A REDUCTION FRAMEWORK FOR APPROXIMATE EXTENDED FORMULATIONS AND A FASTER ALGORITHM FOR CONVEX OPTIMIZATION

A Dissertation
Presented to
The Academic Faculty

By

Daniel Zink

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Industrial and Systems Engineering

Georgia Institute of Technology

April 2017

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Linear programming (LP) and semidefinite programming are among the most important tools in Operations Research and Computer Science. In this work we study the limitations of LPs and SDPs by providing lower bounds on the size of (approximate) linear and semidefinite programming formulations of combinatorial optimization problems. The hardness of (approximate) linear optimization implied by these lower bounds motivates the lazification technique for conditional gradient type algorithms. This technique allows us to replace (approximate) linear optimization in favor of a much weaker subroutine, achieving significant performance improvement in practice.

We can summarize the main contributions as follows:

(i) **Reduction framework for LPs and SDPs.** We present a new view on extended formulations that does not require an initial encoding of a combinatorial problem as a linear or semidefinite program. This new view allows us to define a purely combinatorial reduction framework transferring lower bounds on the size of exact and approximate LP and SDP formulations between different problems. Using our framework we show new LP and SDP lower bounds for a large variety of problems including Vertex Cover, various (binary and non-binary) constraint satisfaction problems as well as multiple optimization versions of Graph-Isomorphism.

(ii) **Exponential lower bound on the size of symmetric SDPs capturing Matching.** Yannakakis (1988) showed in his seminal work that every symmetric linear formulation of the matching problem has exponential size. Symmetry in this context means that for every permutation of the vertices of the underlying graph there is a permutation of the variables of the program leaving the program invariant. Using the sum of squares proof technique we show the semidefinite analog of Yannakakis's result that every symmetric semidefinite program for the matching problem has exponential size.

(iii) **Lazification technique for Conditional Gradient algorithms.** In Convex Programming conditional gradient type algorithms (also known as Frank-Wolfe type methods) are very important in theory as well as in practice due to their simplicity and fast convergence. We show how we can eliminate the linear optimization step performed in every iteration of Frank-Wolfe type methods and instead use a *weak separation oracle*. This oracle is significantly faster in practice and enables caching for additional improvements in speed and the sparsity of the obtained solutions.

Chapter 1

INTRODUCTION

Linear programming (LP) and semidefinite programming (SDP) are among the most important tools in Operations Research and Computer Science. Many exact and approximation algorithms for combinatorial optimization problems are using linear or semidefinite programming. For both paradigms there are algorithms known to find optimal solutions fast in theory, e.g., the Ellipsoid method or interior point methods, while for linear programming there are also practically much more efficient algorithms like the simplex method that are not provable efficient in theory. The main factor determining the running time of all these algorithms is the complexity of the feasible region measured in the number of defining inequalities in the LP case and the dimension of the semidefinite cone in the SDP case. In this thesis we are going to study lower bounds on the minimal size needed to describe combinatorial optimization problems exactly as well as approximately showing limitations on the solvability of (approximate) linear and semidefinite programs. We are further going to show how one can eschew (approximate) linear optimization in one of the most important algorithms in convex optimization, the Frank-Wolfe method, in favor of a much easier subroutine to avoid the mentioned limitations and get significant performance improvements in practice.

1.1 Extended Formulations

Extended formulations is the field studying the sizes of linear and semidefinite formulations. So far almost all lower bounds in extended formulations were shown on a case by case basis. That means one uses specific properties of the problem or polytope in question to show

that a high number of inequalities is needed for its exact or approximate description. We present a new view on extended formulations that does not require an initial encoding of the problem, i.e., instead of considering polytopes that capture a problem, we directly work with the problem itself. This new view is enabling the definition of a reduction framework that allows transferring lower bounds on the size of approximate linear and semidefinite formulations in a purely combinatorial way. The appeals of our reduction mechanism are as follows:

(i) The reduction mechanism is in spirit very similar to reductions in computational complexity. In fact we can reuse many reductions that were used to show computational hardness results. Additionally in order to apply a reduction one does not need to know any details on how the original hardness result was established.

(ii) Once relations between different problems are established through reductions, an improvement to the hardness of the base problem instantly improves all hardness results the base problem was reduced to without any further work. In fact since this reduction framework was presented the first time in Braun, Pokutta, and Zink (2015) the lower bounds from Chan et al. (2013) and Lee, Raghavendra, and Steurer (2014) have been improved by Braun, Pokutta, and Roy (2016) and Kothari, Meka, and Raghavendra (2016) allowing us to present stronger lower bounds in this work while using the same reductions.

(iii) Having a reduction mechanism in place directly imposes an order on combinatorial problems. In this context in particular the question for complete problems is of interest, since we expect the set of hard problems to be quite different due to the fact that Max-Matching is one of the hardest problems in linear programming (see Rothvoß 2014; Braun and Pokutta 2015a) despite its polynomial time solvability. This question however is not the focus of this work.

(iv) A reduction mechanism also propagates upper bounds, i.e., having a combinatorial

problem with a small (approximate) linear or semidefinite formulation allows us to get small formulations for all problems that we can reduce to the original problem. As we will see in this chapter the polyhedral part of the formulation will actually be unchanged by the reduction, i.e., having problem A with a small formulation described by a polyhedron P and a reduction from problem B to problem A, then P can be used as the polyhedral part of the formulation for problem B. The reduction provides a way to rewrite or reinterpret the solutions and instances in order to make the formulation P suitable for both problems.

Our abstract view on extended formulations is based on Chan et al. (2013). Both approaches, the original as well as our abstract one, do not use a particular initial encoding of the combinatorial problem, instead both work with the instances and solutions of the problem itself. We generalize the work of Chan et al. (2013) to capture arbitrary problems as opposed to being restricted to constraint satisfaction problems (CSPs), which have the full 0/1 cube as a feasible region. Further since our framework is also able to capture approximations of optimization problems, it is also a generalization of the results in Braun et al. (2012) and Braun et al. (2014a) working with polyhedral pairs to capture approximations. The motivating question for this work is:

Given a combinatorial optimization problem,
what is the smallest size of any of its LPs or SDPs?

Again we want to stress, we start with a combinatorial optimization problem and not with a specific polytope capturing the problem at hand. While this difference to the traditional extended formulation model is more of a philosophical nature and the results are equivalent to those obtained via the traditional extended formulations setup, on a technical level this perspective significantly simplifies the treatment of approximate LP/SDP formulations and it enables the formulation of the reduction mechanism.

We are able to establish lower bounds on a large variety of different problems using reductions. Among them are NP-hard problems like VertexCover and IndependentSet, there

are CSPs, like Max-k-SAT, which is a binary CSP and Max-MULTI-k-CUT, which is non-binary, and different optimization versions of the Graph-Isomorphism problem, whose decision version is not yet know to be in P or NP-complete. Recently in a breakthrough result Babai (2015) showed that there is a quasi polynomial algorithm deciding Graph-Isomorphism.

In the rest of this section we establish our new view on extended formulations, we define when a LP or an SDP is capturing a combinatorial problem and what symmetry means in this context. We finally relate our approach to polyhedral pairs which were used in extended formulations before to capture approximations.

1.1.1 Optimization problems

We intend to study the required size of a linear program or semidefinite program capturing a combinatorial optimization problem with specified approximation guarantees. In this section we present our encoding independent view on extended formulation. An optimization problem in this context is defined as follows.

Definition 1.1.1 (Optimization problems)**.** An *optimization problem* $\mathcal{P} = (\mathcal{S}, \mathcal{F}, \text{val})$ consists of a set $\mathcal{S}$ of *feasible solutions* and a set $\mathcal{F}$ of *instances*, together with a real-valued objective function $\text{val}\colon \mathcal{S} \times \mathcal{F} \to \mathbb{R}$.

A wide class of examples consist of constraint satisfaction problems (CSPs):

Definition 1.1.2 (Maximum Constraint Satisfaction Problem (CSP))**.** A *constraint family* $\mathcal{C} = \{C_1, \ldots, C_m\}$ on the boolean variables $x_1, \ldots, x_n$ is a family of boolean functions C_i in $x_1, \ldots, x_n$. The C_i are *constraints* or *clauses*. The problem $\mathcal{P}(\mathcal{C})$ corresponding to a constraint family $\mathcal{C}$ has

(i) **feasible solutions** all 0/1 assignments s to $x_1, \ldots, x_n$;

(ii) **instances** all nonnegative weightings $w_1, \ldots, w_m$ of the constraints $C_1, \ldots, C_m$

(iii) **objective function** the weighted sum of satisfied constraints: $\text{sat}_{w_1,\ldots,w_m}(s) = \sum_i w_i C_i(s)$.

The goal is to maximize the weights of satisfied constraints, in particular CSPs are maximization problems. A *maximum Constraint Satisfaction Problem* is an optimization problem $\mathcal{P}(\mathcal{C})$ for some constraint family $\mathcal{C}$. A k-CSP is a CSP where every constraint depends on at most k variables.

For brevity, we shall simply use CSP for a maximum CSP, when there is no danger of confusion with a minimum CSP. In the following, we shall restrict to instances with 0/1 weights, i.e., an instance is a subset $L \subseteq \mathcal{C}$ of constraints, and the objective function computes the number $\mathrm{sat}_L(s) = \sum_{C \in L} C(s)$ of constraints in L satisfied by assignment s. Restriction to specific instances clearly does not increase formulation complexity.

As a special case, the Max-k-XOR problem restricts to constraints, which are XORs of at most k literals. Here we shall write the constraints in the equivalent equation form $x_{i_1} \oplus \cdots \oplus x_{i_k} = b$, where $\oplus$ denotes the addition modulo 2.

Definition 1.1.3 (Max-k-XOR)**.** For fixed k and n, the problem Max-k-XOR is the CSP for variables $x_1, \ldots, x_n$ and the family $\mathcal{C}$ of all constraints of the form $x_{i_1} \oplus \cdots \oplus x_{i_l} = b$ with $1 \leq i_1 < \cdots < i_l \leq n$, $b \in \{0,1\}$ and $l \leq k$.

An even stronger important restriction is MaxCUT, a subproblem of Max-2-XOR as we will see soon. The aim is to determine the maximum size of cuts for all graphs G with $V(G) = [n]$.

Definition 1.1.4 (MaxCUT)**.** The problem MaxCUT has instances all simple graphs G with vertex set $V(G) = [n]$, and feasible solutions all cuts on $[n]$, i.e., functions $s\colon [n] \to \{0,1\}$. The objective function val computes the number of edges $\{i,j\}$ of G cut by the cut, i.e., with $s(i) \neq s(j)$.

The problem MaxCUT_Δ is the subproblem of MaxCUT considering only graphs G with maximum degree at most Δ.

In order to realize MaxCUT as a subproblem of Max-2-XOR we set $\mathrm{val}_G(s) = \mathrm{sat}_{L(G)}(s)$, for the set of constraints $L(G) = \{x_i \oplus x_j = 1 \mid \{i,j\} \in E(G)\}$, while using the same

feasible solutions. Since Max-k-XOR contains all constraints with at most k literals Max-l-XOR is subproblem of Max-k-XOR for $l \leq k$.

1.1.2 Linear and semidefinite formulations

We are interested in approximately solving an optimization problem $\mathcal{P}$ by means of a linear program or a semidefinite program. Recall that a typical PCP inapproximability result states that it is hard to decide between $\max \mathrm{val}_i \leq S(i)$ and $\max \mathrm{val}_i \geq C(i)$ for a class of instances i and some easy-to compute functions S and C usually refereed to as *soundness* and *completeness*. Here and below $\max \mathrm{val}_i$ denotes the maximum value of the function val_i over the respective set of feasible solutions. We adopt the terminology to linear programs and semidefinite programs. We start with the linear case.

Definition 1.1.5 (LP formulation of an optimization problem)**.** Let $\mathcal{P} = (\mathcal{S}, \mathcal{F}, \mathrm{val})$ be an optimization problem with real-valued functions C, S on $\mathcal{F}$, called *completeness guarantee* and *soundness guarantee*, respectively (or *approximation guarantees* together). If $\mathcal{P}$ is a maximization problem, then let $\mathcal{F}^S := \left\{ f \in \mathcal{F} \,\middle|\, \max \mathrm{val}_f \leq S(f) \right\}$ denote the set of instances, for which the maximum is upper bounded by soundness guarantee S. If $\mathcal{P}$ is a minimization problem, then let $\mathcal{F}^S := \left\{ f \in \mathcal{F} \,\middle|\, \min \mathrm{val}_f \geq S(f) \right\}$ denote the set of instances, for which the minimum is lower bounded by soundness guarantee S.

A (C, S)-*approximate LP formulation* of $\mathcal{P}$ is a linear program $Ax \leq b$ with $x \in \mathbb{R}^d$ together with the following *realizations*:

(i) **Feasible solutions** as vectors $x^s \in \mathbb{R}^d$ for every $s \in \mathcal{S}$ so that

$$Ax^s \leq b \qquad \text{for all } s \in \mathcal{S}, \tag{1.1}$$

i.e., the system $Ax \leq b$ is a relaxation (superset) of $\mathrm{conv}\left(x^s \mid s \in \mathcal{S}\right)$.

