize the results for the five instances with 40 facilities, a density of 50 % and random lengths and connectivities between 1 and 10 in Table 12.2.[3]

We succeed in providing optimal solutions within reasonable time for all these instances that can hardly be solved to optimality with any of the other three approaches.

12.2 Heuristics Based on Semidefinite Optimization

For large (SRFLP) instances, not only obtaining tight lower bounds is difficult but also finding very good feasible layouts is a challenging task for larger instances. Thus we propose a new SDP based rounding heuristic and relate to the SDP heuristic of Anjos et al. [10] concerning its computational costs and practical performance. Note that the heuristic of Anjos et al. provided the best known layouts so far for the large instances in the following section.

The SDP relaxations ($\mathtt{A}_0$)– ($\mathtt{A}_2$) and ($\mathtt{SDP}_1$)– ($\mathtt{SDP}_4$) are closely related to the basic SDP relaxation ($\mathtt{MC}_1$) for the Max-Cut Problem used in the seminal paper of Goemans and Williamson [86] to obtain high quality feasible solutions providing upper bounds. However the hyperplane rounding idea suggested in [86] cannot be applied directly to (SRFLP) to get a good layout because it yields a $\{-1, 1\}$ vector $\tilde{y}$, which need not be feasible with respect to the three cycle inequalities (4.6b). That is why Anjos et al. [10] propose a different procedure to obtain a good feasible layout from the optimal solution of the SDP relaxation whereas we suggest to apply a repair strategy to the infeasible $\tilde{y}$.

Anjos et al. propose to use the entries $y^*_{ij,kl}$ of the optimal matrix Y^* of the SDP relaxation in the following way to obtain a good feasible layout: Fix a row ij and compute the values

$$\omega_k^{ij} = \frac{1}{2}\left(n + 1 + \sum_{l \in \mathcal{N}, k \neq l} y^*_{ij,kl}\right), \quad k \in \mathcal{N}.$$

These values are motivated by the fact that if Y^* is rank-one, then the values $\omega_k^{ij}, k \in \mathcal{N}$ are all distinct and belong to $\mathcal{N}$ and thus give a permutation of $\mathcal{N}$. In general, $\operatorname{rank}(Y^*) > 1$ and thus a permutation can be obtained by sorting $w_k^{ij}, k \in \mathcal{N}$ in either decreasing or increasing order (since the objective value is the

[1] Most of the instances can be downloaded from http://flplib.uwaterloo.ca/.

[2] For exact numbers of the speed differences see http://www.cpubenchmark.net/.

[3] These instances and the corresponding optimal orderings are available from http://flplib.uwaterloo.ca/.

Instance	Source	n	Anjos and Vanelli [13] using (A$_2$)	Amaral/Letchford [8]	Amaral [7]	(SDP$_4$) with bundle method
S5	[195]	5		0.1	0.1	0.1
S8	[195]	8		0.5	0.1	0.6
S8H	[195]	8	0.2	0.1	0.1	2.3
S9	[195]	9		0.1	0.1	0.7
S9H	[195]	9		2.4	0.1	9.2
S10	[195]	10	3.4	0.4	0.2	0.6
S11	[195]	11	32.6	0.7	0.3	1.3
P15	[5]	15			2.8	19.7
P17	[6]	17			8.4	34.9
P18	[6]	18			13.3	32.5
H_20	[108]	20	26:54	2:22	30.8	54.3
H_30	[108]	30	15:50:57	28:07:49	27:35	9:07
Cl_5	[108]	5	0.1	0.1	0.2	0.1
Cl_6	[108]	6	0.4	0.1	0.1	0.1
Cl_7	[108]	7	1.2	0.3	0.1	0.6
Cl_8	[108]	8	1.8	0.1	0.1	0.4
Cl_12	[108]	12	32.8	4.0	0.6	7.9
Cl_15	[108]	15	5:53	9.6	3.2	19.6
Cl_20	[108]	20	41:32	5:12	40.1	1:16
Cl_30	[108]	30	51:06:53	17:49:43	1:12:19	14:17
N25_01	[13]	25	3:44:38	7:19:44	3:46	2:48
N25_02	[13]	25	4:50:27	38:35	9:59	5:46
N25_03	[13]	25	5:48:21	1:25:41	4:49	4:11
N25_04	[13]	25	4:04:51	39:34	10:19	5:33
N25_05	[13]	25	8:22:22	1:18:10	3:47	3:31
N30_01	[13]	30	7:41:06	34:00:51	25:41	4:42
N30_02	[13]	30	10:41:53	3:56:53	22:43	6:08
N30_03	[13]	30	19:32:01	13:08:12	23:14	10:12
N30_04	[13]	30	31:03:11	58:20	2:19:22	11:44
N30_05	[13]	30	19:54:07	13:03:51	1:05:36	18:30
Am33_01	[7]	33			1:15:57	19:28
Am33_02	[7]	33			2:35:22	48:07
Am33_03	[7]	33			2:22:32	36:33
Am35_01	[7]	35			1:35:04	17:30
Am35_02	[7]	35			5:27:34	41:01
Am35_03	[7]	35			2:17:52	53:14

Table 12.1: Results for (SRFLP) instances with up to 35 facilities. The running times are given in sec, in min:sec or in h:min:sec respectively.

Instance	n	Optimal cost	(SDP$_4$) with bundle method
N40_1	40	107348.5	1:01:36
N40_2	40	97693	52:52
N40_3	40	78589.5	1:21:40
N40_4	40	76669	1:15:58
N40_5	40	103009	2:20:09

Table 12.2: Results for 5 new (SRFLP) instances with 40 facilities. The running times are given in min:sec or in h:min:sec.

same). The output of the SDP-based heuristic is the best layout found by considering every row ij of Y^* with $i, j \in \mathcal{N}, i < j$.

We suggest to take the $\{-1, 1\}$ vector $\tilde{y}$ obtained from hyperplane rounding and make it feasible with respect to the 3-cycle inequalities by flipping the signs of some of its entries appropriately. Computational experiments demonstrated that the repair strategy is not as critical as one might assume. For example we know from Multi-level Crossing Minimization that the heuristic clearly dominates traditional heuristic approaches (for details see Chapter 14).

The heuristic of Anjos et al. is much cheaper than ours as we have to factorize Y^* to carry out the rounding procedure. Nonetheless the computation times of both heuristics are negligible compared to the computational effort for the lower bound computation. We compared both heuristics concerning the quality of the layouts produced on many test instances and found out that our heuristic is clearly superior. This is also supported by a comparison of the upper bounds achieved by both approaches in Tables 12.3 and 12.4, where our heuristic improves on the ones of Anjos et al. on all instances considered.

Finally let us give a more detailed description of the implementation of our heuristic. We consider a vector y', that encodes a feasible, random ordering on all levels. The algorithm stops after 1000 executions[4] of step 2.

1. Let Y'' be the current primal fractional solution of (SDP_4) obtained by the bundle method. Compute the convex combination $R := \lambda(y'y'^\top) + (1-\lambda)Y''$, using some random $\lambda \in [0.3, 0.7]$. Compute the Cholesky decomposition $DD^\top$ of R.
2. Apply Goemans-Williamson hyperplane rounding to D and obtain a $-1/+1$ vector $\overline{y}$ (cf. [184]).
3. Compute the induced objective value $z(\overline{y})$. If $z(\overline{y}) \geq z(y')$: goto step 2.
4. If $\overline{y}$ satisfies all 3-cycle inequalities: set $y' := \overline{y}$ and goto 2. Else: modify $\overline{y}$ by changing the signs of one of three variables in all violated inequalities and goto step 3.

y' is then the heuristic solution. If the duality gap is not closed after the heuristic, we continue with further bundle iterations and then retry the heuristic (retaining the last vector y').

12.3 Comparison of Globally Optimal Methods for Large Instances

In this section we compare the most competitive approaches to (SRFLP) for obtaining tight bounds of large instances. These are the algorithms of Anjos and Yen [14] building on relaxations (A_0) and (A_1) respectively and again our approach building on relaxation (SDP_4) and using our new SDP heuristic described in the previous section. For solving relaxations (A_0) and (A_1), Anjos and Yen use the interior-point solver CSDP (version 5.0) [24, 146]. In Tables 12.3 and 12.4 we compare the three SDP approaches on instances with 36 – 100 facilities taken from [10] and [14].[5]

As already mentioned in Section 10.2 the function evaluations over the elliptope constitute the computational bottleneck of the bundle method and are responsible for more than 95% of the overall running time for large instances. To control the computational effort of our approach we restrict the number of function evaluations to 500 for instances with up to 64 departments and to 250 for larger instances. This limitation of the number of function evaluations leaves some room for further incremental improvement.

When comparing the running times of the three approaches we do not take into account that Anjos and Yen use a machine (2.4GHz Quad Opteron with 16 Gb of RAM) that is more than 1.5 times faster and has 8 times the memory of our machine.[6]

[4] Before its 501st execution, we perform step 1 again. As step 1 is quite expensive, we refrain from executing it too often.

[5] Most of the instances can be downloaded from `http://flplib.uwaterloo.ca/`. Our improved gaps and the corresponding orderings are also available there.

[6] For details see `http://www.cpubenchmark.net/`.

In Table 12.3 we compare the three approaches for problems with 36 to 56 facilities for which no optimal solution was known before. The table identifies the instance by its name and number of departments n. We then provide the lower bound "lb" and the objective value of the best layout found "blf" as well as the associated running times for the different approaches. Finally we give the running times that our algorithm based on relaxation $(\mathtt{SDP}_4)$ needs to improve on the gaps of the other two approaches "improve gap $(\mathtt{A}_0)$" and "improve gap $(\mathtt{A}_1)$".

The results show that the SDP approaches of Anjos and Yen allow for substantial improvement. On the one hand we reduce the difference between objective value of the best layout and the lower bound for all instances by factors that are, except once, > 10 (both lower and upper bounds are improved for all instances). On the other hand we reach the gaps achieved by the other two approaches considerably faster. Further it is worthwhile to note that all instances with 36 facilities and even one instance with 42 facilities can be solved to optimality for the first time.

In Table 12.4 we compare the cheaper approach from [14] using relaxation $(\mathtt{A}_0)$ (the other one gets too expensive for these instances) to our method (again using our new SDP heuristic) for problems with 60 to 100 facilities.

The results show that the SDP approach of Anjos and Yen again allows for some improvement. On the one hand we reduce the difference between the objective value of the best layout and the lower bound for all instances by factors going from clearly above 10 to 2 as the instance sizes grow (again both lower and upper bounds are improved for all instances). On the other hand the gaps achieved by the approach of Anjos and Yen are reached in average in about half the time by our algorithm. Contrary to the results of the previous chapter on the minimum Linear Arrangement Problem (which is a special case of (SRFLP)), for the Single Row Facility Layout Problem linear programming based approaches are restricted to instances with ≤ 35 departments, as (SRFLP) instances lead to linear programs with a dense cost structure and hence sparsity cannot be exploited. Therefore our SDP approach is the method of choice for instances of challenging size $n \geq 30$, although the corresponding SDP cost matrices are quite sparse, as only $2\binom{n}{3}$ of the $\left(\binom{n}{2}+1\right)^2$ matrix entries are $\neq 0$.

Let us finally compare the SDP-based heuristic from [115] with the recent tabu search based heuristic of Samarghandi and Eshghi [187] and the recent permutation-based genetic algorithm of Datta et al. [49] on the 20 "AKV"-instances [10]. On five instances all three heuristics yield the same upper bound, on 5 instances the heuristics from [187] and [49] yield the same best value, on 5 instances the algorithm of Datta et al. [49] generates the best feasible layouts and on 5 instances the approach from [115] produces the best upper bounds. In general the SDP-based heuristic seems to be preferable when $n \leq 70$ and computation time is not a critical factor as its performance depends on the quality of the lower bounds from the SDP relaxation. The "sko"-instances [14] were not considered in [187] and [49], hence for these instances the lower and upper bounds presented in Tables 12.3 and 12.4 are the best known ones to date.

Instance	n	SDP Anjos/Yen using (A$_0$) [14]			SDP Anjos/Yen using (A$_1$) [14]			SDP Hungerländer/Rendl [115] - restricted to 500 function evaluations					
		lb	blf	time	lb	blf	time	lb	blf	time	gap in %	improve gap (A$_0$)	improve gap (A$_1$)
ste36-1	36	9851	10328	7:15	10087.5	10301	14:57	10287	10287	14:50	0 %	2:23	2:55
ste36-2	36	170759.5	182649	7:12	175387	181910	14:17	181508	181508	25:25	0 %	2:05	2:39
ste36-3	36	96090	104041.5	7:13	98739	102179.5	13:42	101643.5	101643.5	24:01	0 %	2:35	3:09
ste36-4	36	91103	96854.5	7:16	94650.5	96080.5	14:23	95805.5	95805.5	16:15	0 %	2:01	3:49
ste36-5	36	87688	92563.5	7:19	89533	91893.5	14:25	91651.5	91651.5	17:58	0 %	2:00	3:09
sko42-1	42	24517	25779	20:07	24807	25724	45:21	25521	25525	2:23:09	0.02 %	5:20	7:34
sko42-2	42	207357	218117.5	20:21	210785	217296.5	45:14	216099.5	216120.5	2:43:34	0.01 %	5:57	9:38
sko42-3	42	167783.5	174694.5	20:10	169944.5	173854.5	47:32	173245.5	173267.5	2:47:18	0.01 %	8:01	17:51
sko42-4	42	131536	139630	19:21	133429.5	138829	48:18	137379	137615	2:53:05	0.17 %	4:55	7:24
sko42-5	42	238669.5	250501.5	20:18	242925.5	249327.5	45:41	248238.5	248238.5	1:08:42	0 %	6:37	11:13
sko49-1	49	39333.5	41379	59:55	39794.5	41308	2:48:57	40895	41012	4:36:21	0.29 %	33:33	57:43
sko49-2	49	403024.5	418370	1:03:30	407741.5	418288	2:50:32	416142	416178	8:27:34	0.01 %	19:11	31:43
sko49-3	49	313923.5	326004	1:02:13	317628.0	325747	2:51:45	324464	324512	8:03:03	0.02 %	19:15	26:20
sko49-4	49	229809.5	238380.5	1:05:34	232368	237894.5	2:50:40	236718.5	236755	9:15:14	0.02 %	21:59	32:01
sko49-5	49	645406.5	673303	1:04:13	652638	671508	2:51:45	666130	666143	9:30:22	0.002 %	35:04	35:04
sko56-1	56	61789.5	64454	3:05:19	62496.5	64396	8:40:40	63971	64027	12:36:33	0.09 %	41:55	1:03:12
sko56-2	56	480473.5	499700	3:09:35	486426.5	498836	9:07:10	496482	496561	15:59:27	0.02 %	41:05	1:11:58
sko56-3	56	164609.5	171963	3:08:16	166441.5	171860	8:57:50	169644	171032	16:22:56	0.82 %	1:00:39	1:53:27
sko56-4	56	302572.5	325803	2:55:51	306550.5	315175	9:00:52	312656	313497	15:17:25	0.27 %	52:24	1:26:44
sko56-5	56	575501.5	595593.5	2:56:20	582117.5	594477.5	8:57:53	591915.5	592335.5	17:46:46	0.07 %	1:08:30	1:34:20

Table 12.3: Results for well-known (SRFLP) instances with 36–56 facilities. n gives the number of facilities, "lb" denotes the lower bound, "blf" gives the objective value of the best layout found and "improve gap (A$_0$)" and "improve gap (A$_1$)" denote the running times that our algorithm based on relaxation (SDP$_4$) needs to improve on the gaps of the other two approaches. The running times are given in min:sec or in h:min:sec respectively.