(ii) **Instances** as affine functions $w^f : \mathbb{R}^d \to \mathbb{R}$ for every $f \in \mathcal{F}^S$ such that

$$w^f(x^s) = \mathrm{val}_f(s) \qquad \text{for all } s \in \mathcal{S}, \tag{1.2}$$

i.e., we require that the linearization w^f of val_f is exact on all x^s with $s \in \mathcal{S}$.

(iii) **Achieving guarantee** C via requiring

$$\max\left\{ w^f(x) \;\middle|\; Ax \leq b \right\} \leq C(f) \qquad \text{for all } f \in \mathcal{F}^S, \tag{1.3}$$

for maximization problems (resp. $\min\left\{ w^f(x) \;\middle|\; Ax \leq b \right\} \geq C(f)$ for minimization problems).

The *size* of the formulation is the number of inequalities in $Ax \leq b$. Finally, the *LP formulation complexity* $\mathrm{fc}_+(\mathcal{P}, C, S)$ of the problem $\mathcal{P}$ is the minimal size of all its LP formulations.

For all instances $f \in \mathcal{F}$ soundness and completeness should satisfy $C(f) \geq S(f)$ in the case of maximization problems and $C(f) \leq S(f)$ in the case of minimization problems in order to capture the notion of relaxations and we assume this condition in the remainder of the paper.

Remark 1.1.6 (Inequalities vs. Equations). Traditionally in extended formulations, one would separate the description into equations and inequalities and one would only count inequalities. In our framework, equations can be eliminated by restricting to the affine space defined by them, and parameterizing it as a vector space. However, note that restricting to linear functions, one might need an equation to represent affine functions by linear functions.

For determining the exact maximum of a maximization problem, one chooses $C(f) = S(f) := \max \mathrm{val}_f$. In this case we also omit C and S and write $\mathrm{fc}_+(\mathcal{P})$ for $\mathrm{fc}_+(\mathcal{P}, C, S)$. To show inapproximability within an approximation factor $0 < \rho \leq 1$, one chooses guarantees satisfying $\rho C(f) \geq S(f)$. This choice is motivated to be comparable with factors of

approximation algorithms finding a feasible solution s with $\mathrm{val}_f(s) \geq \rho \max \mathrm{val}_f$. For minimization problem, $C(f) = S(f) := \min \mathrm{val}_f$ in the exact case, and $\rho C(f) \leq S(f)$ for an approximation factor $\rho \geq 1$ provided val_f is nonnegative. Again in the exact case we also write $\mathrm{fc}_+(\mathcal{P})$ for $\mathrm{fc}_+(\mathcal{P}, C, S)$. This model of *outer* approximation streamlines the models in Chan et al. (2013), Braun, Fiorini, and Pokutta (2016), and Lee et al. (2014), and also captures, simplifies, and generalizes approximate extended formulations from Braun et al. (2012) and Braun et al. (2014a); see Section 1.1.4 for a discussion.

We will now adjust Definition 1.1.5 to the semidefinite case. For symmetric matrices, as customary, we use the Frobenius product as scalar product, i.e., $\langle A, B \rangle = \mathrm{Tr}[AB]$. Recall that the psd-cone is self-dual under this scalar product.

Definition 1.1.7 (SDP formulation of an optimization problem)**.** Let $\mathcal{P} = (\mathcal{S}, \mathcal{F}, \mathrm{val})$ be a maximization problem with real-valued functions C, S on $\mathcal{F}$. As in Definition 1.1.5, let $\mathcal{F}^S := \left\{ f \in \mathcal{F} \,\middle|\, \max \mathrm{val}_f \leq S(f) \right\}$.

A (C, S)-approximate *SDP formulation* of $\mathcal{P}$ consists of a linear map $\mathcal{A} \colon \mathbb{S}^d \to \mathbb{R}^k$ and a vector $b \in \mathbb{R}^k$ (defining a semidefinite program $\left\{ X \in \mathbb{S}^d_+ \,\middle|\, \mathcal{A}(X) = b \right\}$). Moreover, we require the following *realizations* of the components of $\mathcal{P}$:

(i) **Feasible solutions** as vectors $X^s \in \mathbb{S}^d_+$ for every $s \in \mathcal{S}$ so that

$$\mathcal{A}(X^s) = b \tag{1.4}$$

i.e., the system $\mathcal{A}(X) = b, X \in \mathbb{S}^d_+$ is a relaxation of $\mathrm{conv}\left(X^s \,|\, s \in \mathcal{S}\right)$.

(ii) **Instances** as affine functions $w^f \colon \mathbb{S}^d \to \mathbb{R}$ for every $f \in \mathcal{F}^S$ with

$$w^f(X^s) = \mathrm{val}_f(s) \qquad \text{for all } s \in \mathcal{S}, \tag{1.5}$$

i.e., we require that the linearization w^f of val_f is exact on all X^s with $s \in \mathcal{S}$.

(iii) **Achieving guarantee** C via requiring

$$\max\left\{w^f(X)\;\middle|\;\mathcal{A}(X^s)=b,\ X^s\in\mathbb{S}^d_+\right\}\leq C(f)\qquad\text{for all } f\in\mathcal{F},\tag{1.6}$$

for maximization problems, and the analogous inequality for minimization problems.

The *size* of the formulation is the parameter d. The *SDP formulation complexity* $\mathrm{fc}_\oplus(\mathcal{P},C,S)$ of the problem $\mathcal{P}$ is the minimal size of all its SDP formulations.

We also write $\mathrm{fc}_\oplus(\mathcal{P})$ for $\mathrm{fc}_\oplus(\mathcal{P},C,S)$ if we are interested in SDP formulations that capture the exact problem, i.e., if $C(f)=S(f)=\mathrm{opt}_{\mathcal{P}}\,\mathrm{val}_f$.

1.1.3 Symmetric semidefinite formulations

In this section we define symmetric semidefinite programming formulations, since we are only going to use symmetry in the semidefinite case. However one can define symmetric linear formulations as well.

Recall that a group action of a group G on a set X is defined by a function $\varphi\colon G\times X\to X\colon (g,x)\mapsto\varphi(g,x)$ such that $\varphi(e,x)=x$ for all $x\in X$ and e being the neutral element of G and $\varphi(gh,x)=\varphi(g,\varphi(h,x))$ for all $g,h\in G$ and $x\in X$. For a left action of G on X we also write $g\cdot x$ for $\varphi(g,x)$. The *orbit* of $x\in X$ is $\{g\cdot x\,|\,g\in G\}$ while the *stabilizer* of x is $\{g\in G\,|\,g\cdot x=x\}$. Let A_n denote the alternating group on n letters (the set of even permutations of $[n]$).

Definition 1.1.8 (G-symmetric maximization problem)**.** Let G be group and $\mathcal{P}=(\mathcal{S},\mathcal{F},\mathrm{val})$ be a maximization problem, then $\mathcal{P}$ is called G-symmetric if $\mathrm{val}_{g\cdot f}(g\cdot s)=\mathrm{val}_f(s)$ for all $f\in\mathcal{F}$, $s\in\mathcal{S}$ and $g\in G$.

For a G-symmetric problem we require G-symmetric soundness and completeness guarantees: $C(g\cdot f)=C(f)$ and $S(g\cdot f)=S(f)$ for all $f\in\mathcal{F}$ and $g\in G$.

We now define the notion of a G-symmetric semidefinite programming formulation of a maximization problem.

Definition 1.1.9 (G-symmetric SDP formulation for $\mathcal{P}$)**.** Let G be a group acting on $\mathbb{S}^d_+$ and $\mathcal{P} = (\mathcal{S}, \mathcal{F})$ be a G-symmetric maximization problem with G-symmetric approximation guarantees C and S. Then the (C, S)-approximate SDP formulation of $\mathcal{P}$ consisting of $\mathcal{A}\colon \mathbb{S}^d_+ \to \mathbb{R}^k$ and $b \in \mathbb{R}^k$ together with the realizations $\{X^s \mid s \in \mathcal{S}\}$ and $\{w^f \mid f \in \mathcal{F}\}$ of size d (see Definition 1.1.7) is called G-symmetric if the following compatibility conditions are satisfied for all $g \in G$:

(i) *Action on solutions:* $X^{g\cdot s} = g \cdot X^s$ for all $s \in \mathcal{S}$.

(ii) *Action on functions:* $w^{g\cdot f}(g \cdot X) = w^f(X)$ for all $f \in \mathcal{F}$ with $\max_{s\in\mathcal{S}} f(s) \leq S(f)$.

(iii) *Invariant affine space:* $\mathcal{A}(g \cdot X) = \mathcal{A}(X)$.

A G-symmetric SDP formulation is *G-coordinate-symmetric* if the action of G on $\mathbb{S}^d_+$ is by permutation of coordinates: that is, there is an action of G on $[d]$ with $(g \cdot X)_{ij} = X_{g^{-1}\cdot i, g^{-1}\cdot j}$ for all $X \in \mathbb{S}^d_+$, $i, j \in [d]$ and $g \in G$.

1.1.4 Relation to approximate extended formulations

Traditionally in extended formulations, one would start from an initial polyhedral representation or polyhedral encoding of the problem and bound the size of its smallest possible lift in higher-dimensional space. In the linear case for example, the minimal number of required inequalities would constitute the extension complexity of that polyhedral representation or encoding. Our notion of *formulation complexity* can be understood as the minimum extension complexity over all possible polyhedral encodings of the optimization problem. This independence of encoding addresses previous concerns that the obtained lower bounds are polytope-specific or encoding-specific and alternative linear encodings (i.e., different initial polyhedron) of the same problem might admit smaller formulations: we show that this is not the case. More precisely, in view of the results from above the standard notion of extension complexity and formulation complexity are essentially equivalent, however the

more abstract perspective simplifies the handling of approximations and reductions as we will see in Section 2.2.

The notion of *LP formulation*, its size, and LP formulation complexity are closely related to *polyhedral pairs* and linear encodings (see Braun et al. 2012; Braun et al. 2014a, and also Pashkovich 2012). In particular, given a (C, S)-approximate LP formulation of a maximization problem $\mathcal{P}$ with linear program $Ax \leq b$, representations $\{x^s \mid s \in \mathcal{S}\}$ of feasible solutions and $\left\{w^f \;\middle|\; f \in \mathcal{F}^S\right\}$ of instances, one can define a polyhedral pair encoding of the problem $\mathcal{P}$ as

$$\begin{aligned} P &:= \operatorname{conv}\left(x^s \mid s \in \mathcal{S}\right), \\ Q &:= \left\{x \in \mathbb{R}^d \;\middle|\; \langle w^f, x\rangle \leqslant C(f),\ \forall f \in \mathcal{F}^S\right\}. \end{aligned} \tag{1.7}$$

Then for $K := \{x \mid Ax \leq b\}$, we have $P \subseteq K \subseteq Q$. Note that there is no need for the approximating polyhedron K to reside in extended space, as P and Q are already defined in extended space.

Put differently, the LP formulation complexity of $\mathcal{P}$ is the minimum size of an extended formulation over all possible linear encodings of $\mathcal{P}$. The semidefinite case is similar, with the only difference being that K is now a spectrahedron, being represented by a semidefinite program instead of a linear program.

1.2 Conditional Gradient method

Convex optimization is important from a theoretical as well as from an applications point of view covering applications like video colocalization and matrix completion. The general convex optimization problem is defined as

$$\min_{x \in P} f(x), \tag{1.8}$$

where f is a smooth and convex function and P is a polytope. Access to the function f is given by a first order oracle that returns $f(x)$ and $\nabla f(x)$ for a point $x \in P$. The access to the polytope P depends on the algorithm that we use to solve Problem (1.8) as we will see in the following. If we apply regular gradient descent methods to this problem, we have to solve a projection step in each iteration, since after a step in the direction of the gradient, there is no guarantee that the new iterate will be in the polytope P again. Since projections can be computationally very expensive, projection-free methods like the Frank-Wolfe method (see Frank and Wolfe 1956) (also known as Conditional Gradient Descent, see Levitin and Polyak 1966) gained a lot of attention. For this type of method the access to P is by a linear optimization oracle, which is called in each iteration to get the direction for the next step. Feasibility is maintained by using a step length in $[0, 1]$. The most basic version of the Frank-Wolfe method is given in Algorithm 1. Although convergence rates can be suboptimal,

Algorithm 1 Frank-Wolfe Algorithm (Frank and Wolfe 1956)

Require: smooth convex f function with curvature C, $x_1 \in P$ start vertex, LP_P linear minimization oracle
Ensure: x_t points in P
1: **for** $t = 1$ **to** $T - 1$ **do**
2: $\quad v_t \leftarrow \mathrm{LP}_P(\nabla f(x_t))$
3: $\quad x_{t+1} \leftarrow (1 - \gamma_t)x_t + \gamma_t v_t$ with $\gamma_t := \frac{2}{t+2}$
4: **end for**

solving only a single linear optimization problem over P per iteration still leads to significant speed up in wall clock time compared to using a projection in every step (see e.g., Hazan and Kale 2012). Another advantage of Conditional Gradients methods is that the constructed solution is naturally sparse as a convex combination of extreme points (or atoms), since per iteration at most one new term is added.