Instance	n	SDP Anjos/Yen using (A$_0$) [14]				(SDP$_4$) with bundle method - restricted to 250 function evaluations				
		lb	blf	time	gap in %	lb	blf	time	gap in %	improve gap (A$_0$)
AKV-60-01	60	1473338.5	1478464.0	5:39:13	0.35 %	1477134	1477834	12:38:16	0.05 %	4:42:33
AKV-60-02	60	829956.5	844695.0	5:08:10	1.78 %	841472	841776	11:08:16	0.04 %	2:01:14
AKV-60-03	60	641723	650533.5	4:50:48	1.38 %	647031.5	648337.5	9:51:06	0.20 %	2:58:00
AKV-60-04	60	389733	400669.0	4:55:19	2.81 %	397951	398406	10:49:59	0.11 %	1:57:18
AKV-60-05	60	316284.5	319103.0	5:05:28	0.89 %	318792	318805	12:39:37	0.004 %	2:54:16
sko64-1	64	93388	97842	8:16:08	4.77 %	96569	97194	13:08:05	0.65 %	2:15:21
sko64-2	64	619258	636602.5	8:36:06	2.80 %	633420.5	634332.5	14:28:38	0.14 %	2:44:31
sko64-3	64	402165.5	418083.5	8:47:21	3.96 %	412820.5	414384.5	14:04:55	0.38 %	4:54:09
sko64-4	64	285762.5	300469	8:38:01	5.15 %	295145	298155	13:55:45	1.02 %	2:48:40
sko64-5	64	488035	505185.5	8:47:49	3.51 %	501059.5	502063.5	13:53:04	0.20 %	2:30:01
AKV-70-01	70	1513741.5	1533075	24:25:30	1.28 %	1526359	1528560	26:41:34	0.14 %	10:36:44
AKV-70-02	70	1424673.5	1444720	24:20:39	1.41 %	1439122	1441028	26:11:27	0.13 %	7:56:27
AKV-70-03	70	1503311.5	1526830.5	23:11:47	1.56 %	1517803.5	1518993.5	26:15:14	0.08 %	6:59:29
AKV-70-04	70	951725	972389	22:56:51	2.17 %	967316	969150	27:28:48	0.19 %	6:22:30
AKV-70-05	70	4207969.5	4218730.5	23:42:47	0.26 %	4213774.5	4218002.5	28:16:05	0.10 %	9:38:10
sko72-1	72	135280.5	140209	20:26:35	3.64 %	138885	139231	29:33:19	0.25 %	5:22:09
sko72-2	72	690377	716873	19:58:29	3.84 %	707643	715611	29:40:41	0.11 %	11:06:29
sko72-3	72	1026164	1063314.5	22:19:25	3.62 %	1048930.5	1061762.5	32:38:47	0.12 %	11:57:10
sko72-4	72	898586.5	924542.5	20:20:37	2.89 %	916229.5	924019.5	33:58:28	0.85 %	8:26:20
sko72-5	72	415320.5	432062.5	20:21:15	4.03 %	426224.5	430288.5	31:39:43	0.95 %	6:23:57
AKV-75-01	75	2377176	2394812.5	40:15:12	0.74 %	2387590.5	2393600.5	37:57:53	0.25 %	22:19:37
AKV-75-02	75	4294138	4322967	42:23:20	0.67 %	4309185	4322492	39:28:38	0.31 %	21:08:44
AKV-75-03	75	1230123.5	1255634	38:27:39	2.07 %	1243136	1249251	38:21:06	0.49 %	11:48:54
AKV-75-04	75	3911919	3950444.5	41:27:49	0.99 %	3936460.5	3941845.5	38:42:58	0.14 %	17:57:02
AKV-75-05	75	1763890.5	1797676	43:09:58	1.92 %	1786154	1791469	41:10:37	0.30 %	10:43:21
AKV-80-01	80	2045170.5	2073453.5	49:07:29	1.38 %	2063346.5	2070391.5	58:24:49	0.34 %	21:03:27
AKV-80-02	80	1903788	1923506	48:31:48	1.04 %	1918945	1921202	58:47:15	0.12 %	18:42:50
AKV-80-03	80	3237288.5	3256577	49:22:31	0.60 %	3245254	3251413	58:17:19	0.19 %	26:04:02
AKV-80-04	80	3730569	3747950	52:16:43	0.47 %	3739657	3747829	58:50:47	0.22 %	35:17:04
AKV-80-05	80	1555271.5	1594228	47:03:04	2.51 %	1585491	1590847	58:30:30	0.34 %	13:12:47
sko81-1	81	197416.5	207229	47:42:37	4.97 %	203424	207063	52:44:10	1.79 %	18:28:22
sko81-2	81	507726	527239.5	49:02:44	3.84 %	518711.5	526157.5	59:58:08	1.44 %	22:45:43
sko81-3	81	942850.5	979816	47:45:13	3.92 %	962886	979281	58:17:40	1.70 %	17:27:37
sko81-4	81	1971210.5	2042462	46:48:01	3.62 %	2019058	2035569	57:21:49	0.82 %	17:33:03
sko81-5	81	1267977	1311605	50:42:29	3.44 %	1293905	1311166	58:59:28	1.33 %	22:49:57
sko100-1	100	367048.5	380981	214:49:05	3.80 %	375999	380562	191:47:21	1.21 %	108:20:47
sko100-2	100	2024668	2089757.5	240:13:08	3.21 %	2056997.5	2084924.5	201:46:52	1.36 %	116:16:55
sko100-3	100	15750362	16251391.5	236:03:51	3.18 %	15987840.5	16216076.5	212:38:54	1.43 %	109:48:22
sko100-4	100	3148661	3266569	255:53:11	3.74 %	3200643	3263493	204:14:39	1.96 %	133:18:35
sko100-5	100	1002763.5	1040987.5	219:33:25	3.81 %	1021584.5	1040929.5	201:29:27	1.89 %	111:11:49

Table 12.4: Results for well-known (SRFLP) instances with 60–100 facilities. n gives the number of facilities, "lb" denotes the lower bound, "blf" gives the objective value of the best layout found and "improve gap (A$_0$)" denotes the running times that our algorithm based on relaxation (SDP$_4$) needs to improve on the gaps of the approach by Anjos and Yen. The running times are given in h:min:sec.

Chapter 13

The General Quadratic Ordering Problem

Next we present preliminary computational results for semidefinite relaxations proposed in Chapter 7. In the following section we showcase the potential practical strength of several constraint types obtained by the analysis of the complete outer description of different ordering polytopes in small dimensions. In Section 13.2 we present a practical implementation of the heuristic constraint selection proposed in Section 7.3. Applying our method to several benchmark instances of the Max-Cut problem yields major improvements compared to the state-of-the-art SDP approach.

13.1 Small Facets

First we compare relaxation (SDP$_4$) (which we defined in Section 4.2 and used in our computational experiments in the previous chapters) to the theoretically stronger relaxation (SDP$_5$) (defined in Section 7.2). We reconsider the facet classes (with respect to node permutations) of the linear ordering polytope $\mathcal{P}^n_{LOP}$ for $n = 7$ already used in Section 10.1. We have seen that (SDP$_4$) is exact for all except one of these facet classes. Conducting the same experiments for (SDP$_5$) shows that it identifies all facet classes correctly and thus seems to be a promising practical alternative to (SDP$_4$). But note that further computational experiments on (QOP) instances of more challenging size are needed to get a better idea of the practical benefits of using (SDP$_5$).

Next we try to recover the correct right hand side of the facet classes of $\mathcal{P}^5_{BTW}$. We again solve (SDP$_5$) to optimality using the standard settings of Sedumi [199]. In Table 13.1 we summarize the obtained deviations from the correct right hand sides.

Facet Type	(F1), (F2)	(F3)–(F8)	(F9)	(F10)	(F11)
Deviation	0	0.25	0.52	0.64	0.72

Table 13.1: Deviations of (SDP$_5$) for the facet classes of $\mathcal{P}^5_{BTW}$.

We conclude that adding the $9\binom{n}{5}$ inequalities (F3)–(F11) to (SDP$_5$) is a promising strategy to obtain tighter bounds, especially for instances of the weighted Betweenness Problem (wBP).

13.2 Heuristic Constraint Selection

In this section we give computational evidence for the heuristic constraint selection proposed in Section 7.3. We compare the Max-Cut relaxations $(\mathtt{MC}_2)$, $(\mathtt{MC}_3)$ and $(\mathtt{MC}_4)$ on random rudy-generated instances considered in [184, Section 7, Table 2].[1] We approximately solve the relaxations for the first instance of each problem type with the help of a dynamic version of the bundle method (for further details on the algorithmic approach see Sections 3.3 and 10.2). The first 200 function evaluations belong to Phase 1, where we work with relaxation $(\mathtt{MC}_2)$ to get a good approximation of the important triangle inequalities. In Phase 2 we use $(\mathtt{MC}_2)$, $(\mathtt{MC}_3)$ and $(\mathtt{MC}_4)$ respectively and apply 400 additional function evaluations. For the considered graphs we give the number of vertices n, the density d and the optimal solution $(\mathtt{MC})$. The running times on an Intel Xeon 5160 processor with 3 GHz and 2 GB RAM are given in seconds. We set t in $(\mathtt{MC}_3)$ to $\frac{1}{10}$ and report the results in Table 13.2.

				Phase 1		Phase 2					
graph type	n	d	$(\mathtt{MC})$	time	$(\mathtt{MC}_2)$	time	$(\mathtt{MC}_2)$	time	$(\mathtt{MC}_4)$	time	$(\mathtt{MC}_3)_{\ t=0.1}$
$G_{0.5}$											
	100	0.5	1430	15.2	1442.234	196.3	1442.207	1015.9	1439.551	70.8	1440.055
$G_{-1/0/1}$											
	100	0.99	127	14.3	129.134	269.9	129.070	1037.8	127.353	164.7	127.368
$G_{[-10,10]}$											
	100	0.5	1646	14.6	1740.228	354.7	1740.050	1017.6	1713.819	85.5	1715.363
	100	0.9	2121	15.2	2237.638	321.9	2237.247	1016.2	2206.912	79.9	2208.223
$G_{1,10}$											
	100	0.5	8189	15.6	8285.842	372.4	8285.539	1007.8	8261.225	80.2	8262.689
	100	0.9	13585	16.5	13659.112	276.9	13658.877	1018.1	13640.101	78.7	13641.268

Table 13.2: Comparing several SDP relaxations for the Max-Cut problem on rudy-generated instances with n nodes, density d and optimal solution $(\mathtt{MC})$. The running times are given in seconds.

$(\mathtt{MC}_3)$ yields the best tightness-cost-ratio. It is nearly as fast as $(\mathtt{MC}_2)$[2] and yields upper bounds very close to the ones of the expensive $(\mathtt{MC}_4)$. $(\mathtt{MC}_3)$ reduces the *sdp-gap* $= 100\frac{bound-bks}{bks}$, where *bks* denotes the best known solution, compared to $(\mathtt{MC}_2)$ by $\approx 10-25\%$ and e.g. solves the first instance from $G_{-1/0/1}$ within 40.8 seconds in the root node, whereas the state-of-the-art Branch-and-Bound algorithm Biq Mac proposed in [184] needs in average 651 (and at least 79) Branch-and-Bound nodes for the ten $G_{-1/0/1}$ instances (see again [184, Section 7, Table 2]).

We are confident that using $(\mathtt{SDP}_3)$ in a Branch-and-Bound algorithm will yield improvements for several instance classes from the literature, as the main limitation of Biq Mac is the quality of the upper bounds obtained by $(\mathtt{MC}_2)$.

Finally we want to showcase the different strengths of $(\mathtt{MC}_1)$– $(\mathtt{MC}_4)$ on a small instance. Therefore we apply them to a graph consisting of 2 pentagons with one common edge, see Figure 13.1.

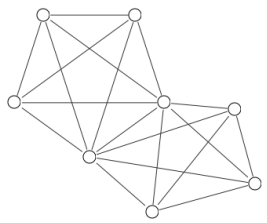


Figure 13.1: 2 pentagons with 1 common edge

[1] The instances can be downloaded from `http://biqmac.uni-klu.ac.at/biqmaclib.html`.

[2] In fact, for the instances considered, it is even faster, because the bundle method slows down when the improvement of the objective value stops. Exactly this happens for $(\mathtt{MC}_2)$ from function evaluation 250 onwards.

The Max-Cut value for this graph is 12, $(\mathtt{MC}_1)$ yields 12.5, $(\mathtt{MC}_2)$ gives 12.25 and $(\mathtt{MC}_3)_{\ t\geq 0}$ and $(\mathtt{MC}_4)$ are exact.

Chapter 14

Multi-level Crossing Minimization

14.1 Introduction

The current chapter is based on Section 4 of the paper "An SDP Approach to Multi-level Crossing Minimization" [45]. We use a dynamic version of the bundle method to obtain approximate solutions of the relaxation (SDP$_{IV}$) (with C and K defined in Section 8.3 and c equal to the zero vector - for further details on our algorithmic approach see Sections 3.3 and 10.2). Our extensive computational experiments on a large benchmark set of graphs show that this new approach in combination with an SDP based heuristic very often provides optimal solutions. We are able to compute optimal solutions for graph instances from the literature that have not been solved to optimality before.

We also compared our approach to a standard ILP formulation, solved via Branch-and-Cut within a generic ILP solver. Surprisingly, while the SDP approach dominates for denser graphs, the ILP turns out to be very fast for sparse, practical instances. It solves almost all instances of the Rome benchmark set, a standard graph drawing library. Yet, our experiments show that the SDP approach solves more instances to optimality than the ILP approach, although the former is not combined with a Branch-and-Bound scheme. This also suggests a new heuristic for Multi-level Crossing Minimization (MLCM) based on SDP which clearly outperforms the classical heuristics.

Having obtained optimal solutions for graphs of interesting size, we can for the first time evaluate heuristic solutions. We show that the upward planarization approach is very close to the optimum concerning the given leveling, while this is not true for the standard barycenter heuristic. For our studies, we collected a large benchmark set of leveled graphs, available at http://www.ae.uni-jena.de/Research_Pubs/MLCM.html.

Due to licensing issues and overall CPU time we conducted our experiments on two different machines. All SDP computations where conducted on an Intel Xeon E5160 (Dual-Core) with 24 GB RAM, running Debian 5.0. The algorithm itself runs on top of MatLab 7.7.

For comparison, we also considered a newly written ILP implementation (along the lines of [119]) using Branch-and-Cut. Thereby the 3-cycle inequalities are separated on the fly, instead of adding all of them initially. We do not specifically separate further inequalities such as those described in Proposition 8.6: It was observed by Healy and Kuusik [97] that even though the number of Branch-and-Bound nodes decreases, the additional effort needed to identify violated constraints—even of the simple cycle types (8.22) and (8.24)—leads to overall increased running times. We also evaluated the ILP variant without separation; as this approach resulted in clearly worse running times, we only report on the results the code with separation. These experiments were conducted on an Intel Xeon E5520 (Dual-CPU, Quad-Core) with 72 GB RAM, running Debian 6.0. The C++ code uses CPLEX 12.1 [117] (with default settings) as a Branch-and-Cut framework.

Each algorithm was run in 32-bit mode, effectively restricting it to 2GB RAM. Notice that both the

second machine as well as the implementation language C++ and the highly tuned commercial (I)LP solver can be expected to be faster than their SDP counterparts. Herein we are not so much interested in the exact running times, but in the order of magnitude. Not only can we assume that our setting can achieve such a comparison, we will in fact see that the SDP approach outperforms the ILP approach despite this setting.

We restrict the SDP approach to 1500 function evaluations of $f(\lambda, \mu)$, as the convergence process of the bundle method usually slows down before that point, independent of problem size ζ. After every fifth function evaluation we search for newly violated constraints at the current primal point. We add all constraints with violation > 0.001 to the bundle and additionally remove constraints with relatively speaking small associated Lagrangian multipliers ($\lambda_i < 0.01 \cdot \lambda_{\text{mean}}$). A further critical operation is the first-time initialization of the dual variables, where we choose the initial λ_i as "$\frac{2 \cdot \text{initial duality gap}}{\| \text{ total constraint violation } \|^2}$ · violation of constraint i". We maintain these parameter settings also for the computational experiments on Multi-level Verticality Optimization in the following chapter.

Standard heuristics and also some metaheuristics perform quite poorly for (MLCM) instances of sizes of our interest [120, 151]. Therefore we apply the SDP based heuristic described in Section 12.2 to obtain high quality feasible solutions. It would be interesting to compare metaheuristics like GRASP and tabu search [151] with our SDP heuristic on challenging instances.

In the next section we compare our approaches on a synthetic benchmark where we have control over this density parameter. In Sections 14.3 and 14.4 we apply the ILP and the SDP algorithm to real world graphs and well-known benchmark instances from the literature respectively. After that we take a closer look at the quality of the achieved lower and upper bounds in Section 14.5 and conduct a case study on how to get information on the set of all optimal solutions of an (MLCM) instance in Section 14.6. Finally in Section 14.7 we examine the effect of including Lovász-Schrijver cuts in the SDP relaxation for different ordering problems.

14.2 Graphs with Varying Densities

First of all we we compare our SDP approach applying the strongest relaxation (SDP_{IV})[1] with the SDP Branch-and-Bound approach building on (SDP_{II}) from [31] for Bipartite Crossing Minimization (BCM). (BCM) is a special case of (MLCM) where the number of levels is set to two. The first exact algorithm for this problem has been introduced by Jünger and Mutzel [120] which only performs well on small, sparse instances ($n \leq 12$) [31].

For our experiments, we used the random instances from [31]. These are generated with the Stanford GraphBase generator [130] which is hardware independent. Results are reported for graphs having $n = 14, 16, 18$ vertices on both layers. For each n, we consider graphs with densities $d \in \{0.1, 0.2, \ldots, 0.9\}$, i.e. with $\lfloor dn^2 \rfloor$ edges. For each pair (n, d), we report the average over 10 random instances.

We summarize the results of our experiments in Table 14.1. The times are given in seconds, "nodes" gives the average number of Branch-and-Bound nodes and "fe" denotes the average number of function evaluations of the bundle method required to prove optimality.

The SDP approach of Buchheim et al. allows for substantial improvements, independent of the one third slower machine used in [31].[2] The main reasons for this improvement are the use of a stronger SDP relaxation and the careful fine tuning of the bundle method, which allowed us to prove optimality at the root node, while [31] had to go through a few steps of branching before being able to prove optimality. Motivated by these results we plan to apply our SDP approach also to Crossing Minimization in Tanglegrams (TCM) (see the book of Page [170] for general information on tanglegrams). Baumann et

[1] We use the strongest relaxation (SDP_{IV}) as the number of violated Lovász-Schrijver cuts (8.13) stays manageable for all considered instances (for details see Section 14.7 below).

[2] Buchheim et al. carry out their experiments on an Intel Xeon 5130 processor with 2 GHz – for exact numbers of the speed differences see http://www.cpubenchmark.net/.

n	d	Buchheim et al. [31]		($\mathtt{SDP}_{IV}$) with bundle method		
		time	nodes	time	nodes	# fe
14	0.1	51.8	1	0.2	1	0.6
14	0.2	61.9	1	10.2	1	7.3
14	0.3	93.2	1	23.1	1	12.3
14	0.4	97.9	1	25.3	1	12.9
14	0.5	117.6	1	30.7	1	14.2
14	0.6	95.9	1	20.2	1	10.6
14	0.7	101.8	1	24.2	1	11.7
14	0.8	101.9	1	19.7	1	10.1
14	0.9	56.3	1	9.6	1	5.8
16	0.1	119.1	1	2.4	1	2.6
16	0.2	200.9	1	28.8	1	10.6
16	0.3	432.9	1.2	53.5	1	16.6
16	0.4	1432.0	2.8	306.3	1	42.9
16	0.5	1181.2	2.4	110.2	1	28.5
16	0.6	1186.8	2.2	89.0	1	24.4
16	0.7	916.9	1.8	79.1	1	21.9
16	0.8	444.92	1.2	57.3	1	16.5
16	0.9	224.13	1.0	35.6	1	10.8
18	0.1	343.2	1.0	8.9	1	4.0
18	0.2	491.9	1.0	57.9	1	12.5
18	0.3	1233.73	1.0	170.4	1	23.7
18	0.4	-	-	299.4	1	35.3
18	0.5	-	-	211.4	1	28.5
18	0.6	-	-	391.3	1	43.2
18	0.7	2624.98	2.0	314.6	1	34.8
18	0.8	2523.04	2.0	190.8	1	25.7
18	0.9	601.30	1.0	78.6	1	12.8

Table 14.1: Comparison of two SDP approaches on random bipartite graphs with varying size (n nodes on both layers) and density d. The running times are given in seconds, "nodes" gives the average number of Branch-and-Bound nodes and "# fe" denotes the average number of function evaluations of the bundle method required to prove optimality.

al. [19] showed that (TCM) can be formulated as (BCM) with additional equations on some products of ordering variables. Their computational study indicates that the SDP approach of Buchheim et al. [31] adapted to (TCM) is currently the best exact method.