Remark that in this context linear optimization and linear optimization oracle refers to solving the problem

$$\min\{cx \mid x \in K\},$$

where $c \in \mathbb{R}^n$ and K is a convex set. In particular linearity here refers to the linear objective

function and not to the feasible region. So the feasible region can be a semidefinite program or even a general convex set.

Unfortunately optimizing linear functions can still be costly, especially if the corresponding problem is NP-hard, e.g., when considering MIPs, or the feasible region has high dimension which is not uncommon in many machine learning applications. Additionally in practice we mostly want to use generic solvers like CPLEX or Gurobi to implement the linear optimization, however in this case the lower bounds shown with the reduction mechanism give an indication that linear optimization is often a hard and time consuming task. Therefore our approach is to use a *weak separation oracle* which is much weaker than linear optimization (see Oracle 1). The first two parameters of Oracle 1, $c \in \mathbb{R}^n$, and $x \in P$,

Oracle 1 Weak Separation Oracle $\text{LPsep}_P(c, x, \Phi, K)$

Require: $c \in \mathbb{R}^n$ linear objective, $x \in P$ point, $K \geq 1$ accuracy, $\Phi > 0$ objective value;
Ensure: Either (1) $y \in P$ vertex with $c(x - y) > \Phi/K$, or (2) **false**: $c(x - z) \leq \Phi$ for all $z \in P$.

are the same as for the linear optimization routine and will be used with the gradient and the current iterate respectively in the Conditional Gradient algorithms. The third parameter Φ is used to guide the progress per iteration: if the oracle returns with the first case (positive case) then we get a new point y that is at least Φ/K better than the current point. In the second (negative) case the oracle gives a guarantee that we are already close enough to the optimal point for the desired convergence rate. Since for the positive case larger values of Φ are favorable and for the negative case smaller ones, we will see in Chapter 4 how to set Φ as a function in t in order to balance these two cases and achieve the convergence rates that we need. Finally the last parameter K is an accuracy parameter. Observe that in the case $K > 1$ the conditions for the two outcomes of the weak separation oracle overlap, which makes it easier (and therefore faster) for a routine to decide between the two cases.

The two main advantages of the weak separation oracle are *early termination* and the possibility of using a *cache*. Early termination means that the oracle returns, when there is a *good enough* point instead of an optimal one, i.e., an optimization routine implementing

the oracle can terminate early. We also call this *lazy* optimization and therefore methods using the weak separation oracle lazy or lazified methods. Further since the condition for the positive case does not involve a guarantee with respect to the optimal solution, we can hold previously used points in a cache and return one that satisfies $c(x-y) > \Phi/K$. This condition can usually be checked much faster than computing a new point. Reusing previous points has the additional advantage that the final solution might be much sparser. For a more detailed discussion of the properties and advantages of weak separation over linear optimization see Remark 4.1.2.

We want to emphasize that our approach is different from the Conditional Gradient method using approximate optimization (see Jaggi 2013), since finding an approximate optimal solution still requires a comparison to the value of the optimal solution and therefore the computation of lower bounds on the optimal objective value. In our approach we only compare the objective value of the current point with the objective value of the new point, which gives us significant computational advantages as we will see in Section 4.5 where we compare our lazified algorithms to the non-lazy counterparts with approximate optimization. After we introduced some notation in the next section, we show in Section 1.2.2 a convergence proof for the Conditional Gradient method with approximate optimization.

1.2.1 Preliminaries

A function $f\colon \mathbb{R}^n \to \mathbb{R}$ is called *convex* if its linearization lies below the function itself:

$$f(y) \geq f(x) + \nabla f(x)(y-x).$$

Let $\|\cdot\|$ be an arbitrary norm on $\mathbb{R}^n$, and let $\|\cdot\|^*$ denote the dual norm of $\|\cdot\|$. We will specify the applicable norm in the later sections. A function f is *L-Lipschitz* if $|f(y) - f(x)| \leq L\|y-x\|$ for all $x, y \in \operatorname{dom} f$. A convex function f is called *smooth* with curvature at

most C if

$$f(\gamma y + (1-\gamma)x) \leq f(x) + \gamma \nabla f(x)(y-x) + C\gamma^2/2$$

for all $x, y \in \operatorname{dom} f$ and $0 \leq \gamma \leq 1$. A function f is *S-strongly convex* if

$$f(y) - f(x) \geq \nabla f(x)(y-x) + \frac{S}{2}\|y-x\|^2$$

for all $x, y \in \operatorname{dom} f$. Unless stated otherwise Lipschitz continuity and strong convexity will be measured in the norm $\|\cdot\|$. Moreover, let $\mathbb{B}_r(x) := \{y \mid \|x-y\| \leq r\}$ be the ball around x with radius r with respect to $\|.\|$. In the following, P will denote the feasible region, a polytope and the vertices of P will be denoted by $v_1, \ldots, v_N$. We will use x^* as the optimal point of Problem 1.8.

One important quantity when working with Conditional Gradient methods is the duality gap or Wolfe gap.

Definition 1.2.1 (Duality gap)**.** Given Problem 1.8 and a feasible point $x \in P$, then the *duality gap* or *Wolfe gap* of this point is defined by:

$$g(x) := \max_{y \in P} \nabla f(x)(x-y).$$

The importance arises from the fact that for a convex function f we can bound the primal error of a feasible point using the duality gap:

$$f(x) - f(x^*) \leq \nabla f(x)(x-x^*) \leq \max_{y \in P} \nabla f(x)(x-y) = g(x).$$

In Frank-Wolfe type algorithms the quantity $g(x)$ is computed in each step as a byproduct of finding the direction for the next step, so although the actual value $f(x^*)$ is unknown one can upper bound the accuracy of the current iterate in every step and for example terminate the algorithm if a certain target accuracy is reached.

1.2.2 Approximate Conditional Gradients

In order to highlight the differences between the analysis of the Conditional Gradient method using approximate optimization of Jaggi (2013) and our lazified methods, we show its convergence in this section. The algorithm is the same as Algorithm 1 only that we use in Line 2 instead of the exact optimization oracle $\mathrm{LP}_P(\nabla f(x_t)) = \arg\min_{x \in P} \nabla f(x_t)x$ an approximate one, setting v_t such that $\nabla f(x_t)v_t \leq \arg\min_{x \in P} \nabla f(x_t)x + \frac{1}{2}\delta\gamma C$, where C is the curvature of f and δ is a fixed accuracy parameter.

Theorem 1.2.2 (Theorem 1, Jaggi 2013)**.** *For each $k \geq 1$, the iterates x_t of the Conditional Gradients method using approximate optimization satisfy*

$$f(x_t) - f(x^*) \leq \frac{2C}{t+2}(1+\delta),$$

where $x^* \in P$ is the optimal solution of Problem (1.8)*.*

The proof of this theorem is based on a lower bound on the improvement in each iteration which is given in Lemma 1.2.3.

Lemma 1.2.3 (Lemma 5, Jaggi 2013)**.** *Let f be a strongly convex function. If v_t is an approximate linear minimizer, i.e., $\nabla f(x_t)v_t \leq \min_{y \in P} \nabla f(x_t)y + \frac{1}{2}\delta\gamma C$, and $x_{t+1} \coloneqq (1-\gamma)x_t + \gamma v_t$ for an arbitrary $\gamma \in [0,1]$, then*

$$f(x_{t+1}) \leq f(x_t) - \gamma g(x_t) + \frac{\gamma^2}{2}C(1+\delta) \tag{1.9}$$

holds.

Proof. Using strong convexity of f provides

$$\begin{aligned} f(x_{t+1}) &= f((1-\gamma)x_t + \gamma v_t) \\ &\leq f(x_t) + \gamma \nabla f(x_t)(v_t - x_t) + \frac{\gamma^2}{2}C. \end{aligned}$$

In order to bound the middle term we use that v_t is the solution of the approximate optimization in this step:

$$\nabla f(x_t)(v_t - x_t) \leq \min_{y \in P} \nabla f(x_t)y - \nabla f(x_t)x_t + \frac{1}{2}\delta\gamma C = -g(x_t) + \frac{1}{2}\delta\gamma C, \quad (1.10)$$

and together we get the desired bound

$$\begin{aligned} f(x_{t+1}) &\leq f(x_t) + \gamma \nabla f(x_t)(v_t - x_t) + \frac{\gamma^2}{2}C \\ &\leq f(x_t) - \gamma g(x_t) + \frac{1}{2}\delta\gamma^2 C + \frac{\gamma^2}{2}C \\ &= f(x_t) - \gamma g(x_t) + \frac{\gamma^2}{2}C(1+\delta). \end{aligned}$$

□

The inequality that we have to replace when showing the convergence of a lazified method is Equation 1.10. As we will see in Chapter 4 we can replace this inequality with two different cases tailored to the outcomes of the weak separation oracle, where in the positive case no comparison to $\nabla f(x_t)x^*$ is used leading to the improved performance.

The proof of the convergence rate is as follows:

Proof of Theorem 1.2.2. We first show a recursive inequality for the primal gap

$$\begin{aligned} f(x_{t+1}) - f(x^*) &\leq f(x_t) - f(x^*) - \gamma_t g(x_t) + \frac{\gamma_t^2}{2}C(1+\delta) \\ &\leq f(x_t) - \gamma_t(f(x_t) - f(x^*)) + \frac{\gamma_t^2}{2}C(1+\delta) \\ &= (1-\gamma_t)(f(x_t) - f(x^*)) + \frac{\gamma_t^2}{2}C(1+\delta), \end{aligned}$$

where we used Lemma 1.2.3 in the first inequality and the fact that the duality gap is a bound on the primal error in the second inequality.

We now show by induction over t the statement of the theorem. The base case $t = 0$ follows with $\gamma_t = \frac{2}{0+2} = 1$ and the recursive inequality above.

For $t \geq 1$ using the recursive inequality again and the induction hypothesis we get

$$\begin{aligned}
f(x_{t+1}) - f(x^*) &\leq (1-\gamma_t)(f(x_t) - f(x^*)) + \frac{\gamma_t^2}{2} C(1+\delta) \\
&= \left(1 - \frac{2}{t+2}\right)(f(x_t) - f(x^*)) + \frac{4}{2(t+2)^2} C(1+\delta) \\
&\leq \left(1 - \frac{2}{t+2}\right)\frac{2C}{t+2}(1+\delta) + \frac{4}{2(t+2)^2} C(1+\delta) \\
&= \left(1 - \frac{2}{t+2} + \frac{1}{t+2}\right)\frac{2}{t+2} C(1+\delta) \\
&= \frac{t+1}{t+2} \cdot \frac{2}{t+2} C(1+\delta) \\
&\leq \frac{t+2}{t+3} \cdot \frac{2}{t+2} C(1+\delta) \\
&= \frac{2}{t+3} C(1+\delta),
\end{aligned}$$

which shows the desired inequality. □

Chapter 2

A REDUCTION FRAMEWORK FOR EXTENDED FORMULATIONS

The contribution in this chapter has three parts, were the most important one is the reduction framework. Despite the similarity to reductions in computational complexity we stress that all hardness results we show in this chapter are independent of P vs. NP.

(i) **Factorization Theorem for Combinatorial Problems.** Yannakakis's factorization theorem is one of the most important tools in extended formulations (see Yannakakis 1991; Yannakakis 1988). It relates the size of an extended formulation of a polytope with an algebraic property of the slack matrix of that polytope. To date many variations and extensions of the original theorem have been stated and used, see e.g., Gouveia, Parrilo, and Thomas (2013), Braun et al. (2012), Braun et al. (2014a), and Chan et al. (2013). We show a unified version of this statement suited for our abstract, encoding independent view.

(ii) **Reduction mechanism.** The main contribution in this chapter is our purely combinatorial and conceptually simple reduction mechanism that allows us to propagate inapproximability results as well as small linear and semidefinite formulations between problems.

(iii) **LP and SDP inapproximability of specific problems.** We use our reduction framework to establish linear and semidefinite inapproximability results for many combinatorial problems. For the linear formulations we use as the base hardness a very recent result by Kothari, Meka, and Raghavendra (2016) showing an inapproxima-

bility of MaxCUT. In the semidefinite case we also use MaxCUT as a base problem with hardness results established by Lee, Raghavendra, and Steurer (2014) in the exact case and by Braun, Pokutta, and Roy (2016) in the approximate case, while the latter was shown by a reduction ultimately going back to a hardness result by Lee, Raghavendra, and Steurer (2014) as well. The problems we establish hardness results for are among others VertexCover, one of the most important problems in extended formulations, Max-k-SAT, a binary CSP, Max-MULTI-k-CUT, a non-binary CSP and several optimization versions of Graph-Homomorphism and Graph-Isomorphism.

Related work

Our encoding independent view and the reduction framework can be seen as an extension and generalization of different lines of work: First Avis and Tiwary (2015) and Pokutta and Van Vyve (2013) showed lower bounds on the exact extension complexity of various polytopes using a reduction framework, however they cannot capture approximations of polytopes or problems. Second the approach in Chan et al. (2013) (and its improvements in Kothari, Meka, and Raghavendra (2016)) is, like ours, encoding independent and able to capture approximations, however only applicable to CSPs, which have the 0/1 cube as a feasible region. Finally we generalize the work of Braun et al. (2014a) and Braun et al. (2012) achieving inapproximability results for linear formulations by using polyhedral pairs however lacking the encoding independent view provided in this work.

For lower bounds on linear extended formulations there have been many results for different polytopes, see e.g., Yannakakis (1991), Yannakakis (1988), Fiorini et al. (2012), and Rothvoß (2014). Recently there has been progress both in the linear as well as in the semidefinite case through the connection to hierarchies (Chan et al. 2013; Lee et al. 2014; Lee, Raghavendra, and Steurer 2014), however most of this work is concerned with CSPs. The reduction framework that we present here also led to several improvements already: Bazzi et al. (2015) show a tight linear inapproximability result for VertexCover improving

on the result we will present in Section 2.3 and Braun, Pokutta, and Roy (2016) established the first SDP inapproximability for MaxCUT by extending reductions to work with fractional objective functions. We in turn use the latter result again as the base hardness for many of our reductions to establish SDP hardness for different problems.

2.1 Factorization theorem and slack matrix

We provide an algebraic characterization of formulation complexity via the *slack matrix of an optimization problem*, similar in spirit to factorization theorems for extended formulations (see e.g., Yannakakis 1991; Yannakakis 1988; Gouveia, Parrilo, and Thomas 2013; Braun et al. 2012; Braun et al. 2014a), with a fundamental difference pioneered in Chan et al. (2013) that there is no linear system to start from. The linear or semidefinite program is constructed from scratch using a matrix factorization. This also extends Braun, Fiorini, and Pokutta (2016), by allowing affine functions, and using a modification of nonnegative rank, to show that formulation complexity depends only on the slack matrix.

Definition 2.1.1 (Slack matrix of $\mathcal{P}$)**.** Let $\mathcal{P} = (\mathcal{S}, \mathcal{F}, \mathrm{val})$ be an optimization problem with guarantees C, S. The (C, S)*-approximate slack matrix* of $\mathcal{P}$ is the nonnegative $\mathcal{F}^S \times \mathcal{S}$ matrix M, with entries

$$M(f,s) := \begin{cases} C(f) - \mathrm{val}_f(s) & \text{if } \mathcal{P} \text{ is a maximization problem,} \\ \mathrm{val}_f(s) - C(f) & \text{if } \mathcal{P} \text{ is a minimization problem.} \end{cases}$$

We introduce the *LP factorization* of a nonnegative matrix, which for slack matrices captures the LP formulation complexity of the underlying problem.

Definition 2.1.2 (LP factorization of a matrix)**.** A *size-r LP factorization* of $M \in \mathbb{R}_+^{m \times n}$ is a factorization $M = TU + \mu\mathbb{1}$ where $T \in \mathbb{R}_+^{m \times r}$, $U \in \mathbb{R}_+^{r \times n}$ and $\mu \in \mathbb{R}_+^{m \times 1}$. Here $\mathbb{1}$ is the $1 \times n$ matrix with all entries being 1. The *LP rank of M* denoted by $\mathrm{rank}_{\mathrm{LP}} M$ is the minimum r such that there exists a size-r LP factorization of M.

A size-r LP factorization is equivalent to a decomposition $M = \sum_{i\in[r]} u_i v_i^\intercal + \mu\mathbb{1}$ for some (column) vectors $u_i \in \mathbb{R}^m_+$, $v_i \in \mathbb{R}^n_+$ with $i \in [r]$ and a column vector $\mu \in \mathbb{R}^m_+$. It is a slight modification of a nonnegative matrix factorization, disregarding simultaneous shift of all columns by the same vector, i.e., allowing an additional term $\mu \cdot \mathbb{1}$ *not* contributing to the size, so clearly, $\operatorname{rank}_{\mathrm{LP}} \leq \operatorname{rank}_+ M \leq \operatorname{rank}_{\mathrm{LP}} M + 1$.

One similarly defines SDP factorizations of nonnegative matrices.

Definition 2.1.3. A *size-r SDP factorization* of $M \in \mathbb{R}^{m\times n}_+$ is a factorization is a collection of matrices $T_1, \ldots, T_m \in \mathbb{S}^r_+$ and $U_1, \ldots, U_n \in \mathbb{S}^r_+$ together with $\mu \in \mathbb{R}^{m\times 1}_+$ so that $M_{ij} = \operatorname{Tr}[T_i U_j] + \mu(i)$. The *SDP rank of M* denoted by $\operatorname{rank}_{\mathrm{SDP}} M$ is the minimum r such that there exists a size-r SDP factorization of M.

For the next theorem, we need the folklore formulation of linear duality using affine functions, see e.g., Schrijver (1986, Corollary 7.1h).

Lemma 2.1.4 (Affine form of Farkas's Lemma)**.** *Let $P := \left\{x \,\middle|\, A_j x \leq b_j, j \in [r]\right\}$ be a non-empty polyhedron. An affine function Φ is nonnegative on P if and only if there are nonnegative multipliers λ_j, λ_0 with*

$$\Phi(x) \equiv \lambda_0 + \sum_{j\in[r]} \lambda_j (b_j - A_j x).$$

We are ready for the factorization theorem for optimization problems.

Theorem 2.1.5 (Factorization theorem for formulation complexity)**.** *Consider an optimization problem $\mathcal{P} = (\mathcal{S}, \mathcal{F}, \mathrm{val})$ with (C, S)-approximate slack matrix M. Then we have*

$$\mathrm{fc}_+(\mathcal{P}, C, S) = \operatorname{rank}_{\mathrm{LP}} M \qquad \textit{and} \qquad \mathrm{fc}_\oplus(\mathcal{P}, C, S) = \operatorname{rank}_{\mathrm{SDP}} M.$$

for linear formulations and semidefinite formulations, respectively.

Remark 2.1.6. The factorization theorem for polyhedral pairs (see Braun et al. 2012; Pashkovich 2012; Braun et al. 2014a) states that the nonnegative rank and extension complexity might

differ by 1, which was slightly elusive. Theorem 2.1.5 clarifies, that this is the difference between the LP rank and nonnegative rank, i.e., formulation complexity is a *property of the slack matrix*. Note also that for the slack matrix of a polytope, every row contains a 0 entry, and hence the $\mu\mathbb{1}$ term in any LP factorization must be 0. Therefore the nonnegative rank and LP rank coincide for polytopes. Similar remarks apply to SDP factorizations.

Proof of Theorem 2.1.5—the linear case. We will confine ourselves to the case of $\mathcal{P}$ being a maximization problem. For minimization problems, the proof is analogous.

To prove $\operatorname{rank}_{\mathrm{LP}} M \leq \mathrm{fc}_+(\mathcal{P}, C, S)$, let $Ax \leq b$ be an arbitrary (C, S)-approximate, size-r LP formulation of $\mathcal{P}$, with realizations $\left\{w^f \,\middle|\, f \in \mathcal{F}^S\right\}$ of instances and $\{x^s \mid s \in \mathcal{S}\}$ of feasible solutions. We shall construct a size-r nonnegative factorization of M. As $\max_{x:Ax\leq b} w^f(x) \leq C(f)$ by Condition (1.3), via the affine form of Farkas's lemma, Lemma 2.1.4 we have

$$C(f) - w^f(x) = \sum_{j=1}^{r} T(f,j)\left(b_j - \langle A_j, x\rangle\right) + \mu(f)$$

for some nonnegative multipliers $T(f,j), \mu(f) \in \mathbb{R}_+$ with $1 \leq j \leq r$. By taking $x = x^s$, we obtain

$$M(f,s) = \sum_{j=1}^{r} T(f,j)U(j,s) + \mu(f), \quad \text{with} \quad U(j,s) := b_j - \langle A_j, x^s\rangle \quad \text{for } j > 0. \tag{2.1}$$

i.e., $M = TU + \mu\mathbb{1}$. By construction, T and μ are nonnegative. By Condition (1.1) we also obtain that U is nonnegative. Therefore $M = TU + \mu\mathbb{1}$ is a size-r LP factorization of M.

For the converse, i.e., $\operatorname{rank}_{\mathrm{LP}} \geq \mathrm{fc}_+(\mathcal{P}, C, S)$, let $M = TU + \mu\mathbb{1}$ be a size-r LP factorization. We shall construct an LP formulation of size r. Let T_f denote the f-row of T for $f \in \mathcal{F}^S$, and U_s denote the s-column of U for $s \in \mathcal{S}$. We claim that the linear system

$x \geq 0$ with representations

$$w^f(x) := C(f) - \mu(f) - T_f x \qquad \forall f \in \mathcal{F}^S \qquad \text{and} \qquad x^s := U_s \qquad \forall s \in \mathcal{S}$$

satisfies the requirements of Definition 1.1.5. Condition (1.2) is implied by the factorization $M = TU + \mu\mathbb{1}$:

$$w^f(x^s) = C(f) - \mu(f) - T_f U_s = C(f) - M(f,s) = \mathrm{val}_f(s).$$

Moreover, $x^s \geq 0$, because U is nonnegative, so that Condition (1.1) is fulfilled. Finally, Condition (1.3) also follows readily:

$$\max\left\{w^f(x) \,\middle|\, x \geq 0\right\} = \max\left\{C(f) - \mu(f) - T_f x \,\middle|\, x \geq 0\right\} = C(f) - \mu(f) \leq C(f),$$

as the nonnegativity of T implies $T_f x \geq 0$; equality holds e.g., for $x = 0$. Recall also that $\mu(f) \geq 0$. Thus we have constructed an LP formulation with r inequalities, as claimed. □

Remark 2.1.7. It is counter-intuitive that 0 is always a maximizer, and actually it is an artifact of the construction. At a conceptual level, the polyhedron $Ax \leq b$ containing $\mathrm{conv}\,(x^s \mid s \in \mathcal{S})$ is represented as the intersection of the nonnegative cone with an affine subspace in the slack space. The affine functions w^f are extended to attain their optimum value on this intersection in the nonnegative cone, and thus also at 0, the apex of the cone. In particular, intersecting with the affine subspace is no longer needed. See Figure 2.1 for an illustration.

Remark 2.1.8 (Solution structure). Observe that the obtained LP formulation via the LP-factorization of the slack matrix also separates solutions $s \in S$ into two disjoint classes $S = S_o \cup S_n$, where S_o contains those solutions that potentially can be optimal for some function, i.e., it is the set of coordinate-wise minimal points in S. The set S_n is the set of solution that are *never* optimal for any $f \in F$ as they are coordinate-wise dominated by at

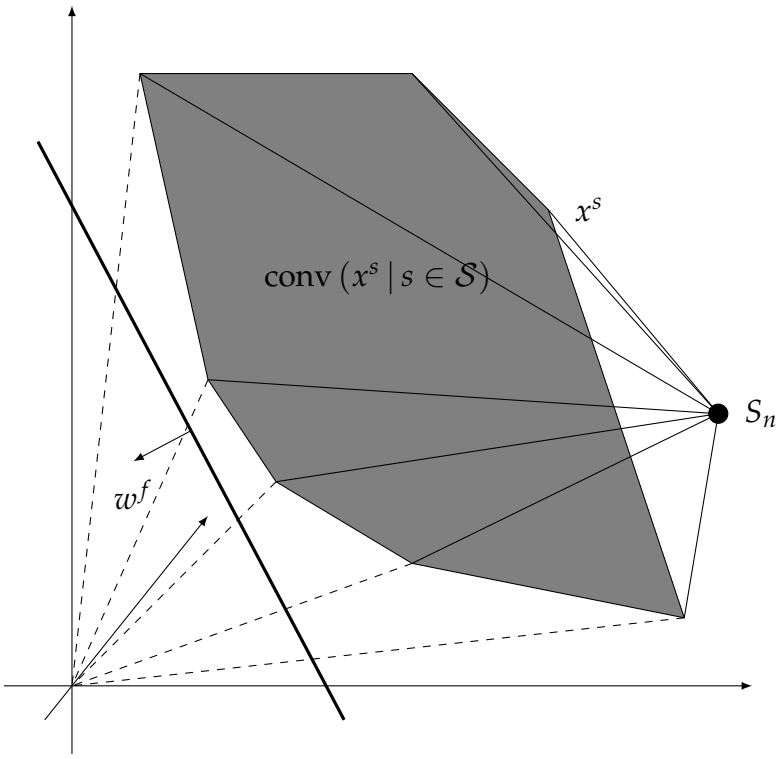

Figure 2.1: Linear program obtained from an LP factorization. The LP is the positive orthant $x \geq 0$. The point 0 is a maximizer for all linearizations $w^f = C(f) - \mu(f) - T_f x$ of objective functions val_f for all instances f. The normals of all objective functions point in nonpositive direction as $T_f \geq 0$.

least one point in S_o, i.e., for any $s_n \in S_n$ there exists $s_o \in S_o$ with $s_o \leq s_n$ coordinate-wise.