We already argued at the end of Section 8.2 that the linear programming gaps become too large for dense instances, in order to allow practically efficient ILP methods to succeed in such cases; this argument is supported by the known results for two-layer crossing minimization [31]. To give further evidence we start out with considering a synthetic benchmark where we have control over this density parameter. We generated a set of instances having $p \in \{2, \dots, 20\}$ layers and $n \in \{8, \dots, 25\}$ vertices on each layer. For each combination of p and n, we consider random instances with equal densities $d \in \{0.1, 0.2, \dots, 0.9\}$ for all layers, where each potential edge has equal probability of being selected. For each triple (p, n, d), we report the average over 10 generated instances.

Table 14.2 summarizes our results. We restricted the ILP approach to 1 hour of computation per instance: We observe that the solved instances always require less than 1 minute (except for four instances with 24, 6, 3 and 1.5 minutes, respectively); for the unsolved instances the gaps are still very large after 1 hour and progress stagnates.

Unsurprisingly, we observe that the graph density is relatively unimportant for the SDP; while very sparse and dense graphs allow the SDP to find solutions quickly, even most of the instances with a more complicated cost structure ($d \approx 0.5$) can be solved within an hour. On the other hand, the ILP approach is applicable only to very sparse graphs: it can solve all instances with $d = 0.1$. In theses cases, it is an order of magnitude faster than the SDP. Yet, it solves not a single instance with $d \geq 0.2$ within 1 hour.

Regarding the two-level case, we note that the approach by Buchheim et al. was unable to solve some of the instances with $n = 18$ nodes per layer whereas our method meets the first difficulties for $n = 21$. There exists another ILP approach suggested by Healy and Kuusik [96] that considers 10 random instances with $p = 8, n = 12, d = 0.109$. We also tested the only still instance of these still available, observing equivalent behavior to our random instances.

14.3 Real-World Graphs

Motivated by the results above we now turn our attention to commonly used benchmark sets in the area of graph drawing, where the considered graphs are relatively sparse, and investigate our algorithms more deeply. Both instance sets described below are said to be at least similar to real-world instances; to our knowledge this is the first time that these instances are tackled in the context of any exact multi-level crossing minimization.

Rome graphs: The Rome graphs, originally proposed by Di Battista et al. [64], were obtained from a basic set of 112 real-world graphs. The collection contains 11,528 instances with 10–100 vertices and 9–158 edges and, although originally undirected, can be unambiguously interpreted as directed acyclic graphs, as proposed by [70].

North DAGs: The North DAGs were introduced in an experimental comparison of algorithms for drawing DAGs by Di Battista et al. [63]. The benchmark set contains 1,158 DAGs collected by Stephen North that were slightly modified by Di Battista et al. [63]. The graphs are grouped into 9 sets, where set i contains graphs with $10i$ to $10i + 9$ arcs for $i = 1, \dots, 9$.

Both instance sets contain regular graphs, which are not proper level graphs. As they have been regularly used as benchmarks for Sugiyama style drawings, we consider two different leveling approaches:

GKNV: As indicated in the introduction, the first step of the traditional Sugiyama approach is to level the given graph. There are multiple strategies to decide on a leveling; in these experiments, we consider the optimal LP-based algorithm by [79]. In this context, we can also evaluate traditional multi-level crossing minimization strategies: In the tables below, we will also give the number of crossings obtained by the level-wise barycenter heuristic (sweeping over all levels until the solution does not further improve).

		SDP									ILP
	$d =$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.1
p	$n,\ \zeta$	✓, time	✓, time	✓, time	✓, time	✓, time	✓, time	✓, time	✓, time	✓, time	✓, time
2	**20**,381	**10**, 45.1	**10**, 223.1	**10**, 578.3	**10**, 504.3	**10**, 478.6	**10**, 672.1	**10**, 513.5	**10**, 392.4	**10**, 319.6	**10**, 3.40
	21,421	**10**, 58.2	**10**, 246.5	**10**, 912.9	**7**, 1148.6	**8**, 1601.5	**9**, 1223.7	**10**, 1292.0	**10**, 740.7	**10**, 458.4	**10**, 15.67
	22,463	**10**, 117.8	**9**, 505.9	**9**, 788.5	**7**, 3863.0	**7**, 3621.3	**9**, 1781.3	**9**, 2866.2	**10**,1127.6	**10**, 923.4	**10**, 28.15
3	**16**,361	**10**, 56.0	**10**, 238.5	**10**, 601.9	**9**, 476.8	**10**, 559.1	**10**, 1184.8	**9**, 858.2	**10**, 576.5	**10**, 209.6	**10**, 3.99
	17,409	**10**, 130.4	**10**, 324.6	**8**, 738.8	**10**, 934.0	**7**, 863.9	**9**, 1967.9	**9**, 1300.2	**10**, 823.7	**10**, 557.2	**10**, 11.22
	18,460	**10**, 112.1	**10**, 804.5	**9**,1278.9	**8**, 1798.0	**7**, 1684.0	**8**, 1925.7	**8**, 2187.5	**9**, 840.9	**10**, 534.2	**10**, 23.70
6	**12**,397	**10**, 52.4	**10**, 279.3	**8**, 158.8	**9**, 982.2	**9**, 641.0	**8**, 226.4	**10**, 1262.7	**10**, 695.1	**10**, 281.1	**10**, 0.70
	13,469	**10**, 149.2	**9**, 772.7	**10**, 978.1	**6**, 773.5	**7**, 2898.5	**8**, 1331.7	**10**, 1079.8	**10**,1040.6	**10**, 688.4	**10**, 4.90
	14,547	**10**, 347.5	**8**, 923.9	**6**,1898.0	**5**, 2345.4	**2**, 2037.4	**8**, 3837.6	**5**, 3643.6	**9**,6235.5	**10**,1602.3	**10**,189.04
11	**10**,495	**10**, 97.1	**10**, 383.8	**10**, 987.8	**9**, 944.9	**9**, 2539.3	**10**, 1365.5	**10**, 1289.7	**10**,1138.9	**10**, 722.4	**10**, 0.45
	11,605	**10**, 248.1	**10**,1189.3	**10**,1861.7	**10**, 3192.6	**9**, 4443.3	**8**, 3471.7	**6**, 5518.5	**9**,2452.6	**10**,1171.3	**10**, 2.07
	12,726	**10**, 615.1	**9**,1843.7	**6**,7864.2	**6**, 8935.6	**8**, 8490.7	**2**, 7063.0	**4**, 8940.0	**9**,6088.1	**10**,2959.9	**10**, 50.52
20	**8**,561	**10**, 13.5	**10**, 356.7	**10**, 701.6	**10**, 926.2	**10**, 1145.9	**9**, 1179.3	**10**, 1220.6	**10**, 914.6	**10**, 591.5	**10**, 0.01
	9,721	**10**, 149.4	**10**,1395.8	**10**,2284.4	**10**, 2614.1	**9**, 3023.7	**9**, 4358.7	**10**, 3523.8	**10**,2822.5	**10**,2605.1	**10**, 0.21
	10,901	**10**,1000.9	**10**,3213.6	**10**,5979.1	**10**,10513.4	**4**,11407.5	**8**,12625.4	**10**,13976.2	**10**,7637.4	**10**,6430.9	**10**, 2.46

Table 14.2: SDP and ILP approaches on random graphs with representatively chosen values for the graph's density d, the number of nodes on each layer n and the number of layers p. "✓" denotes the number of instances solved to optimality (out of 10), "ζ" gives the dimension of the respective SDP cost matrices and "time" denotes the average time (in seconds) over the solved instances. For the ILP, no instance with $d \geq 0.2$ could be solved.

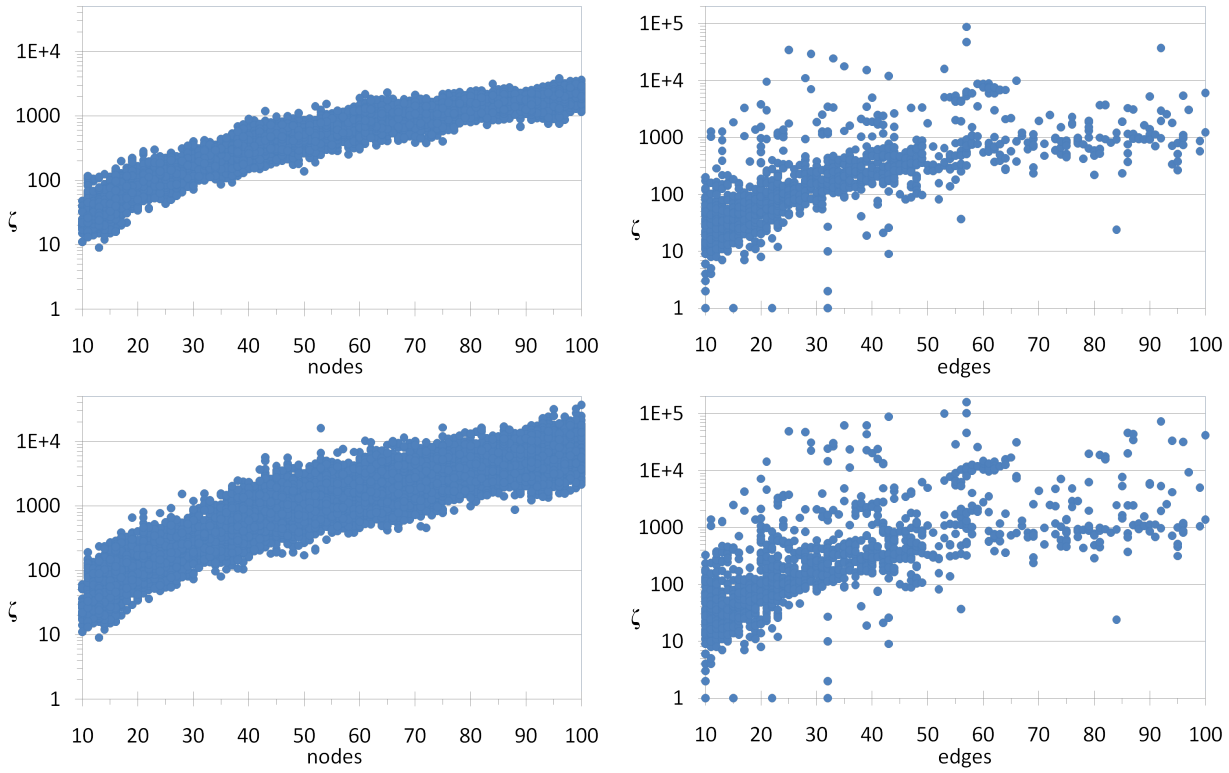


Figure 14.1: Correlation between original graph size and ζ, for the GKNV (on top) and UPL (on bottom) layering. Left: Rome graphs, right: North DAGs. Each dot denotes the dimension of at least one instance of the specified graph size. The diagrams w.r.t. GKNV and UPL show the same tendencies but the same graph size results in slightly larger dimensions for the UPL layering.

UPL: Recent algorithms have combined the first and the second step of Sugiyama's framework to obtain an upward planarization algorithm [40]. Thereby, a planarization P with few crossings is computed without the need for levels. Afterwards, P is fitted into the smallest leveling allowing the specified crossing configuration [41], in order to be applicable for Sugiyama's third step. We will also consider the layering obtained by this approach, as it allows a much smaller number of crossings in practice. This also allows us to deduce if (thinking inversely) the UPL approach gives a (near-)optimal number of crossings with respect to the finally computed layering.

Results: Recall that the matrix dimension ζ does not only depend on the original number of vertices (or edges), but on the derived proper level graph, i.e., also on the number of artificial vertices and the vertex distribution over the layers. Hence the algorithm will be mostly dependent on ζ rather than the original size. Figure 14.1 shows the dependency between these different metrics. We calculated all graphs with $\zeta < 900$ and $\zeta < 1500$ for the Rome and North instances, respectively, and summarize the results in Table 14.3. Our benchmark instances, except for very small graphs, are all sparse: The average density of the considered instances with $\zeta > 300$ is 0.10, 0.11, 0.12, and 0.12 for the combinations, Rome-GKNV, Rome-UPL, North-GKNV, and North-UPL, respectively. For the ILP approach, we applied a time limit of 4 hours for each instance with $\zeta < 900$, and 16 hours for $\zeta < 1500$.[3] These ILP time limits were chosen such that the SDP approach always finished its (at most) 1500 function evaluations within that time, i.e., the ILP approach has at least as much CPU time as the SDP approach.

[3]These large computation times did not leave time for the Rome instances with $\zeta \in [900, 1500)$, so we restricted ourselves to the more diverse North graphs.

Table 14.3 summarizes our experiments. The first and most surprising result is that both approaches are in fact very successful on these real-world instances, as only few instances remain unsolved by either of these approaches. In accordance with our finding with the random graphs, we observe that the ILP is usually faster. Yet we also observe that the SDP is stronger with respect to overall solvability: It solves all instances except for two North-GKNV instances; the ILP approach fails for 21 graphs, including the aforementioned 2. When both algorithms fail, the SDP approach obtained tighter pairs of upper/lower bounds: 498/518 and 853/854, in contrast to the ILP's 418/499 and 336/854. We conclude that for sparse graphs one should usually try the ILP first; when it fails to prove an optimal solution within reasonable time, the SDP approach still has a good chance of succeeding on these hard instances.

Analyzing the distinct benchmark sets, we observe that the traditional leveling and crossing minimization heuristics leave plenty of room for improvement when considering the minimum number of crossings. In contrast to this, the graphs leveled by the UPL approach only allow much smaller improvements. In fact they show that the upward planarization approach by [40] gives near-optimal solutions for its respective leveling. We also observe that the fact that UPL produces more, but smaller, levels, and requires fewer crossings, is beneficial for both exact approaches: they solve all UPL instances, while the GKNV instances are harder.

14.4 Polytopes and Further Instances From Literature

We continue our experimental study by looking at further instances of interest. Often, one considers the graphs modeling the incidence relation between faces (corner, edge, 2D-face,...) of an (LP-)polytope, and hence we are interested in drawing them within a Sugiyama framework. These graphs are naturally very dense. Table 14.4 shows that we can solve such instances as long as the dimension of our matrix is within reasonable bounds. We observe that the SDP approach is clearly beneficial over the ILP. Even for polytope-based instances that cannot be solved to optimality by either approach, the bounds obtained by the SDP are clearly stronger.

We also considered the instances from the Graphviz gallery [88] as suggested recently by Gange et. al [78]. We only report on the non-trivial instances. We observe that our ILP implementation gives comparable running times to those of [78]; thereby our approach is much simpler. This observation validates the finding by [96] which already suggested that additional separation routines need not pay off in practice. Additionally Gange et. al also suggested a SAT-based approach that was however dominated by their ILP.

Finally we report on the traditional real-world instances MS88 [155] and SM96 [194]. For the latter, the prior publications only considered a subgraph consisting of three layers, due to the graph's complexity. For the first time, we also report optimal results for the full graph, which is illustrated in Figure 14.2. Again we can observe that the SDP approach is beneficial when considering the more complex instances.

14.5 Quality of Bounds

Now let us take a closer look at the quality of the lower and upper bounds from the previous sections. First, we are interested in how these bounds develop over time, illustrated in Figure 14.3. In the SDP approach, the best known upper bound is typically found in the first few iterations and often turns out to be optimal in the end. The SDP lower bound improves quickly in the beginning; its progress slows down very smoothly. Thus when stopping the approach at some point it is quite easy to assess the further gap improvement that could be achieved. When the SDP approach cannot close the gap, we assume that this is usually due to the quality of the lower bound relaxation. For the ILP on the other hand, the pure linear programming relaxation only gives a weak lower bound: progress is made only via branching. Also the ILP upper bound (obtained via a feasibility pump) takes longer to find a good (or an optimal) solution.

			optimal, imp					optimal, ni			imp/ni	ILP	
		$\zeta <$	#	cr (std)	diff (max)	t_{lb}	t_{ub}	#	t_{lb}	t_{ub}	#/#	no	time
GKNV	Rome	300	**2015**	11.14 (8.24)	3.7 (18)	20.55	0.25	**1538**	2.38	0.02	0/0	0	0.26
		600	**2572**	26.01 (14.55)	11.2 (52)	4:13	9.34	**11**	2:03	0.19	0/0	0	7.46
		900	**1325**	42.01 (23.02)	24.87 (75)	31:55	101.14	**0**	–	–	0/0	**11**	3:57
	North	300	**90**	16.78 (33.05)	2.04 (10)	16.77	0.09	**316**	3.77	0.03	0/0	**1**	8.20
		600	**80**	25.26 (51.86)	3.81 (20)	4:05	5.88	**35**	3:04	0.32	0/**1**	**6**	9.82
		900	**36**	29.25 (32.99)	8.03 (29)	17:11	22.56	**13**	17:33	2.36	0/0	**1**	1:14
		1200	**29**	47.48 (54.01)	8.24 (32)	3:28:34	3:39	**7**	6:24	4.49	0/**1**	**2**	1:57
		1500	**11**	64.18 (63.22)	8.45 (17)	6:35:53	6:35	**2**	5:14	6.98	0/0	0	8:58
UPL	Rome	300	**136**	1.38 (1.62)	1.18 (4)	5.84	0.10	**442**	3.97	0.05	0/0	0	0.04
		600	**617**	3.12 (2.11)	1.52 (8)	2:44	2.71	**711**	2:36	0.92	0/0	0	0.25
		900	**731**	5.23 (2.95)	2.12 (9)	25:13	21.13	**338**	18:29	3.35	0/0	0	0.93
	North	300	**30**	4.73 (4.27)	1.57 (5)	9.85	0.09	**126**	3.02	0.03	0/0	0	0.04
		600	**45**	5.98 (4.68)	1.91 (5)	3:14	1.10	**43**	1:31	0.3	0/0	0	0.61
		900	**14**	9.50 (6.73)	2.79 (6)	18:24	25.79	**20**	13:24	2.65	0/0	0	18.86
		1200	**11**	11.55 (17.30)	2.27 (6)	1:31:38	38.17	**9**	2:04:10	5.53	0/0	0	1:03
		1500	**11**	35.00 (27.17)	4.64 (9)	4:40:28	5:04	**5**	2:17:12	7.74	0/0	0	10:19

Table 14.3: The results for the SDP approach on real-world benchmark instances with crossing number > 0. The results are split into four categories: whether or not SDP found a proven optimal solution ("optimal"), and whether this solution was better than the one from the respective heuristic ("imp" vs. "ni"=no improvement) (see benchmark description). "#" denotes the number of instances, "cr (std)" reports mean and standard deviation of the optimal crossing numbers, "diff (max)" gives the average and maximal difference between the optimal and the heuristic solution. t_{lb} and t_{ub} give the average time (in sec, in min:sec or in h:min:sec respectively) to compute the lower bound (via the relaxation ($\texttt{SDP}_{IV}$)) and the upper bound (via the SDP rounding heuristic described in Section 12.2), respectively. We also give the number of instances not solved to optimality by the ILP approach ("no"=not optimal) as well as the average solution time over the other instances.