This might have applications in the context of inverse optimization (see e.g., Ahuja and Orlin 2001), where we would like to decide whether for a given solution $s \in \mathcal{S}$, there exists an $f \in \mathcal{F}$, that is maximized at s. This can now be read off the factorization.

Proof of Theorem 2.1.5—the semidefinite case. As before we confine ourselves to the case of $\mathcal{P}$ being a maximization problem. The proof is analogous to the linear case, but for the sake of completeness, we provide a full proof.

To prove $\mathrm{rank}_{\mathrm{SDP}} M \leq \mathrm{fc}_{\oplus}(\mathcal{P}, C, S)$, let $\mathcal{A}(X) = b, X \in \mathbb{S}^r_+$ be an arbitrary size-r SDP formulation of $\mathcal{P}$, with realizations $\left\{w^f \;\middle|\; f \in \mathcal{F}^S\right\}$ of instances and $\left\{X^s \mid s \in \mathcal{S}\right\}$ of feasible solutions. To apply strong duality, we may assume that the convex set $\left\{X \in \mathbb{S}^r_+ \mid \mathcal{A}(X) = b\right\}$ has an interior point because otherwise it would be contained in a proper face of $\mathbb{S}^r_+$, which

is an SDP cone of smaller size. We shall construct a size-r SDP factorization of M. As $\max_{X \in \mathbb{S}^r_+ \colon \mathcal{A}(X)=b} w^f(X) \leq C(f)$ by Condition (1.6), via the affine form of strong duality, we have

$$C(f) - w^f(X) = \langle T_f, X \rangle + \langle y^f, b - \mathcal{A}(X) \rangle + \lambda_f$$

for all $f \in \mathcal{F}$ with some $T_f \in \mathbb{S}^r_+$, $y^f \in \mathbb{R}^k$ and $\lambda_f \in \mathbb{R}_+$. By substituting X^s into X, we obtain

$$M(f,s) = C(f) - w^f(X^s) = \langle T_f, X^s \rangle + \langle y^f, b - \mathcal{A}(X^s) \rangle + \lambda_f = \langle T_f, X^s \rangle + \lambda_f, \tag{2.2}$$

which is an SDP factorization of size r.

For the converse, i.e., $\mathrm{fc}_{\oplus}(\mathcal{P}, C, S) \leq \mathrm{rank}_{\mathrm{SDP}}\, M$, let $M(f,s) = \langle T_f, U_s \rangle + \mu(f)$ be a size-r SDP factorization. We shall construct an SDP formulation of size r. We claim that the SDP formulation:

$$X \in \mathbb{S}^r_+ \tag{2.3}$$

with representations

$$w^f(X) \coloneqq C(f) - \mu(f) - \langle T_f, X \rangle \qquad \forall f \in \mathcal{F}^S \quad \text{and} \quad X^s \coloneqq U_s \qquad \forall s \in \mathcal{S}$$

satisfies the requirements of Definition 1.1.7. Condition (1.5) follows by:

$$w^f(X^s) = C(f) - \mu(f) - \langle T_f, U_s \rangle = C(f) - M(f,s) = \mathrm{val}_f(s).$$

Moreover, the $X^s = U_s$ are psd, hence clearly satisfy the system (2.3), so that Condition (1.4) is fulfilled. Finally, Condition (1.6) also follows readily.

$$\max \left\{ w^f(X) \,\middle|\, X \in \mathbb{S}^r_+ \right\} = \max \left\{ C(f) - \mu(f) - \langle T_f, X \rangle \,\middle|\, X \in \mathbb{S}^r_+ \right\} = C(f) - \mu(f) \leq C(f),$$

as T_f and X being psd implies $\langle T_f, X \rangle \geq 0$; equality holds e.g., for $X = 0$. Thus we have constructed an SDP formulation of size r as claimed. □

2.1.1 Examples

In the context of optimization problems we typically differentiate two types of formulations. The *uniform model* asks for a formulation *for a whole family of instances*. Our Examples in this section are all uniform models. The *non-uniform model* asks for a formulation for the weighted version of *a specific problem*, where the instances differ only in the weighting. Lower bounds or inapproximability factors for non-uniform models are usually stronger statements, as in the non-uniform case the formulation potentially could adapt to the instance resulting in potentially smaller formulations; see Bazzi et al. (2015) for such an example in the context of stable sets. We refer the reader to Chan et al. (2013) and Braun, Fiorini, and Pokutta (2016) for an in-depth discussion.

The difference between uniform and non-uniform sometimes depends on the point of view. For graph problems, often the non-uniform model for a graph G induces a uniform model for the family of all of its (induced) subgraphs by choosing 0/1 weights (see e.g., Definition 2.3.1). In other words, the non-uniform model is actually a uniform model for the class of all subinstances of G. Definition 2.1.9 and Example 2.2.12 with $0/1$ weights demonstrate this: they are uniform models for all subgraphs $G \subseteq K_n$, but can also be viewed as non-uniform models for K_n.

Note that these models can also be used for studying average case complexity. For example, one might consider the complexity of the problem for a randomly selected large class of instances using the uniform model, or one might consider the non-uniform model for a randomly selected instance. For the maximum stable set problem, both random versions were examined in Braun, Fiorini, and Pokutta (2016).

The matching problem revisited

The lower bounds in Fiorini et al. (2012) and Rothvoß (2014) are concerned with specific polytopes, namely the TSP polytope as well as the matching polytope. We obtain, as a slight generalization, the same lower bounds for the Hamiltonian cycle problem (which is captured by the TSP problem with appropriate weights) as well as the matching *problem*, independent of choice of the specific polytope that represents the encoding. Here we present only the lower bounds on the linear formulation complexity of the matching problem. The lower bound for the Hamiltonian cycle problem will be obtained in Section 2.2.3 via a reduction. In Chapter 3 we show how to lower bound the size of *symmetric* semidefinite formulations.

The maximum matching problem PM(n) asks for the maximum size of matchings in a given graph. While it can be solved in polynomial time, the matching polytope has exponential extension complexity as shown in Rothvoß (2014). Using the framework from above, we immediately obtain that the matching problem has high LP formulation complexity, reusing the lower bound on the nonnegative rank of the slack matrix of the matching polytope.

We first give the natural formulation of the problem in our framework.

Definition 2.1.9 (PM(n))**.** Let n be fixed and even, then the set $\mathcal{S}$ of feasible solutions of the perfect matching problem PM(n) consists of all perfect matchings M of K_n, the complete graph on n vertices, and the instances are all (simple) graphs G on $[n]$. The value $\mathrm{val}_G(M)$ for a graph G and a perfect matching M is defined to be

$$\mathrm{val}_G(M) := |M \cap E(G)|$$

the number of edges shared by M and G, i.e., the size of the matching $M \cap E(G)$ of G.

Clearly, all maximum matchings of a graph G on n vertices can be obtained by some perfect matching on K_n (via extension to any perfect matching on $[n]$). Thus $\max \mathrm{val}_G$ is the matching number $\nu(G)$ of G, the size of the maximum matchings of G.

Inspired by the description of the facets of the matching polytope in Edmonds (1965), we only consider complete subgraphs on odd-sized subsets U. For such a complete graph K_U on the odd-sized set U, we have $\max \mathrm{val}_{K_U} = \frac{|U|-1}{2}$. Let $\delta(U)$ denote the set of all edges between U and its complement $[2n] \setminus U$. We have the identity $|U| = 2\,|M \cap E(K_U)| + |M \cap \delta(U)|$ and thus obtain the slack matrix for the exact problem (i.e., $C(G) = \max \mathrm{val}_G$)

$$S(K_U, M) := C(K_U) - \mathrm{val}_{K_U} = \frac{|U|-1}{2} - |M \cap E(K_U)| = \frac{|M \cap \delta(U)| - 1}{2}.$$

This submatrix has nonnegative rank $2^{\Omega(n)}$ by Rothvoß (2014) and hence the LP formulation complexity of the maximum matching problem is $2^{\Omega(n)}$, i.e., $\mathrm{fc}_+(\mathrm{PM}(n)) = 2^{\Omega(n)}$.

The result can be extended to the approximate case with an approximation factor $(1 + \varepsilon/n)^{-1}$, by invoking the lower bound for the resulting slack matrix from Braun and Pokutta (2015a), showing that the maximum matching problem does not admit any fully-polynomial size relaxation scheme. The approximation guarantees in Braun and Pokutta (2015a) are chosen as $C(f) = \max f + \varepsilon/2$ and $S(G) = \max f$.

Theorem 2.1.10 (Theorem 3.1, Braun and Pokutta 2015a)**.** *Let $\varepsilon \in (0, 1)$ be a fixed number and $n \in \mathbb{N}$ even. Then $\mathrm{fc}_+(\mathrm{PM}(n), \max f + \varepsilon/2, \max f) = 2^{\Omega(n)}$, resulting in an inapproximability factor of $(1 + \varepsilon/n)^{-1} \leq 1 - \varepsilon/n$.*

The inapproximability factor follows from $\max f \leq n/2$:

$$\frac{S(f)}{C(f)} = \frac{\max f}{\max f + \varepsilon/2} = \frac{1}{1 + \frac{\varepsilon/2}{\max f}} \leq \frac{1}{1 + \frac{\varepsilon/2}{n/2}} = (1 + \varepsilon/n)^{-1}.$$

Note that this result is not unexpected as for the maximum matching problem an approximation factor of about $1 - \frac{\varepsilon}{n}$ corresponds to an error of less than one edge in the unweighted case for small ε, so that the decision problem could be decided via the approximation. This is a behavior similar to FPTAS and strong NP-hardness for combinatorial optimization problems that are mutually exclusive (under standard assumptions).

Independent set problem

We provide an example for maximum independent sets in a uniform model; see Braun, Fiorini, and Pokutta (2016) for more details as well as an average case analysis. Here there is no bound on the maximum degree of graphs, unlike in Theorem 2.3.3.

Example 2.1.11 (Maximum independent set problem (uniform model)). Let us consider the maximum independent set problem $\mathcal{P}$ over some family $\mathcal{G}$ of graphs G where $V(G) \subseteq [n]$ with aim to estimate the maximum size $\alpha(G)$ of independent sets in each $G \in \mathcal{G}$.

A natural choice is to let the *feasible solutions* be all subsets S of $[n]$, and the *instances* be all $G \in \mathcal{G}$. The objective function is

$$\mathrm{val}_G(S) \coloneqq |V(G) \cap S| - |E(G(S))| .$$

Here $\mathrm{val}_G(S)$ can be easily seen to lower bound the size of an independent set, obtained from S by removing vertices not in G, and also removing one end point of every edge with both end points in S. Clearly, $\mathrm{val}_G(S) = |S|$ for independent sets S of G, i.e., in this case our choice is exact. Thus $\alpha(G) = \max_{S \subseteq [n]} \mathrm{val}_G(S)$.

Let us consider the special case when $\mathcal{G}$ is the set of all simple graphs with $V(G) \subseteq [n]$. We shall use guarantees $S(G) \coloneqq \max \mathrm{val}_G = \alpha(G)$ and $C(G) \coloneqq \rho^{-1} \max \mathrm{val}_G$ for an approximation factor $0 < \rho \leq 1$. Restricting to complete graphs K_U with $U \subseteq [n]$, the obtained slack matrix is a $(\rho^{-1} - 1)$-shift of the (partial) unique disjointness matrix, hence for approximations within a factor of ρ, we obtain the lower bound on the linear formulation complexity $\mathrm{fc}_+(\mathcal{P}, \rho^{-1} \max \mathrm{val}_G, \max \mathrm{val}_G) \geq 2^{\frac{n\rho}{8}}$ with Braverman and Moitra (2013) and Braun and Pokutta (2016).

See Braun, Fiorini, and Pokutta (2016) for other choices of $\mathcal{G}$, such as e.g., randomly choosing the graphs.

k-juntas via LPs

It is well-known that the level-k Sherali–Adams hierarchy captures all nonnegative k-juntas, i.e., functions $f\colon \{0,1\}^n \to \mathbb{R}_+$ that depend only on k coordinates of the input (see e.g., Chan et al. 2013) and it can be written as a linear program using $O(n^k)$ inequalities. We will now show that this is essentially optimal for k small.

Example 2.1.12 (k-juntas). We consider the problem of maximizing nonnegative k-juntas over the n-dimensional hypercube. Let the set of instances $\mathcal{F}$ be the family of all nonnegative k-juntas and let the set of feasible solutions be $\mathcal{S} = \{0,1\}^n$, with $\operatorname{val}_f(s) \coloneqq \operatorname{val}_f(s)$. We set $C(f) = S(f) = \max_{s\in\mathcal{S}} \operatorname{val}_f(s)$.