Type	Instance	p	d	ζ	SDP z_*	SDP t_{lb}	SDP t_{ub}	ILP time	prev ILP time
Polytopes	Tetrahedron	3	0.5	28	22	0.09	0.03	0.08	—
	Octahedron	3	0.29	110	80	10.61	0.05	2.62	—
	Cube3	3	0.29	110	80	10.87	0.06	3.14	TO (1h) [J]
	Dodecahedron	3	0.13	692	393/394	4:40:07	2.03	18h: 132/427	—
	Icosahedron	3	0.13	692	393/394	4:37:23	1.98	16h: 174/401	—
	Cube4	4	0.14	921	1192/1195	7:10:19	7.10	51h: 197/1334	—
	Soccer ball	3	0.04	6272	1627/2353	91:23:06	11:10	52h: 118/2630	—
Graphviz	switch	6	0.2	169	20	1.87	0.05	0.66	0.75 [G]
	unix	11	0.19	176	0	0.24	0.01	0.01	—
	world	9	0.09	815	46	1:11:42	2:07	2:45	TO (1 min) [G]
	profile	9	0.08	846	37	51:32	2:02	2.84	6.81 [G]
Other	MS88	3	0.24	217	91	2.62	0.17	5.02	4:29 [J]
	SM96-3L	3	0.07	615	13	1:19	7.14	0.18	5:32 [J]
	SM96-full	7	0.10	915	162	53:16	12.75	3:05:05	—

Table 14.4: Results for polytopes and further known instances. *Cube3(4)* denotes a 3(4)-dimensional cube. z_* gives the optimal objective value (or final lower and upper bound), and t_{lb} and t_{ub} denote the time for the lower and upper bound computation, respectively. *ILP-time* gives the time of the ILP approach; when the process terminated due to insufficient memory (2GB restriction due to 32bit), we give the respective time up to this point and the final lower and upper bound. Due to its complexity, we only computed 50 function evaluations of $f(\lambda, \mu)$ for soccer ball. The last column gives the reported running time in the cited paper to obtain the optimal solution (or TO=timeout). All times are given in sec, in min:sec or in h:min:sec. "[G]" and "[J]" represent [78] and [119], respectively.

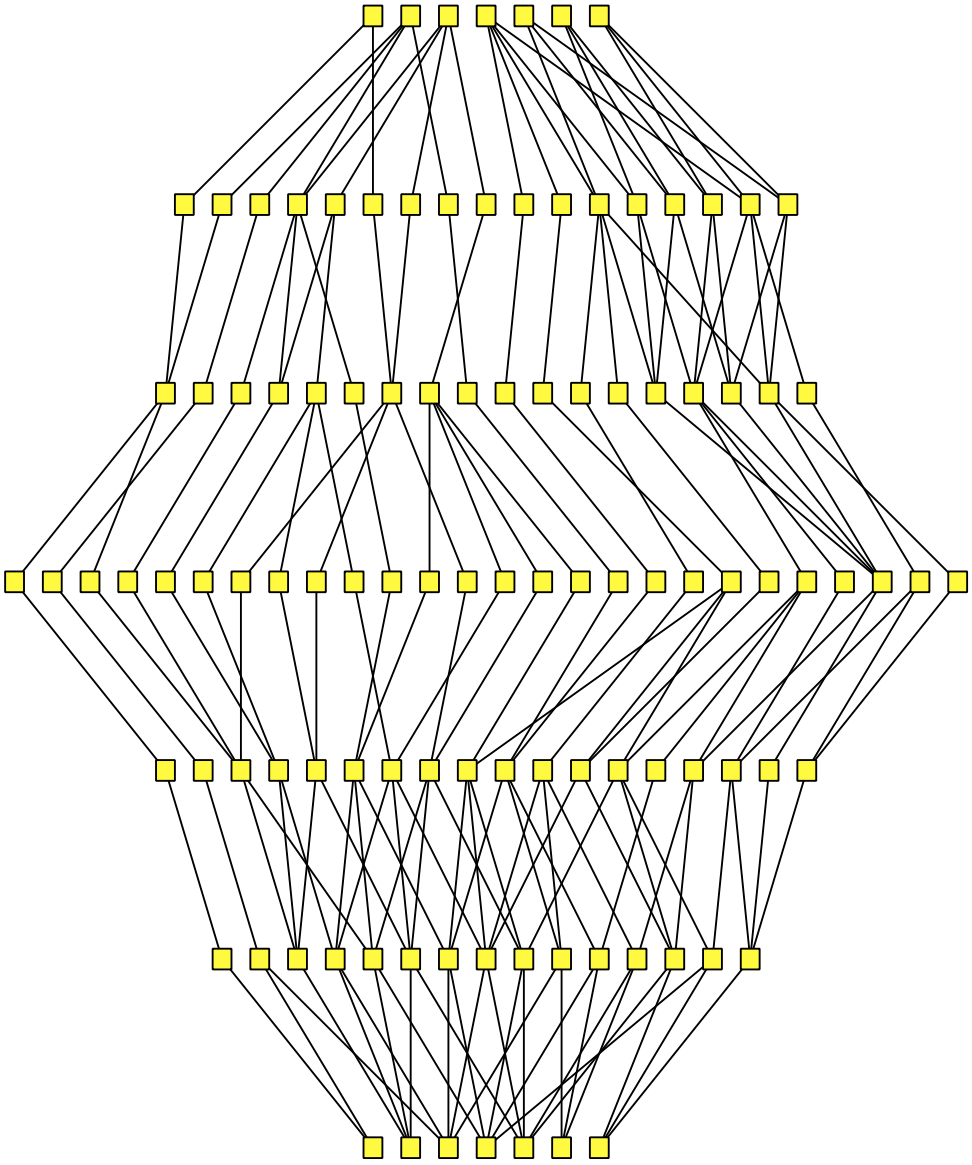
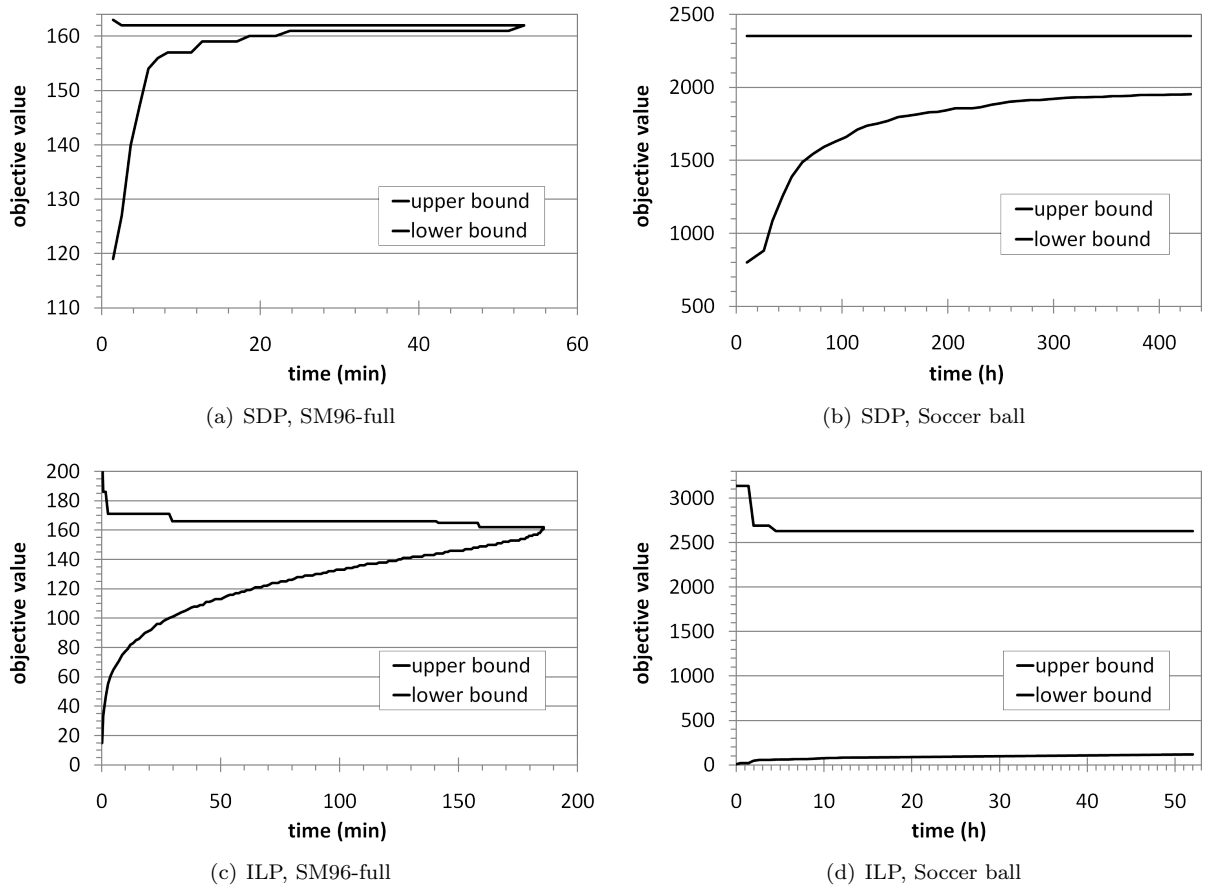


Figure 14.2: Optimal ordering for *SM96-full* (graph proposed by [194]), requiring 149 crossings.



(a) SDP, SM96-full

(b) SDP, Soccer ball

(c) ILP, SM96-full

(d) ILP, Soccer ball

Figure 14.3: Improvements of lower and upper bounds over time for two exemplary instances. Note the different scales at the axis between the SDP and ILP approaches.

While the SDP nearly closes the gap of "SM96-full" after one fifth of its running time, the ILP achieves the same quality of bounds only in the last tenth of its run. This is another argument showing why the SDP approach is more appropriate to provide good bounds for instances that cannot be solved to optimality, such as the "Soccer ball" instance. When using the ILP, both the lower and the upper bound stagnate very far off the optimum, despite the use of branching.

To close our practical investigation, we examine the SDP upper bound heuristic in more detail. We are interested in the importance of the repair strategy as for n objects there exist $n!$ orderings but $2^{\binom{n}{2}}$ -1/+1 vectors. We use the quite diverse instances from Section 14.4 as our test set. Table 14.5 shows that most of the time the hyperplane rounding already produces orderings and thus also the best -1/+1 vector found is an ordering (except for the "profile" instance). When we generate a -1/+1 vector that does not represent an ordering, only few repair steps are needed. Therefore the quite naïve repair strategy is not a critical nor a time consuming component of the upper bound heuristic.

14.6 On Finding all Optimal Solutions

In general the upper bound heuristic gives at most one arbitrary optimal solution. To gain information about the set of all optimal solutions of an `(MLCM)` instance, individual analysis is needed.

Let us showcase this by taking a look at instance 'LOP3' describing the incidence relation between faces of the linear ordering polytope of dimension 3. We use the labels from Figure 14.4 and denote by

Type	Instance	z_*	m_*	no rep	rep	rep steps
Polytopes	Tetrahedron	22	22	1000	0	—
	Octahedron	80	80	999	1	1.00
	Cube3	80	80	999	1	1.00
	Dodecahedron	394	394	999	1	6.00
	Icosahedron	394	394	997	3	3.33
	Cube4	1195	1195	1495	5	3.20
	Soccer ball	2353	2353	1992	8	2.75
Graphviz	switch	20	20	1000	0	—
	unix	0	0	995	5	2.00
	world	46	46	23986	14	3.36
	profile	37	34	21930	70	2.07
Other	MS88	91	91	999	1	1.00
	SM96-3L	13	13	4456	44	1.57
	SM96-full	162	162	2992	8	2.00

Table 14.5: Analysis of the importance of the repair strategy. m_* gives the objective value of the best found -1/+1 vector, thus $m_* \leq z_*$. "no rep" gives the number of trials for which the -1/+1 vector obtained by Goemans-Williamson hyperplane rounding already represents a feasible ordering of the nodes on the layers; inversely, "rep" counts how often a -1/+1 vector was encountered that did not represent an ordering. Considering the latter, "rep steps" gives the average number of repair steps needed to fix such a -1/+1 vector that does not represent an ordering.

'Layer 1' the layer with 6 vertices, by 'Layer 2' the layer with 12 vertices and by 'Layer 3' the layer with 8 vertices. There exist at least 192 different optimal orderings. They can be described in the following way:

1. All pairs of vertices from Layer 1 except (0,5), (1,3), (2,4) have edges leading to one common vertex on Layer 2. Now we put one vertex from each of the three pairs above to the positions 1–3. We have 2^3 different possibilities to select the vertices. Next, we choose one of the 6 possible permutations of the selected vertices. Finally let a selected vertex be at position i, then we put the respective other vertex of the pair to position $7 - i$.

2. Now let one of the 48 orderings from above be given. We denote the ordering by (i, j, k, l, m, n). Then there are at least four optimal orderings on Layer 2. The first one is $(ij, ik, jk, il, jl, im, jn, km, kn, lm, ln, mn)$ and the other three can be produced by changing positions 6,7 and/or positions 8,9 in the ordering.

3. Let again one of the 48 optimal orderings on Layer 1 be given as above. Then we request the ordering $(ijk, ijl, ikm, jkn, ilm, jln, kmn, lmn)$ for the vertices on Layer 3.

In sum, this gives 192 optimal orderings. We have checked optimality for the given orderings and we also used local search that could not find further optimal ones.

14.7 Practical Comparison of Semidefinite Relaxations

Finally we showcase that including the Lovász-Schrijver cuts (4.13)/(8.13) in the SDP model yields worse results for the Linear Ordering Problem (`LOP`) but essential improvements for several other types of ordering problems with linear-quadratic cost structure.

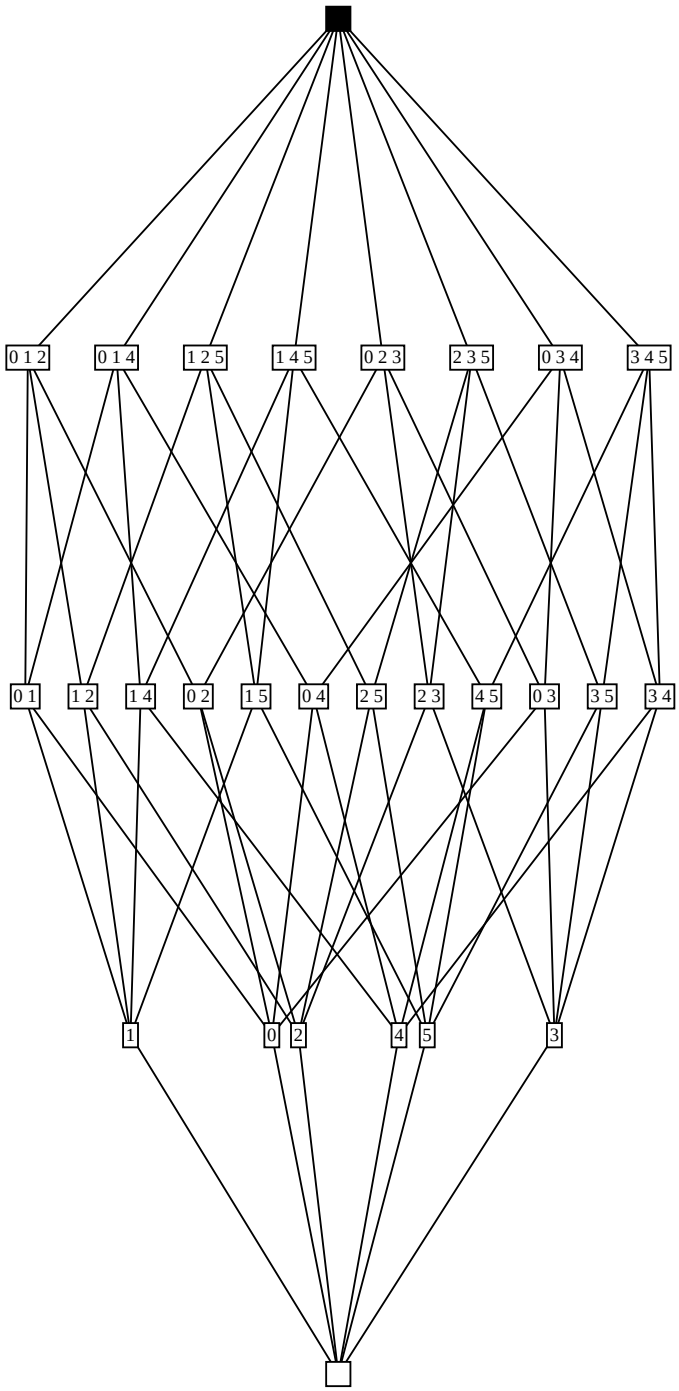


Figure 14.4: Drawing (not optimal) of the linear ordering polytope of dimension 3.

For our analysis we use selected instances form Sections 10.2, 11.1 and 14.2. The effects demonstrated on this sample apply to all instances of the respective problem types. Furthermore the benefits of including the Lovász-Schrijver cuts even increase for the Single Row Facility Layout Problem, (MLCM) and Multi-level Verticality Optimization, as these problem types exhibit a more complex cost structure than the minimum Linear Arrangement Problem (minLA) and (BCM) respectively.

In Table 14.6 we compare the number of inequalities of the different constraint types that are considered by a dynamic version of the bundle method (using the strongest relaxation ($\texttt{SDP}_4$)/($\texttt{SDP}_{IV}$)) at the last function evaluation. We examine two (LOP) instances from Table 10.3, a (minLA) instance from Table 11.1 and a random (BCM) instance with 16 nodes on each layer and a density of 40 % from Table 14.1. For the (LOP) instances the number of Lovász-Schrijver cuts (4.13)/(8.13) is very large and therefore constricts the overall performance of the algorithm. In contrast the number of Lovász-Schrijver cuts stays quite small for the two instances with quadratic cost structure.

graph	problem type	ζ	# (4.11)/(8.10)	# (4.12)/(8.12)	# (4.13)/ (8.13)
pal 31	(LOP)	466	5803	14339	746559
N-p50-05	(LOP)	1226	4874	6800	399720
cube5	(minLA)	497	5730	36768	192
bcm.16.40	(BCM)	241	1262	64903	2524

Table 14.6: Number of constraints considered by the bundle method. "ζ" gives the dimension of the respective SDP cost matrices.

Despite of their small number the Lovász-Schrijver cuts help a lot to tighten the relaxation of ordering problems with quadratic cost structure as can be seen from Table 14.7.

graph	# fe	opt	($\texttt{SDP}_2$)/($\texttt{SDP}_{II}$)	($\texttt{SDP}_4$)/($\texttt{SDP}_{IV}$)
cube5	300	496	490.28	491.47
	400		490.91	492.11
	500		491.21	492.32
	600		491.32	492.41
bcm.16.40	300	1397	1395.18	1395.42
	400		1395.62	1395.88
			1396.01 (595)	1396.01 (455)

Table 14.7: Gain of relaxation tightness through Lovász-Schrijver cuts. "# fe" denotes the number of function evaluations executed by the bundle method. In the last line of the table we give in brackets the number of function evaluations needed to prove optimality for the (BCM) instance.

Chapter 15

Multi-level Verticality Optimization

15.1 Introduction

This chapter is based on Section 5 of the paper "Exact Approaches to Multi-level Vertical Orderings" [42] and Section 5 of the paper "Multi-level Verticality Optimization: Concept, Strategies, and Drawing Scheme" [43]. We compare the relative benefits of the different drawing schemes and solution methods discussed in Chapters 8 and 9. Therefore we apply the exact approaches and heuristics proposed in Sections 8.2, 8.3 and 9.4–9.7 to solve Multi-level Crossing Minimization (MLCM) and Multi-level Verticality Optimization (MLVO) on a variety of test sets.

The computations for the heuristic and semidefinite approaches were conducted on an Intel Xeon E5160 (Dual-Core) with 24 GB RAM, running Debian 5.0 in 32bit mode. The SDP algorithm runs on top of MatLab 7.7, whereas the heuristics are implemented in C++. For the linear and quadratic programming approaches to (MLVO) we use CPLEX [117] 12.1's Branch-and-Cut framework and the applicable form of the degree constraints; the 3-cycle inequalities are separated dynamically, implemented in C++. Due to licensing issues, these algorithms are conducted on an Intel Xeon E5520 (Dual-CPU, Quad-Core) with 72 GB RAM, running Debian 6.0 in 32bit mode. Notice that this machine/software configuration can safely be considered speed-wise stronger than the one used for the SDP approach.

For approximately solving the semidefinite relaxation (SDP_{IV}) for (MLVO) we use a dynamic version of the bundle method (with C, c and K defined in Section 9.7 - for further details on our algorithmic approach see Sections 3.3 and 10.2). Additionally we apply the SDP based heuristic described in Section 12.2 and evaluated in Section 14.5 to obtain high quality feasible solutions. Our SDP approach leaves some room for further incremental improvement as we stop the bundle method after 1500 function evaluations to control the overall computational effort (for further details on the parameter settings of the bundle method see Section 14.1).

For the heuristic, we give the total running time and best found solution, considering 500 independent runs. We observe that while (for larger graphs) this is beneficial to fewer runs, there are nearly no more improvements in the solution quality when further increasing this number.

We consider input graphs from three different sources, which are often considered in related experimental investigations, e.g., [78, 96, 97, 119]. Table 15.1 gives the instances' central properties. All graphs considered in this chapter (including their optimal solutions, where available), as well as an implementation of the non-planar drawing style, are online available at http://www.ae.uni-jena.de/Research_Pubs/MLVO.html.

Polytopes. Often, one considers the graphs modeling the incidence relation between faces (corner, edge, 2D-face,...) of an (LP-)polytope, and hence we are interested in drawing them. We choose from a wide variety starting with a simple 3-dimensional tetrahedron to the face polyhedral body of a soccer ball.

Graphviz gallery. The gallery [88] constitutes a set of various real-world graphs from different appli-

cations. We only consider the largest of these graphs, as only they constitute difficult problems for our approach.

Other literature. The two worldcup instances, visualizing soccer world cup results of the full history up until the specified year, were proposed in [3, Fig. 12&13]. The graphs MS88 [155] and SM96 [194] are well known instances recurring in multiple publications, and represent certain social networks. Due to the size of the latter, a 3-level subgraph of SM96 is also often considered. As this subgraph includes many originally LEDs even on the lowest and highest level, we cannot reasonably consider this reduced graph w.r.t. Non-proper (MLVO).

In the following section we evaluate the different exact approaches to (MLVO). Based on the computational results we decide to concentrate on the SDP approach in the remaining experiments. In Section 15.3 we apply exact and heuristic methods to proper and non-proper graphs from the literature. Motivated by the encouraging computational results (small gaps between upper and lower bound and reasonable running times) we consider two well-known large benchmark sets containing real world graphs in Section 15.4. In Section 15.5 we practically compare (MLVO) to the closely related (MLCM). There we also experimentally examine several ways of combining the two optimization strategies. Finally we showcase the visual results and relative benefits of the different optimization goals and drawing strategies in Section 15.6.

15.2 Experimental Comparison of Exact Approaches

We start with evaluating the alternatives to the SDP approach (theoretically discussed in Section 9.5 and 9.6). We already argued why the linearization of (OM') and (OM) will be unfavorable compared to the SDP. Hence (concentrating on Proper (MLVO)) it remains to evaluate solving (DM') directly (using CPLEX's built in QP solver) and its linearization (introducing the variables $d_{e,i}$ described in Section 9.6). We denote these approaches by *DMQ* and *DML*, respectively. Furthermore, we want to investigate the influence of the quadratic objective function w.r.t. to the program's solvability. Therefore we consider the ILP which only minimizes $\sum_{e \in E'} d'_e$, instead of the sum of squares. Yet note that this ILP clearly does *not* really solve (MLVO) as defined in Chapter 9. We denote this algorithm by *LC*.

For both alternative approaches *DMQ* and *DML*, and even for *LC*, we observe running times that are orders of magnitudes larger than the SDP's, cf. Table 15.2. This is mainly due to the weak lower bounds and the resulting large number of required Branch-and-Bound nodes. Recall that the SDP results are always obtained without any branching. In fact, the non-SDP algorithms run out of memory in all but the smallest instance. At the end, they obtain clearly weaker lower and upper bounds than the SDP approach, although they require much more CPU time. We conclude that these approaches are no match for the SDP and concentrate on the latter in the following.

15.3 Polytopes and Further Instances From Literature

We conducted the (MLVO) experiments for proper and non-proper graphs. For the SDP approach, we consider both narrow and wide alignment scheme, whereas for the heuristics we only offer details for the aesthetically more pleasing wide alignment scheme. Table 15.3 gives an overview of our results. We observe that the semidefinite relaxation (SDP_{IV}) is tight enough to find and prove optimal solutions, supported by our SDP based upper bound heuristic, for the smaller instances, and gives surprisingly small gaps for the instances of challenging size.

For the heuristics we observe that the pure median and barycenter heuristic behave very similar but only give weak results. The local search routines are still fast and offer vastly superior solutions. Interestingly, due to the multiple runs performed for each instance, in turns out that it is usually most beneficial to start with a random initial order, than one based on the median or barycenter heuristic; this avoids to repetitively find the same weak local optima. Generally, 2-opt gives slightly weaker results than sifting or

			Proper								Non-proper							
	Instance	p	$\|V'\|$	$\|E'\|$	ω'	dens.	ζ	d_C	ζ^+	d_C^+	$\|V\|$	$\|E\|$	ω	dens.	ζ	d_C	ζ^+	d_C^+
Polytopes	Tetrahedron	3	14	24	6	0.50	28	0.58	46	0.40								
	Octahedron	3	26	48	12	0.29	110	0.45	199	0.17								
	Cube3	3	26	48	12	0.29	110	0.45	199	0.27								
	Dodecahedron	3	62	120	30	0.13	692	0.24	1306	0.13				*always proper*				
	Icosahedron	3	62	120	30	0.13	692	0.24	1306	0.13								
	Cube4	4	80	208	32	0.14	921	0.25	1985	0.10								
	Soccer ball	3	182	360	90	0.04	6272	0.09	12016	0.05								
Graphviz	switch	6	48	64	8	0.20	169	0.22	169	0.22				*already proper*				
	unix	11	59	66	11	0.19	176	0.16	606	0.04	41	48	7	0.06	77	0.22	232	0.07
	world	9	116	137	20	0.09	815	0.11	1711	0.04	48	69	9	0.07	132	0.27	325	0.10
	profile	9	92	116	28	0.08	846	0.14	3403	0.02	61	85	14	0.06	309	0.21	820	0.05
Other	MS88	3	37	80	15	0.24	217	0.38	316	0.26				*already proper*				
	worldcup86	4	35	55	19	0.19	223	0.30	685	0.08	25	45	11	0.22	92	0.46	221	0.18
	worldcup02	4	50	65	23	0.11	411	0.23	1013	0.07	31	46	14	0.15	149	0.34	365	0.13
	SM96-3L	3	61	58	26	0.07	615	0.16	976	0.11				*not applicable*				
	SM96-full	7	108	179	26	0.10	915	0.13	2276	0.05	63	134	14	0.08	295	0.26	638	0.11

Cube3 and Cube 4 correspond to a 3- and 4-dimensional cube, respectively.

$\omega^{(\prime)}$ denotes the maximum width of any level, *dens.* the graph's density relative to the case of all possible edges.

The columns ζ and d_C (ζ^+ and d_C^+) give the resulting dimension and density of the SDP cost matrix for the narrow (wide) alignment scheme, respectively.

Table 15.1: Instance properties

	DMQ			*DML*			*LC*			(SDP$_{IV}$) with bundle method	
Instance	v_*	bb	time	v_*	bb	time	(v_*)	bb	time	v_*	time
Cube3	262	3.5	1:57:16	262	0.3	0:13:29	(94)	1.4	0:33:59	$261^{+}1$	0:01:34
Icosahedron	115/–	0.5	[174h]	166/3566	0.4	[99h]	(122/510)	0.6	[105h]	$3046^{+}34$	4:51:06
profile	{566/–}	{0.7}	{240h}	583/1565	0.3	[51h]	(177/279)	0.5	[105h]	$1303^{+}9$	7:09:51
SM96-full	{100/–}	{0.4}	{240h}	138/1595	0.3	[42h]	(123/364)	0.4	[57h]	$1212^{+}13$	8:47:37

Table 15.2: Comparing *DMQ*, *DML*, and *LC* to the SDP approach on selected instances. The column *bb* gives the number of Branch-and-Bound nodes in millions. Number of hours in square brackets denote when the program runs out of memory (32bit), data in curly brackets denote when the program was terminated after 10 days, as the lower bounds stagnated for over 100h. v^* gives either the optimal solution or the final lower bound and the absolute gap to the upper bound: "$a^{+}b$" means lower bound a, absolute gap b and upper bound $a+b$; when the gaps become large, we write a/c instead, where c is the upper bound.

both (i.e., using both methods alternatingly). The latter two variants are virtually indistinguishable w.r.t. solution quality, but *both* (which starts with 2-opt) is usually faster.

Comparing the upper bounds obtained by the SDP to our special purpose local search heuristics, there is no clear winner. We conclude that simply using both—as they run fast anyhow—may be the best alternative.

The running time of the SDP lower bound computation is mainly dependent on ζ. Hence it is not surprising that the somehow "nicer" wide alignment scheme – requiring a much larger matrix Z – comes at a non-trivial cost w.r.t. the running time. Dropping the LEDs and solving the semidefinite program for Non-proper `(MLVO)` with a smaller but more tightly packed cost matrix instead, allows us to go well beyond the graph sizes to which exact approaches to Proper `(MLVO)` and `(MLCM)` (which cannot be directly applied to a non-proper setting) are restricted. The general behavior of the heuristic approaches is similar (although faster, of course) to the observations noted above.

15.4 Real-World Graphs

Motivated by these results, we now investigate our algorithms on two well-known larger benchmark sets, which were also considered for `(MLCM)` in Section 14.3. The *Rome graphs* [64] contain 11,528 instances with 10–100 vertices and, although originally undirected, can be unambiguously interpreted as directed acyclic graphs, as proposed in [70]. The *North DAGs* [63] contain 1,158 DAGs, with 10–99 arcs. Consistent with Section 14.3 (where we give more details on benchmark sets and layering methods), we consider two different ways of layering the graphs of both benchmark sets: the optimal linear programming based algorithm by [79] and the layering resulting from applying an upward planarization [40]. Both yield similar results in terms of `(MLVO)` running time and solvability.

Our main finding is that the observations from the previous section w.r.t. the heuristic variants hold. Yet, as the layerings introduce many more LEDs than the graphs considered before, the advantage of not requiring LEDs becomes even more pronounced: considering the largest graphs of the North DAGs (Rome graphs) with originally more than 90 edges (nodes), a single run of the heuristic requires only 6ms (2ms) on average, whereas the proper graphs require 1.8sec (0.8sec, resp.).

Similarly, the SDP approach is applicable to all Rome and nearly all North graphs (98%) in the non-proper setting, as the approach works well up to $\zeta \approx 5000$. It yields average gaps ranging from 5% to 80% with growing instance size. For small instances with up to 30 nodes (Rome graphs) or 60 edges (North DAGs) there is a chance ranging from 70% to 25% to solve instances to optimality. Using the proper drawing scheme, the SDP approach is applicable to 80% of the graphs with originally up to 60 nodes (Rome graphs) or 40 edges (North DAGs) and yields average gaps ranging from 10% to 80%.

		Proper (MLVO)							Non-proper (MLVO)						
		narrow - SDP		wide - SDP		wide - Heuristic			narrow - SDP		wide - SDP		wide - Heuristic		
	Instance	v_*	time	v_*	time	v_{50}	v_{500}	time_{500}	v_*	time	v_*	time	v_{50}	v_{500}	time_{500}
Polytopes	Tetrahedron	48	2.27	48	2.27	48	48	0.09	*always proper*						
	Octahedron	$261^{+}1$	0:02:37	$239^{+}5$	0:03:28	244	244	0.70							
	Cube3	$261^{+}1$	0:01:34	$239^{+}5$	0:04:11	244	244	0.71							
	Dodecahedron	$3051^{+}27$	3:31:58	$1815^{+}81$	29:55:48	1837	1834	10.78							
	Icosahedron	$3046^{+}34$	4:51:06	$1807^{+}61$	27:10:23	1837	1834	11.22							
	Cube4	$6336^{+}86$	7:57:46	$5279^{+}121$	80:49:47	5364	5360	33.74							
Graphviz	switch	$53^{+}1$	0:05:54	$53^{+}1$	0:05:54	54	54	0.44	*already proper*						
	unix	111	0:04:27	$58^{+}5$	1:19:41	74	69	1.73	49	5.3	$30^{+}3$	0:10:11	34	33	0.28
	world	$620^{+}41$	6:33:10	$331^{+}95$	54:30:21	486	479	14.84	129	0:02:06	$103^{+}7$	0:43:50	114	109	0.62
	profile	$1303^{+}9$	7:09:51	$876^{+}169$	95:45:58	962	959	21.57	$367^{+}2$	0:23:56	$254^{+}5$	3:11:43	266	260	1.87
Other	MS88	249	0:01:27	$155^{+}2$	0:52:17	157	157	2.62	*already proper*						
	Worldcup86	559	0:05:43	$349^{+}26$	1:44:46	368	356	2.60	116	19.6	$113^{+}3$	0:31:30	116	116	0.44
	Worldcup02	$501^{+}1$	1:24:56	$385^{+}15$	7:19:38	405	399	6.44	167	0:05:26	$150^{+}1$	0:36:42	156	151	0.73
	SM96-3L	$108^{+}4$	2:30:57	$43^{+}8$	6:52:03	68	54	7.66	*not applicable*						
	SM96-full	$1212^{+}13$	8:47:37	$658^{+}36$	137:21:07	809	758	39.78	655	0:09:37	$408^{+}9$	2:16:06	435	421	3.54

Table 15.3: Different approaches for proper and non-proper (MLVO) with narrow and wide alignment scheme. The time is suitably given either in seconds or as hh:mm:ss. $v_* = X^{+}Y$ gives the final lower bound X and upperbound $X + Y$ on the verticality. Due to its complexity, we stopped the bundle method after 250 function evaluations for the proper *profile* instance with wide alignment scheme. The heuristic uses a random initial order and both local optimization schemes (starting with 2-opt) alternatingly. We give the best results after 50 and 500 independent runs; the time is specified as the total for 500 independent runs.

15.5 A Comparison with Multi-level Crossing Minimization

As described in Section 9.1 the standard graph drawing scheme would be to optimize the node orderings w.r.t. the minimum crossing number. To investigate the (MLVO) and (MLCM) optimization strategies, we give a comparison between our (MLVO) SDP, and the results of the currently strongest SDP and a state-of-the-art ILP approach for (MLCM), extracted from Section 14.4. Notice that for (MLCM), the ILP can benefit from the relative sparsity of the linearized variables, and is hence better suited for small and sparse instances than the SDP approach. We already discussed (and verified experimentally above) that this is not the case for (MLVO).

We compare the (MLCM) approaches to their closest relative in the (MLVO) setting: Proper (MLVO) with narrow alignment scheme. Table 15.4 gives an overview. We observe that (MLVO) is harder than (MLCM) from the SDP point of view: in general, (MLVO) requires more computing time and cannot close the optimality gap as often. We are confident that the inclusion of additional Lovász-Schrijver-cuts (see Section 8.3) and the application of the strategies to strengthen semidefinite relaxations from Sections 7.2 and 7.3 will help to further reduce these gaps.

Being able to optimize both (MLCM) and (MLVO) using an SDP on common variables also gives rise to the idea of combining them using an objective function where the cost functions for (MLCM) and (MLVO) are balanced via coefficients $c_{\mathrm{z}}, c_{\mathrm{v}}$, respectively. We should be careful when choosing these coefficients to still allow some kind of rounding of lower bounds. To demonstrate the applicability of this approach (not arguing over the visual merits of certain blending coefficients), we choose integrals $c_{\mathrm{z}} = 10$ and $c_{\mathrm{v}} = 1$ (such that $c_{\mathrm{z}}/c_{\mathrm{v}} \approx \sum v_* / \sum z_*$). We denote this combined problem by (MLCVO). Table 15.4 shows that (MLCVO) is in general harder than (MLCM) but easier than (MLVO). The resulting solutions seem to yield quite convincing compromises between both objectives.