As we are interested in a lower bound we will confine ourselves to a specific subfamily of functions $\mathcal{F}' \coloneqq \{f_a \mid a \in \{0,1\}^n, |a| = k\} \subseteq \mathcal{F}$ with

$$f_a(b) \coloneqq a^\intercal b - 2\binom{a^\intercal b}{2},$$

and hence $C(f_a) = 1$. Clearly $|\mathcal{F}'| = \binom{n}{k}$, so that the nonnegative rank of the slack matrix

$$S_{a,b} \coloneqq C(f_a) - f_a(b) = 1 - a^\intercal b + 2\binom{a^\intercal b}{2} = (1 - a^\intercal b)^2,$$

with $a, b \in \{0,1\}^n$ and $|a| = k$ is at most $\binom{n}{k}$.

Now for each $f_a \in \mathcal{F}'$ we have that

$$C(f_a) - f_a(b) = (1 - a^\intercal b)^2 = 1 \qquad \text{if } a \cap b = \emptyset$$

and there are 2^{n-k} such choices for b for a given a. Thus the matrix S has $\binom{n}{k}2^{n-k}$ entries 1 arising from disjoint pairs a, b. However in Kaibel and Weltge (2015) it was shown that any nonnegative rank-1 matrix can cover at most 2^n of such pairs. Thus the nonnegative rank of

S is at least

$$\frac{\binom{n}{k}2^{n-k}}{2^n} = \frac{\binom{n}{k}}{2^k}.$$

The latter is $\Omega(n^k)$ for k constant and at least $\Omega(n^{k-\alpha})$ for $k = \alpha \log n$ with $\alpha \in \mathbb{N}$ constant and $k > \alpha$. Thus the LP formulation for k-juntas derived from the level-k Sherali-Adams hierarchy is essentially optimal for small k.

2.2 Affine Reductions for LPs and SDPs

We will now introduce natural reductions between problems, with control on approximation guarantees that translate to the underlying LP and SDP level.

Definition 2.2.1 (Reductions between problems)**.** Let $\mathcal{P}_1 = (\mathcal{S}_1, \mathcal{F}_1, \mathrm{val})$ and $\mathcal{P}_2 = (\mathcal{S}_2, \mathcal{F}_2, \mathrm{val})$ be maximization problems. Let C_1, S_1 and C_2, S_2 be guarantees for $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively. A *reduction* from $\mathcal{P}_1$ to $\mathcal{P}_2$ respecting these guarantees consist of two maps:

(i) $\beta\colon \mathcal{F}_1^{S_1} \to \mathrm{cone}\left(\mathcal{F}_2^{S_2}\right) + \mathbb{R}$ rewriting instances as formal nonnegative combinations: $\beta(f_1) \coloneqq \sum_{f \in \mathcal{F}_2^{S_2}} b_{f_1,f} \cdot f + \mu(f_1)$ with $b_{f_1,f} \geq 0$ for all $f \in \mathcal{F}_2^{S_2}$; the term $\mu(f_1)$ is called the *affine shift*

(ii) $\gamma\colon \mathcal{S}_1 \to \mathrm{conv}\,(\mathcal{S}_2)$ rewriting solutions as formal convex combination of $\mathcal{S}_2$: $\gamma(s_1) \coloneqq \sum_{s \in \mathcal{S}_2} a_{s_1,s} \cdot s$ with $a_{s_1,s} \geq 0$ for all $s \in \mathcal{S}_2$ and $\sum_{s \in \mathcal{S}_2} a_{s_1,s} = 1$;

subject to

$$\mathrm{val}_{f_1}(s_1) = \sum_{\substack{f \in \mathcal{F}_2^{S_2} \\ s \in \mathcal{S}_2}} b_{f_1,f} a_{s_1,s} \cdot \mathrm{val}_f(s) + \mu(f_1), \qquad s_1 \in \mathcal{S}_1,\ f_1 \in \mathcal{F}_1^{S_1}, \tag{2.4}$$

expressing representation of the objective function of $\mathcal{P}_1$ by that of $\mathcal{P}_2$, and additionally

$$C_1(f_1) \geq \sum_{f \in \mathcal{F}_2^{S_2}} b_{f_1,f} \cdot C_2(f) + \mu(f_1), \qquad f_1 \in \mathcal{F}_1^{S_1}, \tag{2.5}$$

ensuring feasibility of the completeness guarantee.

Observe that the role of soundness guarantees of $\mathcal{P}_1$ and $\mathcal{P}_2$ in the definition is to restrict the instances considered: the map β involves only the instances whose optimum value is bounded by these guarantees. One can analogously define reductions involving minimization problems. E.g., for a reduction from a maximization problem $\mathcal{P}_1$ to a minimization problem $\mathcal{P}_2$, the formulas are

$$\begin{aligned}
\beta(f_1) &\coloneqq \mu(f_1) - \sum_{f \in \mathcal{F}_2^{S_2}} b_{f_1,f} \cdot f \\
\mathrm{val}_{f_1}(s_1) &= \mu(f_1) - \sum_{\substack{f \in \mathcal{F}_2^{S_2} \\ s \in \mathcal{S}_2}} b_{f_1,f} a_{s_1,s} \cdot \mathrm{val}_f(s) \\
C_1(f_1) &\geq \mu(f_1) - \sum_{\mathrm{val}_f \in \mathcal{F}_2^{S_2}} b_{f_1,f} \cdot C(f).
\end{aligned}$$

Note that elements in $\mathcal{S}_2$ are obtained as convex combinations, while elements in $\mathcal{F}_2$ are obtained as nonnegative combinations and a shift. The additional freedom for instances allows scaling and shifting the function values.

In a first step we will verify that a reduction between optimization problems $\mathcal{P}_1$ to $\mathcal{P}_2$ naturally extends to potential LP and SDP formulations.

Proposition 2.2.2 (Reductions of formulations)**.** *Consider a reduction from an optimization problem $\mathcal{P}_1$ to another one $\mathcal{P}_2$ respecting completion and soundness guarantees C_1, S_1 and C_2, S_2. Then* $\mathrm{fc}_+(\mathcal{P}_1, C_1, S_1) \leq \mathrm{fc}_+(\mathcal{P}_2, C_2, S_2)$ *and* $\mathrm{fc}_\oplus(\mathcal{P}_1, C_1, S_1) \leq \mathrm{fc}_\oplus(\mathcal{P}_2, C_2, S_2)$.

Proof. We will use the notation from Definition 2.2.1 for the reduction. We only prove the claim for LP formulations and for two maximization problems, as the proof is analogous

for SDP formulations and when either or both problems are minimization problems. Let us choose an LP formulation $Ax \leq b$ of $\mathcal{P}_2$ with x^s realizing $s \in \mathcal{S}_2$ and w^f realizing $f \in \mathcal{F}_2^{\mathcal{S}_2}$. For $\mathcal{P}_1$ we shall use the same linear program $Ax \leq b$ with the following realizations y^{s_1} of feasible solutions $s_1 \in \mathcal{S}_1$, and u^{f_1} of instances $f_1 \in \mathcal{F}_1^{\mathcal{S}_1}$, where

$$y^{s_1} \coloneqq \sum_{s \in \mathcal{S}_2} a_{s_1,s} \cdot x^s, \qquad u^{f_1}(x) \coloneqq \sum_{f \in \mathcal{F}_2^{\mathcal{S}_2}} b_{f_1,f} \cdot w^f(x) + \mu(f_1).$$

As y^{s_1} is a convex combination of the x^s, obviously $Ay^{s_1} \leq b$. The u^{f_1} are clearly affine functions with

$$\begin{aligned} u^{f_1}(y^{s_1}) &= \sum_{f \in \mathcal{F}_2^{\mathcal{S}_2}} b_{f_1,f} \cdot w^f \left(\sum_{s \in \mathcal{S}_2} a_{s_1,s} \cdot x^s \right) + \mu(f_1) \\ &= \sum_{f \in \mathcal{F}_2^{\mathcal{S}_2}} b_{f_1,f} \sum_{s \in \mathcal{S}_2} a_{s_1,s} \cdot w^f(x^s) + \mu(f_1) \\ &= \mathrm{val}_{f_1}(s_1) \end{aligned}$$

by Eq. (2.4). Moreover, by Eq. (2.5).

$$\max_{Ax \leq b} u^{f_1}(x) \leq \sum_{f \in \mathcal{F}_2^{\mathcal{S}_2}} b_{f_1,f} \cdot \max_{Ax \leq b} w^f(x) + \mu(f_1) \leq \sum_{f \in \mathcal{F}_2^{\mathcal{S}_2}} b_{f_1,f} \cdot C_2(f) + \mu(f_1) \leq C_1(f_1).$$

□

Remark 2.2.3. At the level of matrices, Proposition 2.2.2 can be equivalently formulated as follows. Whenever $M_1 = R \cdot M_2 \cdot C + t\mathbb{1}$ with M_1, M_2, R, C nonnegative matrices, and t a nonnegative vector, such that $\mathbb{1}C = \mathbb{1}$, then $\mathrm{rank}_{\mathrm{LP}} M_1 \leq \mathrm{rank}_{\mathrm{LP}} M_2$ and $\mathrm{rank}_{\mathrm{SDP}} M_1 \leq \mathrm{rank}_{\mathrm{SDP}} M_2$. Note that given a reduction of $\mathcal{P}_1$ to $\mathcal{P}_2$ with the notation as in Definition 2.2.1, one chooses M_1 and M_2 to be the slack matrices of $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively, together with

matrices R, C and a vector t with the following entries:

$$R(f_1, f) = b_{f_1,f}, \tag{2.6}$$

$$C(s, s_1) = a_{s_1,s}, \tag{2.7}$$

$$t(f) = C_2(f_1) + \mu(f_1) - \sum_{f \in \mathcal{F}_2} b_{f_1,f} \cdot C(f), \tag{2.8}$$

all nonnegative, satisfying $M_1 = R \cdot M_2 \cdot C + t\mathbb{1}$. Now we briefly indicate a proof for this alternative formulation of Proposition 2.2.2.

First, we prove the LP case. Given a size-r LP factorization $M_2 = \sum_{i \in [r]} u_i v_i + \mu\mathbb{1}$ of M_2, one gets the size-r LP factorization $M_1 = \sum_{i \in [r]} Ru_i \cdot v_i C + (R\mu + t)\mathbb{1}$ of M_1.

For the SDP case, let $M_2(i, j) = \mathrm{Tr}[T_i U_j] + \mu(i)$ be an SDP factorization of size r. Then one can construct the following SDP factorization of M_1 of size r:

$$M_1(f, s) = \sum_{i,j} R(f, i) M_2(i, j) C(j, s) + t(f) = \sum_{i,j} R(f, i)(\mathrm{Tr}[T_i U_j] + \mu(i)) C(j, s) + t(f)$$

$$= \mathrm{Tr}\left[\underbrace{\left(\sum_i R(f, i) T_i \right)}_{\widehat{T}_f} \cdot \underbrace{\left(\sum_j U_j C(j, s) \right)}_{\widehat{U}_s} \right] + \underbrace{\left(\sum_i R(f, i)\mu(i) + t(f) \right)}_{\widehat{\mu}(f)},$$

using $\sum_j C(j, s) = 1$, i.e., $\mathbb{1}C = \mathbb{1}$. Using the nonnegativity of R, C, μ, and t, the $\widehat{T}_f$ and $\widehat{U}_s$ are psd, and the $\widehat{\mu}(f)$ are nonnegative.

2.2.1 Simpler Reductions

In many cases reductions as in Definition 2.2.1 are too general. In this chapter we present special cases of reductions which are powerful enough in most cases however much simpler to apply. The first modification we are going to make concerns the ability to consider linear combinations in reductions. Throughout the whole chapter we consider reductions of the form $\beta \colon \mathcal{F}_1 \to \mathcal{F}_2$ and $\gamma \colon \mathcal{S}_1 \to \mathcal{S}_2$, leading to a simpler version of condition (2.4) given by

$\mathrm{val}_{\beta(f_1)}[\gamma(s_1)] = \alpha\,\mathrm{val}_{f_1}(s_1) + \mu(f_1)$, where we rearranged the equation for convenience. If additionally to the requirement (2.4) optimal values are mapped to optimal values we call a reduction exact, which is captured in Definition 2.2.4.

Definition 2.2.4. A reduction consisting of the maps $\beta\colon \mathcal{F}_1 \to \mathcal{F}_2$ and $\gamma\colon \mathcal{S}_1 \to \mathcal{S}_2$ is called *exact*, if we have

$$\mathrm{opt}_{\mathcal{P}_2}\,\mathrm{val}_{\beta(f_1)} = \alpha\,\mathrm{opt}_{\mathcal{P}_1}\,\mathrm{val}_{f_1} + \mu(f_1),$$

where the operator $\mathrm{opt}_{\mathcal{P}_1}$ is max when $\mathcal{P}_1$ is a maximization problem, and the operator $\mathrm{opt}_{\mathcal{P}_1}$ is min when $\mathcal{P}_1$ is a minimization problem. The operator $\mathrm{opt}_{\mathcal{P}_2}$ is defined similarly for $\mathcal{P}_2$.

The first special case we explicitly state is concerned with problems having exact completeness and soundness guarantees. That means if we are only interested in exact formulations as opposed to approximation factors we can use Corollary 2.2.5.