Finally we apply the SDP to (MLVO) after (MLCM), i.e., proper (MLVO) with wide alignment scheme and fixed crossing minimal orderings for the non-PDs (for details see Section 9.9 – additionally to the adjustments of the SDP described there, we also have to adapt the SDP heuristic by fixing the ordering of the "real" nodes and LEDs before hyperplane rounding and then only allowing to flip signs of variables involving PDs). Comparing the results for (MLVO) after (MLCM) given in Table 15.5 with the ones for proper (MLVO) with wide alignment scheme from Table 15.3 shows that their optimal solutions are closely related but always different. (MLVO) after (MLCM) yields substantially smaller gaps and faster running times than the corresponding proper (MLVO). Thus fixing the relative positions of several variables makes the underlying optimization problem considerably easier.

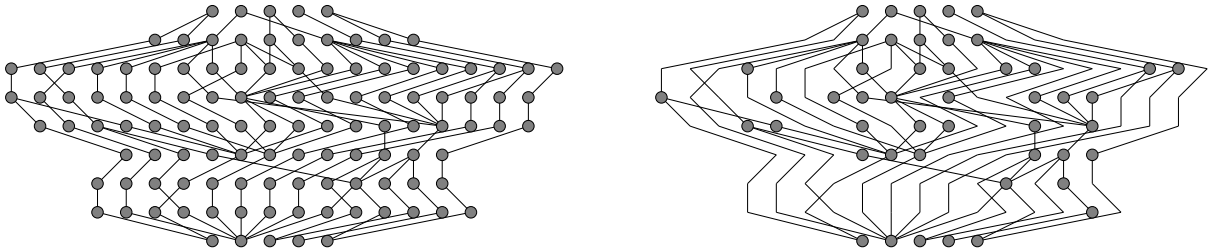
15.6 Visual Results for Different Drawing Strategies

We conclude by showcasing the visual results and relative benefits of the various problem solutions in Figures 15.1 and 15.2. Additionally in Figure 15.3 we visually compare (MLVO) to (MLCM) within the proper drawing style with wide alignment scheme, indicating the potential merit of verticality optimization over focusing on the crossing number.

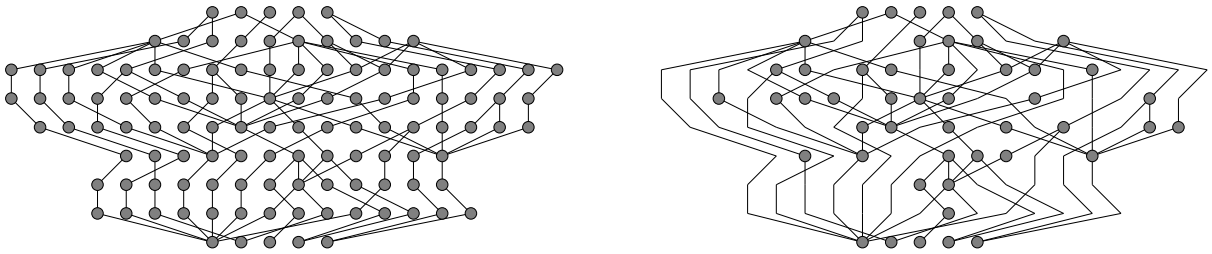
		(MLCM)					(MLVO)			(MLCVO)			
	Instance	z_*	v	d_C	time	ILP	z	v_*	time	b_*	z	v	time
Polytopes	Tetrahedron	22	48	0.245	0.08	0.12	24	48	2.27	268	22	48	2.83
	Octahedron	80	264	0.077	10.66	2.62	81	$261^{+}1$	0:02:37	$1063^{+}1$	80	264	0:04:46
	Cube3	80	264	0.077	10.93	3.14	81	$261^{+}1$	0:01:34	$1063^{+}1$	80	264	0:05:52
	Dodecahedron	$393^{+}1$	3096	0.014	4:40:09	(132/427)	399	$3051^{+}27$	3:31:58	$6972^{+}48$	394	3080	2:49:21
	Icosahedron	$393^{+}1$	3148	0.014	4:37:25	(174/401)	395	$3046^{+}34$	4:51:06	$6968^{+}52$	394	3080	4:18:09
	Cube4	$1192^{+}3$	6594	0.017	7:10:19	(197/1334)	1247	$6336^{+}86$	7:57:46	$18416^{+}128$	1195	6594	9:02:52
	Soccer ball	1627^{+} 726	72648	0.002	91:34:16	(118/2630)	2681	52392^{+} 21759	141:56:52	82898^{+} 15900	2538	73418	123:23:29
Graphviz	switch	20	56	0.024	1.92	0.66	23	$53^{+}1$	0:05:54	1064	20	56	0:01:34
	unix	0	141	0.011	0.25	0.01	7	111	0:04:27	126	0	126	0:03:19
	world	46	847	0.003	1:13:49	16.08	83	$620^{+}41$	6:33:10	$1221^{+}13$	50	734	8:39:12
	profile	37	2767	0.003	0:53:34	2.84	75	$1303^{+}9$	7:09:51	$1835^{+}2$	45	1387	7:19:20
Other	MS88	91	300	0.053	2.79	5.02	109	249	0:01:27	1209	91	299	22.65
	Worldcup86	49	762	0.020	25.3	1.12	72	559	0:05:43	1131	52	611	0:05:39
	Worldcup02	45	790	0.009	0:01:33	6.66	63	$501^{+}1$	1:24:56	1051	51	541	1:39:45
	SM96-3L	13	388	0.004	0:01:26	0.18	16	$108^{+}4$	2:30:57	246	13	116	3:20:54
	SM96-full	162	1491	0.006	0:53:29	3:03:05	222	$1212^{+}13$	8:47:37	3010	163	1380	7:15:34

The columns z_*, v_*, and b_* give the optimal solutions (or final bounds) of `(MLCM)`, `(MLVO)`, and `(MLCVO)`, respectively.
The columns z and v give the crossing number and non-verticality of the found solution.
The column d_C gives the density of the `(MLCM)` cost matrix.
The column "ILP" gives the time required by the ILP if successful, or the final bounds if it ran out of memory after 51h, 16h, 18h, and 52h, respectively.
Due to its complexity, we stopped the bundle method after 50 function evaluations for soccer ball.

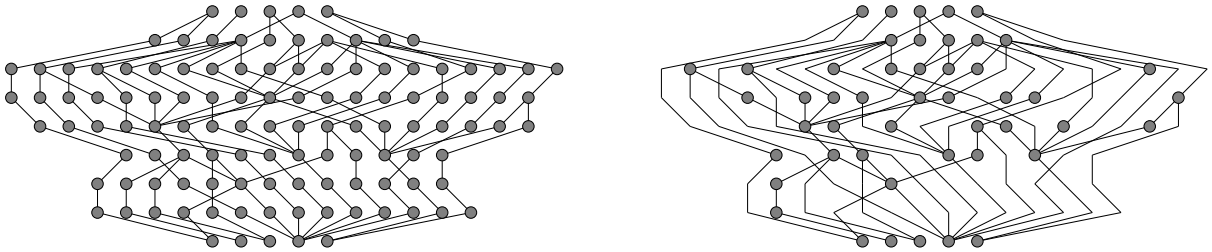
Table 15.4: Comparing `(MLVO)` with `(MLCM)`, and combining them to obtain `(MLCVO)`



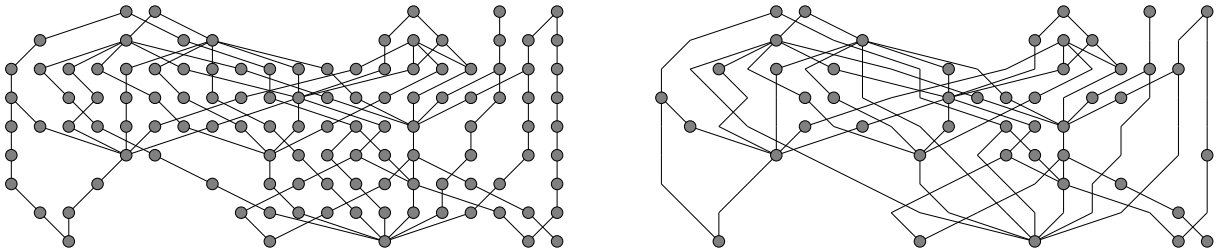
(a) `(MLCM)`, with and without explicitly drawn LEDs



(b) Proper `(MLVO)`, narrow alignment scheme, with and without explicitly drawn LEDs



(c) Combined `(MLCVO)`, with and without explicitly drawn LEDs

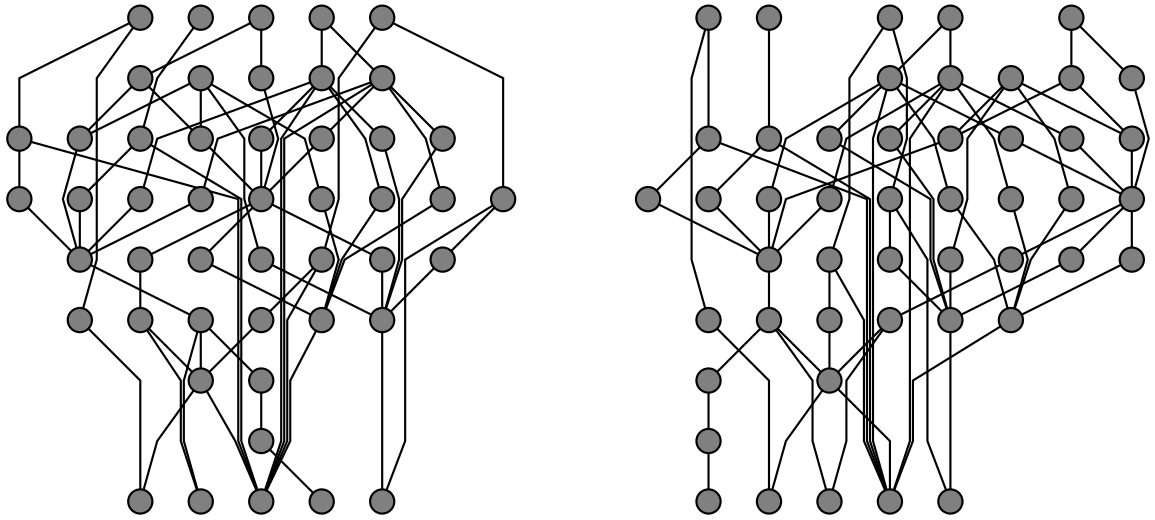


(d) Proper `(MLVO)`, wide alignment scheme, with and without explicitly drawn LEDs

Figure 15.1: Example drawings of different (near-)optimal solutions for different problem paradigms on proper level graphs. Instance: *world*.

		(MLVO) after (MLCM)	
	Instance	$v_*(z_*)$	time
Polyt.	Octahedron	$243^{+}1$	0:11:49
	Dodecahedron	$1851^{+}15$	7:57:00
	Cube4	$5414^{+}32$	36:23:34
Gr.viz	unix	$86^{+}5$	1:01:46
	world	$459^{+}29$	17:46:48
	profile	$1363^{+}38$	178:54:16
Other	MS88	$154^{+}4$	0:48:26
	Worldcup86	$506^{+}5$	2:01:36
	Worldcup02	$520^{+}9$	3:17:16
	SM96-full	$711^{+}67$	61:30:45

Table 15.5: Using the SDP as an exact quadratic compactor for Sugiyama's third stage by solving (MLVO) after (MLCM). $v_*(z_*)$ gives bounds on the optimal non-verticality with assured minimal crossing number. The times are given in hh:mm:ss.



(a) Non-proper MLVO, narrow alignment scheme

(b) Non-proper MLVO, wide alignment scheme

Figure 15.2: Example drawings of (near-)optimal solutions for Non-Proper (MLVO). Instance: *world*.

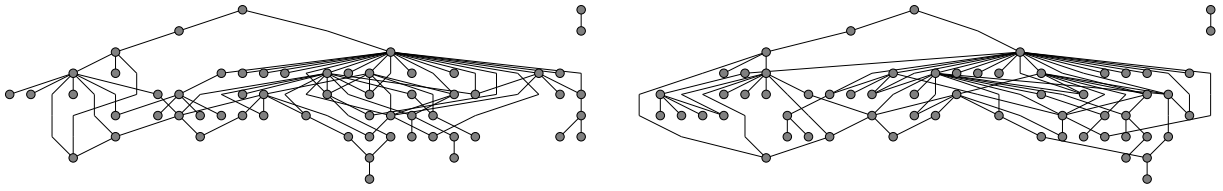


Figure 15.3: Instance *profile* drawn as a proper level graph (SDP upper bounds, not necessarily optimal). The left drawing optimizes the verticality (MLVO), whereas the right drawing optimizes the number of crossings (in fact, the latter solves (MLVO) after (MLCM), i.e., it optimizes verticality within a crossing optimal solution).

Chapter 16

Conclusion and Outlook

In this thesis we have presented a systematic investigation and comparison of SDP based relaxations to various ordering problems with linear and quadratic cost structure. We demonstrated that semidefinite relaxations provide theoretically and practically substantially tighter bounds than the corresponding linear programming relaxations. Although computing the former relaxations is more time consuming, our experiments demonstrate that using semidefinite relaxations pays off in practice. As the SDP approach is relatively independent of the density structure of the underlying cost matrix, it is not surprising that it clearly dominates exact ILP and IQP approaches based on ordering, betweenness or distances variables on dense problems. Yet, we also showed that our SDP approach is beneficial on various sparse benchmark sets. Always when the LP bounds are too weak for efficient pruning in Branch-and-Bound enumeration, it pays off to work with the theoretically stronger semidefinite relaxations. In the course of developing the SDP algorithm, we also obtained a seemingly strong SDP-based rounding heuristic that provided the best known feasible layouts for various problem types.

Furthermore we demonstrated that our SDP algorithm outperforms some other semidefinite methods proposed for special ordering problems. This is due to the interaction of the following three advancements

- the usage of a stronger semidefinite relaxation,
- the appropriate algorithmic approach for solving this relaxation,
- the usage of a strong SDP based heuristic.

While there exist quite diverse exact ILP approaches to the various ordering problems (due to the different structures of the underlying polytopes and the different cost functions), the semidefinite approach is uniformly applicable to all of these problems, as it works on the more general linear-quadratic ordering polytope. This generality distinguishes the semidefinite approach. We only have to adapt the cost function to compute all kinds of ordering problems with linear or quadratic cost structure.

Therefore it seems to be worthwhile to think about ways to further improve the presented approach. There are three (combinable) directions to enhance the presented SDP based relaxations of ordering problems. First we could include additional constraint classes to further tighten the relaxation (we already presented several ideas to do so and provided encouraging preliminary computational results). Secondly we could incorporate the SDP based bounds in a Branch-and-Bound framework and thirdly we could speed up the computations over the elliptope, which constitute the computational bottleneck of our algorithm, by using first-order methods instead of interior-point methods.

In this thesis, we also introduced a new drawing paradigm for layered graphs. We presented the concept of verticality as a novel optimization goal. Using our non-proper drawing scheme, we can vastly reduce the size of the according Multi-level Verticality Optimization Problem and consequently obtain (near-)optimal, (well-)readable drawings of graphs much too large for other approaches available. We also proposed several

heuristic and exact approaches to solve Multi-level Verticality Optimization, compared them theoretically and practically and designed a drawing algorithm to visually illustrate the (near-)optimal solutions.

We want to conclude by pointing out several research questions and plans that arose during the development of the material of this thesis. Regarding the theoretical part we plan to provide further results comparing the strength of SDP and ILP approaches, e.g. for the minimum Linear Arrangement Problem. We also want to investigate if some of the constraint classes that we deduced for tightening the basic semidefinite relaxation are facet defining for the linear-quadratic ordering polytope. Additionally we plan to conduct a polyhedral study of the multi-level quadratic ordering polytopes in small dimensions.

With respect to the practical part we plan to apply our SDP approach to further ordering problems mentioned in this thesis, like the Physical Mapping Problem with End Probes or Crossing Minimization in Tanglegrams. We want to apply the tightening strategies proposed in Chapter 7 to several ordering problems, studying their practical merits and computational costs. Additionally we think that it is worthwhile to think about combining different ordering problems or detecting new ones that now can be solved quite efficiently by our SDP approach, independent of the complexity of the quadratic cost structure. Two such problems are the Unidirectional Circular Facility Layout Problem [112] and the Multi-row Facility Layout Problem [113, 114].

Some time ago we started working on another interesting combinatorial optimization problem called Target Visitation Problem that is a combination of the Linear Ordering Problem and the Traveling Salesman Problem. The Target Visitation Problem can not be formulated as quadratic ordering problem and thus we had to design and investigate a different SDP model based on so-called position variables. We conducted extensive polyhedral studies of the underlying polytope and proposed exact approaches based on linear and semidefinite relaxations [109]. We also compared both approaches from a theoretical and practical point of view. We are interested in finding other such problems that contain ordering problems as a special case.

We also came across a very challenging problem in graph drawing. We want to design an exact semidefinite approach for minimizing the number of crossings (or also maximizing the verticality) in extended level graphs, i.e. level graphs with both inter-level and intra-level edges. Drawing extended level graphs, e.g. for displaying centrality values of actors, was addressed as one of the open problems in social network visualization [28]. Bachmaier et al. [16] proposed several heuristics for this problem, but there does not exist any exact approach yet. In extended level graphs intra-level edges are drawn as semicircles with different radii on one side of the level lines. Thus minimizing intra-level edge crossings is equivalent to crossing minimization on the circle, where the vertices lie on a circle and edges are drawn as straight lines within the circle. We can formulate this problem as an IQP in betweenness variables. Hence matrix liftings and semidefinte relaxations seem to be the appropriate tools to design an efficient exact method for crossings minimization in extended level graphs.

Bibliography

[1] H. Achatz, P. Kleinschmidt, and J. Lambsdorff. Der corruption perceptions index und das linear ordering problem. *ORNews*, 26:10–12, 2006.

[2] D. Adolphson. Singlemachine job sequencing with precedence constraints. *SIAM Journal on Computing*, 6:40–54, 1977.

[3] A. Ahmed, X. Fu, S.-H. Hong, Q. H. Nguyen, and K. Xu. Visual analysis of history of world cup: A dynamic network with dynamic hierarchy and geographic clustering. In *Proceedings of the Visual Information Communications International [VINCI'2009]*, pages 25–39. Springer, 2010.

[4] F. Alizadeh, J.-P. Haeberly, and M. Overton. Complementarity and nondegeneracy in semidefinite programming. *Mathematical Programming*, 77:111–128, 1997. 10.1007/BF02614432.

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

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

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

[8] A. R. S. Amaral and A. N. Letchford. A polyhedral approach to the single row facility layout problem. Technical report, 2011. preprint available from `www.optimization-online.org/DB_FILE/2008/03/1931.pdf`.

[9] M. Andersen, J. Dahl, and L. Vandenberghe. Implementation of nonsymmetric interior-point methods for linear optimization over sparse matrix cones. *Mathematical Programming Computation*, 2:167–201, 2010.

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

[11] M. F. Anjos and J. B. Lasserre, editors. *Handbook on Semidefinite, Conic and Polynomial Optimization Theory, Algorithms, Software and Applications*. International Series in Operations Research & Management Science. Springer-Verlag, New York, 2011.

[12] M. F. Anjos and F. Liers. Global approaches for facility layout and VLSI floorplanning. In M. F. Anjos and J. B. Lasserre, editors, *Handbook on Semidefinite, Conic and Polynomial Optimization: Theory, Algorithms, Software and Applications*, International Series in Operations Research and Management Science. Springer, New York, 2011.

[13] M. F. Anjos and A. 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.

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

[15] C. Bachmaier, F. J. Brandenburg, W. Brunner, and F. Hübner. A global k-level crossing reduction algorithm. In *WALCOM 2010*, LNCS 5942, pages 70–81, 2010.

[16] C. Bachmaier, H. Buchner, M. Forster, and S.-H. Hong. Crossing minimization in extended level drawings of graphs. *Discrete Applied Mathematics*, 158:159–179, 2010.

[17] V. Balakrishnan and F. Wang. Sdp in systems and control theory. In H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors, *Handbook of semidefinite programming*, volume 27 of *International Series in Operations Research & Management Science*, pages 421–442. Springer US, 2000.

[18] F. Barahona and A. Mahjoub. On the cut polytope. *Mathematical Programming*, 36:157–173, 1986.

[19] F. Baumann, C. Buchheim, and F. Liers. Exact bipartite crossing minimization under tree constraints. In P. Festa, editor, *Experimental Algorithms*, volume 6049 of *Lecture Notes in Computer Science*, pages 118–128. Springer Berlin / Heidelberg, 2010.

[20] R. Bellman and K. Fan. On systems of linear inequalities in hermitian matrix variables. In V. L. Klee, editor, *Convexity*, Proceedings of Symposia in Pure Mathematics, pages 1–11. American Mathematical Society, 1963.

[21] A. Belloni and C. Sagastizabal. Dynamic bundle methods. *Mathematical Programming, Series A*, 120(2):289–311, 2009.

[22] A. Ben-Tal and A. Nemirovski. Structural design. In H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors, *Handbook of Semidefinite Programming*, volume 27 of *International Series in Operations Research & Management Science*, pages 443–467. Springer US, 2000.

[23] K. Boenchendorf. *Reihenfolgenprobleme/Mean-flow-time sequencing.* Mathematical Systems in Economics 74. Verlagsgruppe AthenÄaum, Hain, Scriptor, 1982.

[24] B. Borchers. *CSDP 5.0 User's Guide.* Available at `http://www.optimization-online.org/DB_HTML/2002/10/551.html`, 2005.

[25] C. F. Bornstein and S. Vempala. Flow metrics. *Theoretical Computer Science*, 321:13–24, 2004.

[26] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, New York, NY, USA, 2004.

[27] S. E. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15 of *Studies in Applied Mathematics.* SIAM, Philadelphia, USA, 1994.

[28] U. Brandes, T. Raab, and D. Wagner. Exploratory network visualization: Simultaneous display of actor status and connections. *Journal of Social Structure*, 2:1–28, 2001.

[29] T. A. Brown. *Gene Cloning and DNA Analysis: An Introduction.* Wiley-Blackwell, Malden, USA, 5 edition, 2010.

[30] C. Buchheim, F. Liers, and M. Oswald. Speeding up ip-based algorithms for constrained quadratic 0–1 optimization. *Mathematical Programming*, 124:513–535, 2010.

[31] C. Buchheim, A. Wiegele, and L. Zheng. Exact Algorithms for the Quadratic Linear Ordering Problem. *INFORMS Journal on Computing*, 22(1):168–177, 2010.

[32] S. Burer and R. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming (B)*, 95:329–357, 2003.

[33] K. c. Toh, M. j. Todd, and R. H. Tütüncü. Sdpt3 - a matlab software package for semidefinite programming. *Optimization Methods and Software*, 11:545–581, 1999.

[34] A. Caprara, M. Jung, M. Oswald, G. Reinelt, and E. Traversi. A betweenness approach for solving the linear arrangement problem. Technical report, DEIS, Università di Bologna, 2009.

[35] A. Caprara, A. N. Letchford, and J.-J. Salazar-González. Decorous lower bounds for minimum linear arrangement. *INFORMS Journal on Computing*, 23(1):26–40, 2011.

[36] A. Caprara and J.-J. Salazar-González. Laying out sparse graphs with provably minimum bandwidth. *INFORMS Journal on Computing*, 17:356–373, 2005.

[37] M. Charikar, M. T. Hajiaghayi, H. Karloff, and S. Rao. l_2^2 spreading metrics for vertex ordering problems. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, SODA '06, pages 1018–1027, New York, USA, 2006.

[38] P. P.-S. Chen. The entity-relationship model–toward a unified view of data. *ACM Transactions on Database Systems*, 1:9–36, 1976.

[39] H. Chenery and T. Watanabe. International comparisons of the structure of production. *Econometrica*, 26:487–521, 1958.

[40] M. Chimani, C. Gutwenger, P. Mutzel, and H.-M. Wong. Layer-free upward crossing minimization. *ACM Journal of Experimental Algorithmics*, 15, 2010.

[41] M. Chimani, C. Gutwenger, P. Mutzel, and H.-M. Wong. Upward planarization layout. In *Proceedings of the Symposium on Graph Drawing [GD'09]*, volume 5849 of *LNCS*, pages 94–106. Springer, 2010.

[42] M. Chimani and P. Hungerländer. Exact approaches to multi-level vertical orderings. *INFORMS Journal on Computing*, 2012. accepted, preprint available at `www.ae.uni-jena.de/Research\_Pubs/MLVO.html`.

[43] M. Chimani and P. Hungerländer. Multi-level verticality optimization: Concept, strategies, and drawing scheme. *Journal of Graph Algorithms and Applications*, 2012. accepted, preprint available at `www.ae.uni-jena.de/Research\_Pubs/MLVO.html`.

[44] M. Chimani, P. Hungerländer, M. Jünger, and P. Mutzel. An SDP approach to multi-level crossing minimization. In *Proceedings of Algorithm Engineering & Experiments [ALENEX'2011]*, 2011.

[45] M. Chimani, P. Hungerländer, M. Jünger, and P. Mutzel. An SDP approach to multi-level crossing minimization. *Journal of Experimental Algorithmics*, 2011. accepted (extended version of the proceedings paper with the same title).

[46] T. Christof. *Low-dimensional 0/1-polytopes and branch-and-cut in combinatorial optimization*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, Aachen, 1997.

[47] T. Christof, M. Jünger, J. Kececioglu, P. Mutzel, and G. Reinelt. A branch-and-cut approach to physical mapping with end-probes. In *Proceedings of the first annual international conference on Computational molecular biology*, RECOMB '97, pages 84–92, New York, USA, 1997. ACM.

[48] T. Christof, M. Oswald, and G. Reinelt. Consecutive ones and a betweenness problem in computational biology. In *Proceedings of the 6th Conference on Integer Programming and Combinatorial Optimization (IPCO 1998)*, Lecture Notes in Computer Science 1412, pages 213–228. Springer, 1998.

[49] D. Datta, A. R. S. Amaral, and J. R. Figueira. Single row facility layout problem using a permutation-based genetic algorithm. *European Journal of Operational Research*, 213(2):388–394, 2011.

[50] T. Davi and F. Jarre. High accuracy solution of large scale semidefinite programs. 2011. Preprint available at `www.optimization-online.org/DB_HTML/2011/04/2999.html`.

[51] T. Davi and F. Jarre. Solving large scale problems over the doubly nonnegative cone. 2011. Preprint available at `www.optimization-online.org/DB_HTML/2011/04/3000.html`.

[52] E. de Klerk. *Aspects of semidefinite programming: interior point algorithms and selected applications*. Kluwer Academic Publishers, 2002.

[53] E. de Klerk, J. Peng, C. Roos, and T. Terlaky. A scaled gauss-newton primal-dual search direction for semidefinite optimization. *SIAM Journal on Optimization*, 11:870–888, 2001.

[54] C. Delorme and S. Poljak. Laplacian eigenvalues and the maximum cut problem. *Mathematical Programming*, 62:557–574, 1993.

[55] C. Delorme and S. Poljak. The performance of an eigenvalue bound on the max-cut problem in some classes of graphs. *Discrete Mathematics*, 111(1-3):145 – 156, 1993.

[56] E. D. Demaine, S. P. Fekete, G. Rote, N. Schweer, D. Schymura, and M. Zelke. Integer point sets minimizing average pairwise l1 distance: What is the optimal shape of a town? *Computational Geometry*, 44(2):82 – 94, 2011. Special issue of selected papers from the 21st Annual Canadian Conference on Computational Geometry.

[57] M. Deza. On the hamming geometry of unitary cubes. *Doklady Akademii Nauk SSR*, 134:1037–1040, 1960. [English translation in: *Soviet Physics Doklady* 5 (1961) 940–943].

[58] M. Deza and M. Laurent. Facets for the cut cone I. *Mathematical Programming*, 56:121–160, 1992.

[59] M. Deza and M. Laurent. Facets for the cut cone II: clique-web inequalities. *Mathematical Programming*, 56:161–188, 1992.

[60] M. Deza and M. Laurent. Applications of cut polyhedra–I. *Journal of Computational and Applied Mathematics*, 55:191–216, 1994.

[61] M. Deza and M. Laurent. Applications of cut polyhedra–II. *Journal of Computational and Applied Mathematics*, 55:217–247, 1994.

[62] M. Deza and M. Laurent. *Geometry of Cuts and Metrics*, volume 15 of *Algorithms and Combinatorics*. Springer Verlag, Berlin, 1997.

[63] G. Di Battista, A. Garg, G. Liotta, A. Parise, R. Tamassia, E. Tassinari, F. Vargiu, and L. Vismara. Drawing directed acyclic graphs: An experimental study. *International Journal of Computational Geometry and Applications*, 10(6):623–648, 2000.

[64] G. Di Battista, A. Garg, G. Liotta, R. Tamassia, E. Tassinari, and F. Vargiu. An experimental comparison of four graph drawing algorithms. *Computational Geometry: Theory and Applications*, 7(5-6):303–325, 1997.

[65] J. Díaz, J. Petit, and M. Serna. A survey of graph layout problems. *ACM Computing Surveys*, 34:313–356, September 2002.

[66] I. S. Duff, R. G. Grimes, and J. G. Lewis. Users' guide for the harwell-boeing sparse matrix collection. Technical report, CERFACS, Toulouse, France, 1992.

[67] R. J. Duffin. Infinite programs. In *Linear inequalities and related systems*, number 38 in Annals of Mathematics Studies, pages 157–170. Princeton University Press, 1956.

[68] V. Dujmović, H. Fernau, and M. Kaufmann. Fixed parameter algorithms for one-sided crossing minimization revisited. *J. of Discrete Algorithms*, 6(2):313–323, 2008.

[69] P. Eades and N. C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, 1994.

[70] M. Eiglsperger, M. Kaufmann, and F. Eppinger. An approach for mixed upward planarization. *Journal of Graph Algorithms and Applications*, 7(2):203–220, 2003.

[71] U. Feige and J. Kilian. Zero knowledge and the chromatic number. *Journal of Computer and System Sciences*, 57:187–199, 1998.

[72] U. Feige and J. R. Lee. An improved approximation ratio for the minimum linear arrangement problem. *Information Processing Letters*, 101:26–29, 2007.

[73] I. Fischer, G. Gruber, F. Rendl, and R. Sotirov. Computational experience with a bundle method for semidefinite cutten plane relaxations of max-cut and equipartition. *Mathematical Programming*, 105:451–469, 2006.

[74] P. C. Fishburn. Induced binary probabilities and the linear ordering polytope: a status report. *Mathematical Social Sciences*, 23(1):67–80, 1992.

[75] A. Frangioni. Generalized bundle methods. *SIAM Journal on Optimization*, 13:117–156, 2002.

[76] K. Fujisawa, M. Kojima, and K. Nakata. Exploiting sparsity in primal-dual interior-point methods for semidefinite programming. *Mathematical Programming*, 79:235–253, 1997.

[77] C. P. Gane and T. Sarson. *Structured Systems Analysis: Tools and Techniques*. Prentice Hall Professional Technical Reference, 1st edition, 1979.

[78] G. Gange, P. J. Stuckey, and K. Marriott. Optimal k-level planarization and crossing minimization. In *Proceedings of the Symposium on Graph Drawing [GD'10]*, LNCS. Springer, 2010. to appear.

[79] E. R. Gansner, E. Koutsofios, S. C. North, and K. P. Vo. A technique for drawing directed graphs. *IEEE Trans. Softw. Eng.*, 19(3):214–230, 1993.

[80] M. R. Garey and D. S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*, 4(4):397–411, 1975.

[81] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[82] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified np-complete problems. In *STOC '74: Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 47–63, New York, 1974.

[83] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified np-complete graph problems. *Theoretical Computer Science*, 1(3):237 – 267, 1976.

[84] F. Glover, T. Klastorin, and D. Klingman. Optimal weighted ancestry relationships. *Management Science*, 20:1190–1193, 1974.

[85] M. Goemans. Semidefinite programming in combinatorial optimization. *Mathematical Programming*, 79:143–161, 1997.

[86] M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

[87] A. Gomes de Alvarenga, F. J. Negreiros-Gomes, and M. Mestria. Metaheuristic methods for a class of the facility layout problem. *Journal of Intelligent Manufacturing*, 11:421–430, 2000.

[88] Graphviz gallery, `http://www.graphviz.org/Gallery.php`, Oktober 2010.

[89] M. Grötschel, M. Jünger, and G. Reinelt. A Cutting Plane Algorithm for the Linear Ordering Problem. *Operations Reasearch*, 32(6):1195–1220, 1984.

[90] K. M. Hall. An r-dimensional quadratic placement algorithm. *Management Science*, 17(3):219–229, 1970.

[91] P. Hammer. Some network flow problems solved with pseudo-Boolean programming. *Operations Research*, 13:388–399, 1965.

[92] L. H. Harper. Optimal assignments of numbers to vertices. *SIAM Journal on Applied Mathematics*, 12:131–135, 1964.

[93] L. H. Harper. Optimal numberings and isoperimetric problems on graphs. *Journal of Combinatorial Theory*, 1:385–393, 1966.

[94] J. Håstad. Clique is hard to approximate within $n^{1\text{-epsilon}}$. *Electronic Colloquium on Computational Complexity (ECCC)*, 4(38), 1997.

[95] J. Håstad. Some optimal inapproximability results. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, STOC '97, pages 1–10, New York, NY, USA, 1997. ACM.

[96] P. Healy and A. Kuusik. The vertex-exchange graph: A new concept for multilevel crossing minimisation. In *Proceedings of the Symposium on Graph Drawing [GD'99]*, pages 205–216. Springer, 1999.

[97] P. Healy and A. Kuusik. The vertex-exchange graph and its use in multi-level graph layout. Technical Report UL-CSIS-99-1, Department of Computer Science and Information Systems, University of Limerick, Ireland, 1999.

[98] C. Helmberg. Fixing variables in semidefinite relaxations. *SIAM Journal on Matrix Analysis and Applications*, 21(3):952–969, 2000.

[99] C. Helmberg. Semidefinite programming. *European Journal of Operational Research*, 137:461–482, 2002.

[100] C. Helmberg. Numerical validation of SBmethod. *Mathematical Programming*, 95:381–406, 2003.

[101] C. Helmberg, K. Kiwiel, and F. Rendl. Incorporating inequality constraints in the spectral bundle method. In E. B. R.E. Bixby and R. Rios-Mercado, editors, *Integer Programming and combinatorial optimization*, pages 423–435. Springer Lecture Notes 1412, 1998.

[102] C. Helmberg, B. Mohar, S. Poljak, and F. Rendl. A spectral approach to bandwidth and separator problems in graphs. *Linear and Multilinear Algebra*, 39:73–90, 1995.

[103] C. Helmberg and F. Oustry. Bundle methods to minimize the maximum eigenvalue function. In R. S. H. Wolkowicz and L. Vandenberghe, editors, *Handbook of semidefinite programming: theory, algorithms and applications*, volume 27 of *International Series in Operations Research & Management Science*, pages 307–337. Springer US, 2000.

[104] C. Helmberg and F. Rendl. A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization*, 10:673–696, 2000.

[105] C. Helmberg, F. Rendl, R. Vanderbei, and H. Wolkowicz. An interior-point method for semidefinite programming. *SIAM Journal on Optimization*, 6:342–361, 1996.

[106] S. S. Heragu and A. S. Alfa. Experimental analysis of simulated annealing based algorithms for the layout problem. *European Journal of Operational Research*, 57(2):190 – 202, 1992.

[107] S. S. Heragu and A. Kusiak. Machine Layout Problem in Flexible Manufacturing Systems. *Operations Research*, 36(2):258–268, 1988.

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

[109] A. Hildenbrandt, P. Hungerländer, and G. Reinelt. Exact approaches to the target visitation problem. Technical report, 2011.

[110] J.-B. Hiriart-Urruty and C. Lemarechal. *Convex Analysis and minimization algorithms (vol. 1 and 2)*. Springer, 1993.

[111] H.-W. Holub and H. Schnabl. *Input-Output-Rechnung: Input-Output-Tabellen*. Oldenbourg Wissenschaftsverlag, 1982.

[112] P. Hungerländer. A semidefinite optimization approach to unidirectional circular facility layout. Technical report, 2012.

[113] P. Hungerländer and M. Anjos. A semidefinite optimization approach to space-free multi-row facility layout. Cahier du GERAD G-2012-03, GERAD, Montreal, QC, Canada, 2012.

[114] P. Hungerländer and M. F. Anjos. Semidefinite optimization approaches to multi-row facility layout. Technical report, submitted, 2012.