Corollary 2.2.5 (Exact reduction between exact problems)**.** *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two optimization problems with exact completeness and soundness guarantees, i.e., $C_1(f_1) = S_1(f_1) = \mathrm{opt}_{\mathcal{P}_1}\,\mathrm{val}_{f_1}$ and $C_2(f_2) = S_2(f_2) = \mathrm{opt}_{\mathcal{P}_2}\,\mathrm{val}_{f_2}$. Further let $\gamma\colon \mathcal{S}_1 \to \mathcal{S}_2$ and $\beta\colon \mathcal{F}_1^{S_1} \to \mathcal{F}_2$ be maps with*

$$\mathrm{val}_{\beta(f_1)}[\gamma(s_1)] = \alpha\,\mathrm{val}_{f_1}(s_1) + \mu(f_1)$$

and

$$\mathrm{opt}_{\mathcal{P}_2}\,\mathrm{val}_{\beta(f_1)} = \alpha\,\mathrm{opt}_{\mathcal{P}_1}\,\mathrm{val}_{f_1} + \mu(f_1).$$

Then γ and β form an exact reduction and in particular $\mathrm{fc}_+(\mathcal{P}_2) \geq \mathrm{fc}_+(\mathcal{P}_1)$ and $\mathrm{fc}_\oplus(\mathcal{P}_2) \geq \mathrm{fc}_\oplus(\mathcal{P}_1)$ holds, i.e., there is no exact linear or SDP formulation of $\mathcal{P}_2$ of size less than $\mathrm{fc}_+(\mathcal{P}_1)$ or $\mathrm{fc}_\oplus(\mathcal{P}_1)$ respectively.

Proof. We have to verify Equations (2.4) and (2.5). Equation (2.4) follows directly from $\operatorname{val}_{\beta(f_1)}[\gamma(s_1)] = \alpha \operatorname{val}_{f_1}(s_1) + \mu(f_1)$ satisfied by γ and β. Equation (2.5) is a direct consequence from $\operatorname{opt}_{\mathcal{P}_2} \operatorname{val}_{\beta(f_1)} = \alpha \operatorname{opt}_{\mathcal{P}_1} \operatorname{val}_{f_1} + \mu(f_1)$ and the exactness of the completeness and soundness guarantees of $\mathcal{P}_1$ and $\mathcal{P}_2$. □

If we are interested in approximate formulations, we observe that for most problems, there is a natural *size* $|f|$ of an instance f, which is a nonnegative number, and guarantees are often proportional to it.

Corollary 2.2.6 (Inapproximability reduction with relative guarantees)**.** *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two optimization problems. Let $\mathcal{P}_1$ the completeness guarantee of $\mathcal{P}_1$ have the form*

$$C_1(f) = \tau_1 |f|, \qquad f \in \mathcal{F}_1$$

proportional to the size $|f|$ of each instance f with $|f| \geq 0$. Furthermore, let $\gamma\colon \mathcal{S}_1 \to \mathcal{S}_2$ and $\beta\colon \mathcal{F}_1^{S_1} \to \mathcal{F}_2$ be maps satisfying for some constants α, μ and η

$$\operatorname{val}_{\beta(f_1)}[\gamma(s_1)] = \alpha \operatorname{val}_{f_1}(s_1) + \mu |f_1|$$
$$|\beta(f_1)| = \eta \cdot |f_1|,$$

where $\alpha > 0$ if $\mathcal{P}_2$ is a maximization problem, and $\alpha < 0$ if $\mathcal{P}_2$ is a minimization problem. Let the completeness guarantee C_2 of $\mathcal{P}_2$ be given as

$$C_2(f) := \frac{\alpha\tau_1 + \mu}{\eta} |f| \qquad f \in \mathcal{F}_2.$$

Furthermore let σ_2 be a nonnegative number satisfying for all $f_1 \in \mathcal{F}_1^{S_1}$

$$\max \operatorname{val}_{\beta(f_1)} \leq \sigma_2 |\beta(f_1)| \qquad \text{if } \mathcal{P}_2 \text{ is a maximization problem}$$
$$\min \operatorname{val}_{\beta(f_1)} \geq \sigma_2 |\beta(f_1)| \qquad \text{if } \mathcal{P}_2 \text{ is a minimization problem}$$

and we set

$$S_2(f) = \sigma_2 |f| \qquad\qquad f \in \mathcal{F}_2$$

Then β and γ form a reduction from $\mathcal{P}_1$ with guarantees C_1, S_1 to $\mathcal{P}_2$ with guarantees C_2, S_2. In particular, $\mathcal{P}_2$ is inapproximable within a factor of $\sigma_2\eta/(\alpha\tau_1 + \mu)$ by LP and SDP formulations of size less than $\mathrm{fc}_+(\mathcal{P}_1, C_1, S_1)$ and $\mathrm{fc}_\oplus(\mathcal{P}_1, C_1, S_1)$, respectively.

Proof. Equation (2.4) follows directly from $\mathrm{val}_{\beta(f_1)}[\gamma(s_1)] = \alpha\,\mathrm{val}_{f_1}(s_1) + \mu |f_1|$. Using the parameters as given by the same relation to rewrite Equation (2.5), we get

$$C_1(f_1) \geq \frac{1}{\alpha} C_2(\beta(f_1)) - \frac{\mu |f_1|}{\alpha}.$$

Plugging in the definition of C_2 and the relation between the sizes of the two problems, $|\beta(f_1)| = \eta \cdot |f_1|$, we get

$$C_1(f_1) \geq \frac{\alpha\tau_1 + \mu}{\alpha\eta} |\beta(f_1)| - \frac{\mu |f_1|}{\alpha} = \frac{\alpha\tau_1 + \mu}{\alpha\eta} \eta |f_1| - \frac{\mu |f_1|}{\alpha} = \tau_1 |f_1|,$$

showing that Equation (2.5) holds. □

In many cases also the soundness guarantee of $\mathcal{P}_1$ is proportional to the size of f, i.e.,

$$S_1(f) = \sigma_1 |f|, \qquad\qquad f \in \mathcal{F}_1.$$

Then a common choice is $\sigma_2 = (\alpha\sigma_1 + \mu)/\eta$, provided that the reduction is exact. We shall write $\mathrm{fc}_+(\mathcal{P}, \tau, \sigma)$ for $\mathrm{fc}_+(\mathcal{P}, C, S)$ with $C(f) = \tau |f|$ and $S(f) = \sigma |f|$ and analogously for $\mathrm{fc}_\oplus$. In this case the approximation factor is calculated as

$$\frac{S_2}{C_2} = \frac{\alpha\sigma_1 + \mu}{\alpha\tau_1 + \mu}.$$

Finally we show the special case of a reduction, when the affine shift is 0. In this case the approximation guarantee is preserved without change.

Corollary 2.2.7 (Reductions without affine shift)**.** *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two optimization problems, where $\mathcal{P}_1$ has completeness guarantee C_1 and soundness guarantee S_1. Let $\gamma\colon \mathcal{S}_1 \to \mathcal{S}_2$ and $\beta\colon \mathcal{F}_1 \to \mathcal{F}_2$ be maps satisfying for some constant α*

$$\operatorname{val}_{\beta(f_1)}[\gamma(s_1)] = \alpha \operatorname{val}_{f_1}(s_1) \tag{2.9}$$

and

$$\operatorname{opt}_{\mathcal{P}_2} \operatorname{val}_{\beta(f_1)}[\gamma(s_1)] = \alpha \operatorname{opt}_{\mathcal{P}_1} \operatorname{val}_{f_1}(s_1). \tag{2.10}$$

By setting $C_2(\beta(f_1)) \coloneqq \alpha C_1(f_1)$ and $S_2(\beta(f_1)) \coloneqq \alpha S_1(f_1)$, γ and $\beta|_{\mathcal{F}^{S_1}}$ form an exact reduction. In particular $\mathcal{P}_1$ and $\mathcal{P}_2$ have the same inapproximability guarantee.

Proof. Equation (2.4) follows directly from $\operatorname{val}_{\beta(f_1)}[\gamma(s_1)] = \alpha \operatorname{val}_{f_1}(s_1)$. We have to verify Equation (2.5):

$$C_1(f_1) \geq \frac{1}{\alpha} C_2(\beta(f_1)) = \frac{1}{\alpha} \alpha C_1(f_1) = C_1(f_1).$$

From $\operatorname{opt}_{\mathcal{P}_2} \operatorname{val}_{\beta(f_1)}[\gamma(s_1)] = \alpha \operatorname{opt}_{\mathcal{P}_1} \operatorname{val}_{f_1}(s_1)$ it follows that $\beta|_{\mathcal{F}^{S_1}}$ is a map from $\mathcal{F}^{S_1}$ to $\mathcal{F}^{S_2}$. Thus γ and β form an exact reduction.

□

2.2.2 Base hardness results for MaxCUT and Max-k-XOR

The base problems $\mathcal{P}_1$ from which we reduce will be MaxCUT, MaxCUT$_\Delta$ or Max-k-XOR, which are all CSPs, as well as Max-Matching. For CSPs, the size of an instance, i.e., weighting $(w_1, \ldots, w_m)$ is the total weight $\sum_{i \in [m]} w_i$ of all clauses, For 0/1 weightings representing a

subset L of clauses, the size is just the number of elements of L.

For better readability we summarize the hardness results for MaxCUT and Max-k-XOR in Table 2.1 and for MaxCUT_Δ in Table 2.2. The hardness result for Max-Matching was shown in Theorem 2.1.10. Observe that the problems in this section play the same role as e.g., Max-3-XOR in Håstad's PCP theorem (see Håstad 2001).

Table 2.1: of the hardness results for MaxCUT and Max-k-XOR for $k \geq 2$. Indeed the hardness results for MaxCUT and Max-k-XOR are the same, since MaxCUT is a subproblem of Max-k-XOR for all $k \geq 2$ and this is how we establish the hardness of Max-k-XOR.

	completeness	soundness	inapprox factor	size	source
fc_+	$1-\varepsilon$	$1/2+\varepsilon$	$1/2+\Theta(\varepsilon)$	$2^{n^{c(\varepsilon)}}$	Thm. 2.2.8
$\text{fc}_\oplus$	exact case		1	$2^{\Omega\left(n^{2/13}\right)}$	Thm. 2.2.9
	$4/5-\varepsilon$	$3/4+\varepsilon$	$15/16+\Theta(\varepsilon)$	$n^{\Omega(\log n/\log\log n)}$	Thm. 2.2.10
	$1-\varepsilon$	$c_{GW}+\varepsilon$	$c_{GW}+\Theta(\varepsilon)$	superpolynomial	Conj 2.2.11

Table 2.2: Summary of the hardness results for the bounded degree case MaxCUT_Δ.

	completeness	soundness	inapprox factor	size	source
fc_+	$1-\varepsilon$	$1/2+\varepsilon$	$1/2+\Theta(\varepsilon)$	$2^{n^{c(\varepsilon)}}$	Thm. 2.2.8
$\text{fc}_\oplus$	$1-\varepsilon$	$c_{GW}+\varepsilon$	$c_{GW}+\Theta(\varepsilon)$	superpolynomial	Conj 2.2.11

Theorem 2.2.8 (Kothari, Meka, and Raghavendra 2016, Corollary 1.3)**.** *For every $\varepsilon > 0$ there exists a constant $c(\varepsilon)$ such that for every $k \geq 2$, we have $\text{fc}_+(\text{Max-}k\text{-XOR}, 1-\varepsilon, 1/2+\varepsilon)$ and $\text{fc}_+(\text{MaxCUT}, 1-\varepsilon, 1/2+\varepsilon)$ are both at least $2^{n^{c(\varepsilon)}}$ for infinitely many n, resulting in an inapproximability factor of $1/2+\Theta(\varepsilon)$. Moreover, for the bounded degree case we have $\text{fc}_+(\text{MaxCUT}_\Delta, 1-\varepsilon, 1/2+\varepsilon) = 2^{n^{c(\varepsilon)}}$ for infinitely many n, where Δ is large enough depending on ε and therefore the same inapproximability factor.*

Proof. We first show the hardness result for MaxCUT. Although not explicitly stated in Corollary 1.3 of Kothari, Meka, and Raghavendra (2016) the statement shown is that there exists a constant $c(\varepsilon)$ so that there is no LP relaxation of size less than $2^{n^{c(\varepsilon)}}$ that achieves a

completeness guarantee of $1-\varepsilon$ and a soundness guarantee of $1/2+\varepsilon$, which is a reformulation of our statement. Since the argument ultimately goes back to the proof of Theorem 5.3(I) in Charikar, Makarychev, and Makarychev (2009) and that construction only uses bounded degree graphs, where the bound Δ depends on ε but not on n, it follows that we get the same statement for MaxCUT_Δ.

The result for Max-k-XOR follows, since MaxCUT is a subproblem of Max-2-XOR and therefore of Max-k-XOR for $k \geq 2$.

□

In the SDP case we have the following results.

Theorem 2.2.9. *For the exact semidefinite formulation complexity we have* $\mathrm{fc}_\oplus(\mathsf{MaxCUT}) = 2^{\Omega\left(n^{2/13}\right)}$ *for infinitely many n and for* $k \geq 2$ *that* $\mathrm{fc}_\oplus(\text{Max-}k\text{-XOR}) = 2^{\Omega\left(n^{2/13}\right)}$.