[115] P. Hungerländer and F. Rendl. Semidefinite relaxations of ordering problems. *Mathematical Programming B*, 2011. accepted, preprint available at `www.optimization-online.org/DB\_HTML/2010/08/2696.html`.

[116] P. Hungerländer and F. Rendl. A computational study and survey of methods for the single-row facility layout problem. *Computational Optimization and Applications*, 2012. accepted, preprint available at `http://www.optimization-online.org/DB\_HTML/2011/05/3029.html`.

[117] *IBM ILOG CPLEX V12.1 User's Manual for CPLEX*. Available at `ftp://public.dhe.ibm.com/software/websphere/ilog/docs/optimization/cplex/ps_usrmancplex.pdf`, 2009.

[118] F. Jarre and F. Rendl. An augmented primal-dual method for linear conic problems. *SIAM Journal on Optimization*, 20:808–823, 2009.

[119] M. Jünger, E. K. Lee, P. Mutzel, and T. Odenthal. A polyhedral approach to the multi-layer crossing minimization problem. In *GD '97: Proceedings of the 5th International Symposium on Graph Drawing*, pages 13–24, London, UK, 1997. Springer-Verlag.

[120] M. Jünger and P. Mutzel. 2-layer straightline crossing minimization: performance of exact and heuristic algorithms. *Journal of Graph Algorithms and Applications*, 1:1–25, 1997.

[121] M. Juvan and B. Mohar. Optimal linear labelings and eigenvalues of graphs. *Discrete Applied Mathematics*, 36(2):153–168, 1992.

[122] R. Kaas. A branch and bound algorithm for the acyclic subgraph problem. *European Journal of Operational Research*, 8(4):355 – 362, 1981.

[123] H. Karloff. How good is the goemans-williamson max cut algorithm? In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, STOC '96, pages 427–434, New York, NY, USA, 1996.

[124] R. Karp. Reducibility among combinatorial problems. In R. Miller and J.W.Thather, editors, *Complexity of Computer Computation*, pages 85–103. Plenum Press, 1972.

[125] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[126] R. M. Karp. Mapping the genome: some combinatorial problems arising in molecular biology. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '93, pages 278–285, New York, USA, 1993. ACM.

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

[128] J. Kelly. Hypermetric spaces. In L. Kelly, editor, *The Geometry of Metric and Linear Spaces*, Lecture Notes in Mathematics, pages 17–31. Springer Berlin / Heidelberg, 1975.

[129] J. G. Kemeny. Mathematics without numbers. *Daedalus*, 88(4):577–591, 1959.

[130] D. E. Knuth. *The Stanford GraphBase: a platform for combinatorial computing*. ACM, New York, NY, USA, 1993.

[131] M. Kojima, S. Shindoh, and S. Hara. Interior-point methods for the monotone semidefinite linear complementarity problem in symmetric matrices. *SIAM Journal on Optimization*, 7(1):86–125, 1997.

[132] Y. Koren and D. Harel. A multi-scale algorithm for the linear arrangement problem. In *Revised Papers from the 28th International Workshop on Graph-Theoretic Concepts in Computer Science*, WG '02, pages 296–309, London, UK, 2002. Springer-Verlag.

[133] S. Kruk, M. Muramatsu, F. Rendl, R. J. Vanderbei, and H. Wolkowicz. The gauss-newton direction in semidefinite programming. *Optimization Methods and Software*, 15(1):1–28, 2001.

[134] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2:83–97, 1955.

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

[136] M. Laurent and S. Poljak. On a positive semidefinite relaxation of the cut polytope. *Linear Algebra and its Applications*, 223/224:439–461, 1995.

[137] M. Laurent and S. Poljak. On the facial structure of the set of correlation matrices. *SIAM Journal on Matrix Analysis and Applications*, 17:530–547, 1996.

[138] M. Laurent, S. Poljak, and F. Rendl. Connections between semidefinite relaxations of the max-cut and stable set problems. *Mathematical Programming*, 77:225–246, 1997.

[139] M. Laurent and F. Rendl. Semidefinite programming and integer programming. In K. Aardal, G. Nemhauser, and R. Weismantel, editors, *Discrete Optimization*, pages 393–514. Elsevier, 2005.

[140] C. Lemarechal. An extension of davidon methods to nondifferentiable problems. *Mathematical Programming Study*, 3:95–109, 1975.

[141] C. Lemarechal. Nonsmooth optimization and descent methods. Technical report, International Institute for Applied Systems Analysis, 1978.

[142] C. Lemarechal, A. Nemirovskii, and Y. Nesterov. New variants of bundle methods. *Mathematical Programming*, 69:111–147, 1995.

[143] W. Leontief. Quantitative input-output relations in the economic system of the united states. *The Review of Economics and Statistics*, 18, 1936.

[144] W. Leontief. *Input-output economics*. Oxford University Press, 1966.

[145] L. Li and K.-C. Toh. An inexact interior point method for l1-regularized sparse covariance selection. *Mathematical Programming Computation*, 2:291–315, 2010.

[146] R. Lougee-Heimer. The common optimization interface for operations research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, 47:57–66, 2003.

[147] L. Lovász. On the shannon capacity of a graph. *IEEE Transactions on Information Theory*, 25:1–7, 1979.

[148] L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization*, 1:166–190, 1991.

[149] R. F. Love and J. Y. Wong. On solving a one-dimensional space allocation problem with integer programming. *INFOR*, 14:139–143, 1967.

[150] J. Malick, J. Povh, F. Rendl, and A. Wiegele. Regularization methods for semidefinite programming. *SIAM Journal on Optimization*, 20:336–356, 2009.

[151] R. Martí and M. Laguna. Heuristics and meta-heuristics for 2-layer straight line crossing minimization. *Discrete Applied Mathematics*, 127(3):665–678, 2003.

[152] R. Marti and G. Reinelt. *The Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization*. Applied Mathematical Sciences. Springer, 2011.

[153] R. Martí, G. Reinelt, and A. Duarte. A benchmark library and a comparison of heuristic methods for the linear ordering problem. *Computational Optimization and Applications*, pages 1–21, 2011.

[154] S. Masuda, K. Nakajima, T. Kashiwabara, and T. Fujisawa. Crossing minimization in linear embeddings of graphs. *IEEE Trans. Comput.*, 39:124–127, 1990.

[155] M. May and K. Szkatula. On the bipartite crossing number. *Control and Cybernetics*, 72:85–97, 1988.

[156] J. E. Mitchell and B. Borchers. Solving linear ordering problems with a combined interior point/simplex cutting plane algorithm. In T. T. H. Frenk, K. Roos and S. Zhang, editors, *High Performance Optimization*, pages 349–366. Kluwer Academic Publishers, 2000.

[157] G. Mitchison and R. Durbin. Optimal numberings of an N N array. *SIAM Journal on Algebraic and Discrete Methods*, 7:571–582, 1986.

[158] H. Mittelmann. Benchmarks for optimization software. `http://plato.asu.edu/bench.html`.

[159] R. Monteiro. Primal-dual path-following algorithms for semidefinite programming. *SIAM Journal on Optmization*, 7:663–678, 1997.

[160] J. Munkres. Algorithms for the assignment and transportation problems. *SIAM Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[161] P. Mutzel and R. Weiskircher. Two-layer planarization in graph drawing. In *Proceedings of the International Symposium on Algorithms and Computation [ISAAC'98]*, volume 1533 of *LNCS*, pages 69–78. Springer, 1998.

[162] L. Nachmanson, S. Pupyrev, and M. Kaufmann. Improving layered graph layouts with edge bundling. In *Proceedings of the Symposium on Graph Drawing [GD'2010]*, volume 6502 of *LNCS*. Springer, 2010.

[163] A. S. Nemirovski and M. J. Todd. Interior-point methods for optimization. *Acta Numerica*, 17:191–234, 2008.

[164] Y. Nesterov. Quality of semidefinite relaxation for nonconvex quadratic optimization. Discussion paper 9719, CORE, Catholic University of Louvain, Belgium, 1997.

[165] Y. Nesterov and A. Nemirovski. *Interior Point Polynomial Algorithms in Convex Programming*. SIAM Publications. SIAM, Philadelphia, USA, 1994.

[166] Y. E. Nesterov and M. J. Todd. Self-scaled barriers and interior-point methods for convex programming. *Mathematics of Operations Research*, 22:1–42, 1997.

[167] A. Newman and S. Vempala. Fences are futile: On relaxations for the linear ordering problem. In *Integer Programming and Combinatorial Optimization*, pages 333–347, 2001.

[168] M. Oswald. *Weighted Consecutive Ones Problems*. PhD thesis, Ruprecht-Karls-Universität, Heidelberg, 2003.

[169] M. Oswald. A combined approach for solving linear ordering and linear arrangement problems. Talk given at the University of Klagenfurt, Austria, 2009.

[170] R. D. M. Page. *Tangled Trees: Phylogeny, Cospeciation, and Coevolution*. University of Chicago Press, 2002.

[171] P. A. Parillo. *Structured Semidefinite Programs and Semi-algebraic Geometry Methods in Robustness and Optimization*. PhD thesis, California Institute of Technology, Pasadena, California, USA, 2000.

[172] J. Petit. Approximation heuristics and benchmarkings for the minla problem. In *Proceedings of Algorithms and Experiments*, pages 112–128, Università di Trento, 1998.

[173] J. Petit. *Layout Problems*. PhD thesis, Universitat Politècnica de Catalunya, 2001.

[174] J.-C. Picard and M. Queyranne. On the one-dimensional space allocation problem. *Operations Research*, 29(2):371–391, 1981.

[175] S. Poljak and F. Rendl. Nonpolyhedral relaxations of graph bisection problems. *SIAM Journal on Optimization*, 5:467–487, 1995.

[176] F. A. Potra and R. Sheng. A superlinearly convergent primal-dual infeasible-interior-point algorithm for semidefinite programming. *SIAM Journal on Optimization*, 8:1007–1028, 1998.

[177] J. Povh, F. Rendl, and A. Wiegele. A boundary point method to solve semidefinite programs. *Computing*, 78:277–286, 2006.

[178] H. Purchase. Which aesthetic has the greatest effect on human understanding? In *Proceedings of the Symposium on Graph Drawing [GD'97]*, volume 1353 of *LNCS*, pages 248–259. Springer, 1997.

[179] M. V. Ramana and P. M. Pardalos. Semidefinite programming. In T. Terlaky, editor, *Interior point methods of mathematical programming*, pages 369–398. Kluwer, Dordrecht, The Netherlands, 1996.

[180] S. Rao and A. W. Richa. New approximation techniques for some linear ordering problems. *SIAM Journal on Computing*, 34:388–404, 2005.

[181] R. Ravi, A. Agrawal, and P. Klein. Ordering problems approximated: Single-processor scheduling and interval graphs connection. In J. L. Albert, B. R. Artalejo, and B. Monien, editors, *18th International Colloquium on Automata, Languages and Programming*, volume 150 of *Lecture Notes in Computer Science*, pages 751–762. Springer-Verlag New York, 1991.

[182] F. Rendl. Difficult graphs for sdp relaxations of max-cut. Unpulished manuscript, 1997.

[183] F. Rendl. Semidefinite relaxations for integer programming. In M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, editors, *50 Years of Integer Programming 1958-2008*, pages 687–726. Springer Berlin Heidelberg, 2010.

[184] F. Rendl, G. Rinaldi, and A. Wiegele. Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 212:307–335, 2010.

[185] R. T. Rockafellar. *Convex analysis*. Princeton Mathematical Series, No. 28. Princeton University Press, 1970.

[186] D. Romero and A. Sánchez-Flores. Methods for the one-dimensional space allocation problem. *Computers & Operations Research*, 17(5):465 – 473, 1990.

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

[188] S. Sanjeevi and K. Kianfar. A polyhedral study of triplet formulation for single row facility layout problem. *Discrete Applied Mathematics*, 158:1861–1867, 2010.

[189] H. Schramm and J. Zowe. A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results. *SIAM J. Optimization*, 2:121–152, 1992.

[190] R. Schwarz. *A branch-and-cut algorithm with betweenness variables for the linear arrangement problems*. Diploma Thesis, Heidelberg, 2010.

[191] H. Seitz. *Contributions to the Minimum Linear Arrangement Problem*. PhD thesis, University of Heidelberg, Germany, 2010.

[192] H. D. Sherali and W. P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990.

[193] M. Shida, S. Shindoh, and M. Kojima. Existence of search directions in interior-point algorithms for the sdp and the monotone sdlcp. *SIAM Journal on Optimization*, 8(2):387–396, 1998.

[194] F. Shieh and C. McCreary. Directed graphs drawing by clan-based decomposition. In *Proceedings of the Symposium on Graph Drawing [GD'95]*, volume 1027 of *LNCS*, pages 472–482. Springer, 1996.

[195] D. M. Simmons. One-Dimensional Space Allocation: An Ordering Algorithm. *Operations Research*, 17:812–826, 1969.

[196] D. M. Simmons. A further note on one-dimensional space allocation. *Operations Research*, 19:249, 1971.

[197] C. D. Simone. The cut polytope and the Boolean quadric polytope. *Discrete Mathematics*, 79(1):71–75, 1990.

[198] P. Slater. Inconsistencies in a schedule of paired comparisons. *Biometrika*, 48(3-4):303–312, 1961.

[199] J. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11–12:625–653, 1999.

[200] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981.

[201] J. Suryanarayanan, B. Golden, and Q. Wang. A new heuristic for the linear placement problem. *Computers & Operations Research*, 18(3):255 – 262, 1991.

[202] M. Todd. A study of search directions in primal-dual interior-point methods for semidefinite programming. *Optimization Methods and Software*, 11:1–46, 1999.

[203] A. W. Tucker. On directed graphs and integer programs. Technical report, IBM Mathematical Research Project, 1960.

[204] R. H. Tütüncü, K. C. Toh, and M. J. Todd. Solving semidefinite-quadratic-linear programs using sdpt3. *Mathematical Programming*, 95:189–217, 2003.

[205] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38:49–95, 1996.

[206] A. Vanelli and G. S. Rowan. An eigenvector based approach for multistack vlsi layout. In *Proceedings of the Midwest Symposium on Circuits and Systems*, volume 29, pages 135–139, 1986.

[207] H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors. *Handbook of Semidefinite Programming*. Kluwer Academic Publishers, Boston, MA, 2000.

[208] S. Wright. *Primal-dual interior point methods*. SIAM, Philadelphia, 1997.

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

[210] Y. Zhang. On extending some primal–dual interior-point algorithms from linear programming to semidefinite programming. *SIAM Journal on Optimization*, 8:365–386, 1998.

Index

applications, 9–10, 20, 27, 31, 45, 70–71

benchmark instances, 19, 77, 79, 82, 86, 88, 94, 97, 100, 111
benchmark set, 114
Bipartite Crossing Minimization, 23, 45, 98–100, 110
Boundary Point Method, 11
Branch-and-Cut, 19–21, 28, 32, 37, 45, 46, 79, 85, 97, 111
Branch-and-Cut-and-Price, 28, 82
bundle method, 11
    dynamic version, 13–15, 41, 42, 77–81, 85, 94, 97–98, 110, 111

computational geometry, 56
Consecutive Ones Problem, 37
constraints
    complete-bipartite, 63–66
    degree, 62–66
CPLEX, 97, 111, 112
CSDP, 13, 35, 85, 88

elliptope, 12, 22, 47, 77
exact quadratic compactor, 68, 116–119

first-order methods, 11, 79

Goemans-Williamson rounding scheme, 10, 86–88
graph drawing, 45–68
Graphviz gallery, 103, 111

heuristic, 19, 31, 45, 60–61, 82, 85–89, 98, 103, 107, 111–114, 116
hpothetical routing, 58
Hungarian method, 70

inequalities
    clique, 32, 35, 54, 66, 85
    hypermetric, 32, 42
    pentagonal, 23, 35, 40–42, 54, 66
    rank, 28–29, 35
    triangle, 10, 23, 34–35, 40–42, 48, 75–79, 85, 94
interior-point methods, 11, 35, 75, 79
    primal-dual path-following, 12–13, 77

Lagrangian, 13–15, 77, 98
Laplacian, 10, 81
Linear Assignment Problem, 70
long-edge dummy nodes, 56–58, 114, 116, 118
Lovász θ-function, 9
lower bound
    combinatorial, 81
    spectral, 81
LS-cuts, 23, 49, 75–79, 81, 108, 116

Möbius ladder, 21, 43
Max-Cut, 9–10, 32, 37, 41, 86, 94–95
Minimax inequality, 8, 14, 15
Minimum Bandwidth Problem, 82
monotonous drawings, 60, 67–68
Multi-level Planarization, 56

North DAGs, 100–103, 114
NP-hard, 9, 10, 19, 27, 31, 37, 45, 56, 60, 69–70

optimality conditions, 8, 12
optimization
    eigenvalue, 9
    quadratic, 15

partition, 69–70
Physical Mapping Problem, 37
polytope, 103, 105, 111
    betweenness, 23, 32, 38–40, 93
    crossing, 49–54
    cut, 34, 40–42
    distance, 32
    linear ordering, 20, 75, 107
    linear-quadratic ordering, 22, 38–40
    multi-level quadratic ordering, 39, 40, 47, 49–54
    ordering, 93
    quadratic ordering, 34
    triple, 40
PORTA, 38–40
positional dummy nodes, 57–58, 61, 67, 116

programming
    dynamic, 27, 31
    linear, 7–8, 19–21, 27–29, 32–33, 46, 64, 79–80, 86, 89, 97, 103, 112, 114
    quadratic, 9, 33, 46, 61–64, 112

repair strategy, 86–88, 107
Rome graphs, 100–103, 114

sandwich theorem, 9
Schur complement theorem, 8, 22
SDPT3, 13
Sedumi, 13, 75, 93
shifted routing, 58–60
smallest linear subspace, 22, 32–34, 47
Stanford GraphBase generator, 98
strict complementarity, 8
strict feasibility, 8, 12, 14
strong duality, 8, 12
subgradient, 13–14
    inequality, 14
Sugiyama's framework, 45, 55, 100, 103

tanglegrams, 98

upward planarization, 45, 102, 114

variables
    betweenness, 19, 28–29, 32–34, 37, 39, 40, 85
    crossing, 46
    distance, 27–29, 31, 32, 85
    ordering, 20, 33–34, 37, 39, 40, 46, 61, 100
    position, 27
    triple, 21, 40
vertex-exchange graph, 45
verticality, 57

weighted Betweenness Problem, 37, 93