Proof. We use a result by Lee, Raghavendra, and Steurer (2014) stating that the slack matrix of the cut polytope has PSD rank at least $2^{\Omega\left(n^{2/13}\right)}$. Together with the factorization theorem (Theorem 2.1.5) we get the result for MaxCUT.

The result for Max-k-XOR follows since MaxCUT is a subproblem of Max-k-XOR for each $k \geq 2$. □

In the approximate case we can use a result by Braun, Pokutta, and Roy (2016).

Theorem 2.2.10 (Braun, Pokutta, and Roy 2016, Theorem 7.1)**.** *For every* $\varepsilon > 0$ *and* $k \geq 2$ *there are infinitely many n such that* $\mathrm{fc}_\oplus(\mathsf{MaxCUT}, 4/5-\varepsilon, 3/4+\varepsilon) = n^{\Omega(\log n/\log\log n)}$ *and* $\mathrm{fc}_\oplus(\text{Max-}k\text{-XOR}, 4/5-\varepsilon, 3/4+\varepsilon) = n^{\Omega(\log n/\log\log n)}$*, resulting in inapproximability factors of* $15/16+\Theta(\varepsilon)$.

Recall from Khot et al. (2007) that under the Unique Games Conjecture, MaxCUT cannot be approximated better than c_{GW} by a polynomial-time algorithm. This motivates the following conjecture, which provides the SDP-hard base problem with the strongest approximation guarantee.

Conjecture 2.2.11 (SDP inapproximability of MaxCUT)**.** *For every $\varepsilon > 0$, and for every constant Δ large enough depending on ε, the formulation complexity* $\mathrm{fc}_{\oplus}(\mathsf{MaxCUT}_{\Delta}, 1-\varepsilon, c_{GW}+\varepsilon)$ *of* MaxCUT *is superpolynomial and thus inapproximable within a factor of* $c_{GW} + \Theta(\varepsilon)$.

Note however that for fixed Δ, there are algorithms achieving an approximation factor of $c_{GW} + \varepsilon$ by Feige, Karpinski, and Langberg (2002), hence in the conjecture Δ should go to infinity as ε tends to 0.

Finally, we remark that by Karloff (1999, Lemma 2.9) there are graphs G where the Goemans–Williamson SDP is off by a factor of $c_{GW} + \varepsilon$. For simplicity of calculations we assume for the conjecture that there are also such graphs with SDP optimum $(1-\varepsilon)\,|E(G)|$.

2.2.3 Facial reductions and formulation complexity

As the notion of formulation complexity does not directly deal with polytopes, there is no direct translation of monotonicity of extension complexity under faces and projections (see Fiorini et al. 2012). Thus many reductions that have been used in the context of extension complexity and polytopes do not apply, such as e.g., the one from TSP to matching in Yannakakis (1988) and Yannakakis (1991). Often however, a reduction between the problems as defined in Definition 2.2.1 underlies the facial reduction. To exemplify this we provide the underlying reduction from matching to TSP.

Example 2.2.12 (Maximum weight Hamiltonian cycles (uniform model)). We want to find a Hamiltonian cycle with maximum weight in a weighted graph. We consider only nonnegative weights as customary.

Therefore for a fixed n, we choose the *feasible solutions* to be all Hamiltonian cycles C of the complete graph K_n on $[n]$, and the *instances* are weighted subgraphs G of K_n with

nonnegative weights. The objective function has the form

$$\mathrm{val}_G(C) := \sum_{e \in C \cap E(G)} w_e.$$

We shall consider the exact problem, i.e., with guarantees $C(G) = S(G) = \max \mathrm{val}_G$.

In order to have a finite family of instances, one could restrict the weights to e.g., 1, essentially asking for the maximum number of edges a Hamiltonian cycle can have in common with a given subgraph. For the following reduction, we will use weights $\{1, 2\}$ and we adapt Yannakakis's construction to reduce the maximum matching problem on K_{2n} to the maximum weight Hamiltonian cycle problem on K_{4n}. To simplify notation, we identify $[4n]$ with $\{0, 1\} \times [2n]$, i.e., the vertices are labeled by pairs (i, j) with i being 0 or 1 and $j \in [2n]$. Given a graph G on $[2n]$, we think of it as being supported on $\{0\} \times [2n]$.

We consider the weighted graph $\widetilde{G}$ with edges and weights:

Edge	Weight	
$\{(0, j), (1, j)\}$	2	$j \in [2n]$
$\{(1, j), (1, k)\}$	1	$j, k \in [2n]$
$\{(0, j), (0, k)\}$	1	$\{j, k\} \in E(G)$

For every perfect matching M on $[2n]$, choose a Hamiltonian cycle C_M containing the edges

(i) $\{(0, j), (1, j)\}$ for $j \in [2n]$,

(ii) $\{(0, j), (0, k)\}$ for $\{j, k\} \in E(G)$,

(iii) n additional edges of the form $\{(1, j), (1, k)\}$ to obtain a Hamiltonian cycle.

Note that

$$\mathrm{val}_{\widetilde{G}}(C_M) = 5n + |M \cap E(G)|. \tag{2.11}$$

We now determine the maximum of $\mathrm{val}_{\widetilde{G}}$ on all Hamiltonian cycles. Therefore let C be an arbitrary Hamiltonian cycle. Let us consider C restricted to $\{0\} \times [2n]$; its components

are (possible empty) paths. Let k be the number of components, which are non-empty paths, and contained in G. Obviously, $k \leq \nu(G)$, where $\nu(G)$ is the matching number, as selecting one edge from every such component provides a k-matching of G.

Let l be the number of components containing at least one edge not in G. Note that $k + l \leq n$, because choosing one edge of all these $k + l$ components, we obtain a $(k + l)$-matching on $[2n]$, similarly as in the previous paragraph.

Finally, let m be the number of single vertex components. Therefore C contains exactly $2n - (k + l + m)$ edges on $\{0\} \times [2n]$, of which at least l are not contained in G. Hence the contribution of these edges to the weight $\mathrm{val}_{\widetilde{G}}(C)$ is at most

$$\mathrm{val}_{\widetilde{G}}(C \cap E(\{0\} \times [2n])) \leq 2n - (k + l + m) - l = 2n - k - 2l - m. \tag{2.12}$$

Moreover, the cycle C contains exactly $2n - (k + l + m)$ edges on $\{1\} \times [2n]$ whose contribution to the weight is

$$\mathrm{val}_{\widetilde{G}}(C \cap E(\{1\} \times [2n])) = 2n - (k + l + m). \tag{2.13}$$

Finally, C contains $2(k + l + m)$ edges between the partitions $\{0\} \times [2n]$ and $\{1\} \times [2n]$, all of which have weight at most 2. In fact, at each of the m single vertex components in $\{0\} \times [2n]$, only one of the edges can be of the form $\{(0, j), (1, j)\}$, the other edge must have weight 0. Therefore the contribution of the edges between the partitions is at most

$$\mathrm{val}_{\widetilde{G}}(C \cap E(\{0\} \times [2n], \{1\} \times [2n]) \leq 2[2(k + l + m) - m] = 4k + 4l + 2m. \tag{2.14}$$

Summing up Eqs. (2.12), (2.13) and (2.14), we obtain the following upper bound on the weight of C:

$$\mathrm{val}_{\widetilde{G}}(C) \leq 4n + 2k + l \leq 5n + k \leq 5n + \nu(G).$$

Together with Eq. (2.11), this proves $\max_C \mathrm{val}_{\widetilde{G}}(C) = 5n + \nu(G)$. Thus the $\mathrm{val}_{\widetilde{G}}$ and C_M

reduce the maximum matching problem to the maximum Hamiltonian cycle problem with $\beta(G) = \widetilde{G}$, $\gamma(M) = C_M$ and $\mu(G) = 5n$. Hence the LP formulation complexity of the maximum Hamiltonian cycle problem is $2^{\Omega(n)}$ by Proposition 2.2.2.

2.3 Inapproximability of VertexCover and IndependentSet

We will now establish inapproximability results for VertexCover and IndependentSet via reduction from MaxCUT, even for bounded degree subgraphs. These two problems are of particular interest, answering a question of Singh (2010) and Chan et al. (2013) as well as a weak version of sparse graph conjecture from Braun, Fiorini, and Pokutta (2014). Moreover, VertexCover is not of the CSP type, therefore the methods from Chan et al. (2013) do not apply. Using our reduction framework, recently these results have been further improved in Bazzi et al. (2015) to obtain $(2-\varepsilon)$-inapproximability for VertexCover (which is optimal) and inapproximability of IndependentSet within any constant factor.

Formulation complexity depend heavily on how a problem is formulated. For example, the model of IndependentSet used here is motivated by its combinatorial counterpart, and captures standard LPs, like the ones coming from Sherali–Adams hierarchies. In this model, IndependentSet for a given graph G is approximable within a factor of $2\sqrt{n}$ with a polynomial sized LP, see Bazzi et al. (2015). However, the formulation complexity of *another model of* the maximum independent set problem with an approximation factor $n^{1-\varepsilon}$ is subexponential, rephrasing Fiorini et al. (2012) (see also Braun et al. 2012; Braverman and Moitra 2013; Braun et al. 2014a), see Section 2.1.1. In this model the instances come from the polytope world, and are actually formal linear combinations of several graphs, and this makes the difference.

The current best PCP bound for bounded degree IndependentSet can be found in Chan (2013). See also Austrin, Khot, and Safra (2009) for inapproximability results assuming the Unique Games Conjecture.

The minimization problem VertexCover(G) of a graph G asks for a minimum weighted vertex cover of G. We consider the non-uniform model with instances being the induced subgraphs of G.

Definition 2.3.1 (VertexCover)**.** Given a graph G, the problem VertexCover(G) has all vertex covers S of G as feasible solutions, and instances all induced subgraphs H of G. The problem VertexCover(G) is the minimization problem with its objective function having values $\mathrm{val}_H(S) := |S \cap V(H)|$. The problem VertexCover$(G)_\Delta$ is the restriction of instances to induced subgraphs H, with maximum degree at most Δ.

Note that for every vertex cover S of G, any induced subgraph H has $S \cap V(H)$ as a vertex cover, and all vertex covers of H are of this form. In particular, $\min \mathrm{val}_H$ is the minimum size of a vertex cover of H.

The problem IndependentSet asks for maximum sized independent sets in graphs. As independent sets are exactly the complements of vertex covers, it is natural to use a formulation similar to VertexCover.

Definition 2.3.2 (IndependentSet)**.** Given a graph G, IndependentSet(G) is the maximization problem that has all independent sets S of G as feasible solutions, and instances are all induced subgraphs H of G. The objective function is $\mathrm{val}_H(S) := |S \cap V(H)|$. The subproblem IndependentSet$(G)_\Delta$ is the restriction to all induced subgraphs H with maximum degree at most Δ.

For both VertexCover and IndependentSet, we shall use the following conflict graph G for a fixed n, similar to Feige et al. (1991); we might think of G as a *universal* graph encoding all possible instances. Let the vertices of G be all partial assignments σ of two variables x_i and x_j satisfying the 2-XOR clause $x_i \oplus x_j = 1$. Two vertices σ_1 and σ_2 are connected if and only if the assignments σ_1 and σ_2 are incompatible (i.e., assign different truth values to some common variable), see Figure 2.2 for an illustration. As we are considering problems for

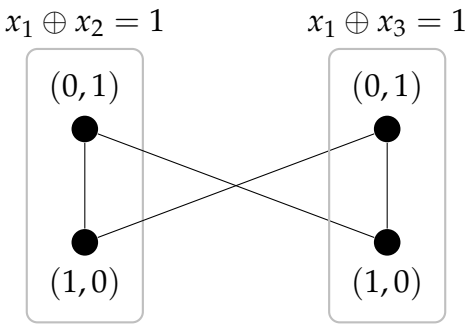

Figure 2.2: Conflict graph of 2-XOR clauses. We include edges between all conflicted partial assignment to variables.

optimizing size of vertex sets, it is natural to define the size of an instance, i.e., a subgraph K, as the size of its vertex set $|V(K)|$.

We give a summary of the results for VertexCover and IndependentSet in Tables 2.3-2.6.

Table 2.3: Summary of the hardness results for VertexCover achieved by a reduction from MaxCUT which is described in Theorem 2.3.3. The last line in this table holds assuming Conjecture 2.2.11.

	completeness	soundness	inapprox factor	size
fc_+	$1/2+\varepsilon$	$3/4-\varepsilon$	$3/2-\Theta(\varepsilon)$	$2^{m^{c(\varepsilon)}}$
$\mathrm{fc}_\oplus$	exact case		1	$2^{\Omega\left(m^{2/13}\right)}$
	$6/10+\varepsilon$	$5/8-\varepsilon$	$25/24-\Theta(\varepsilon)$	$m^{\Omega(\log m/\log\log m)}$
	$1/2+\varepsilon$	$1-c_{GW}/2-\varepsilon$	$2-c_{GW}-\Theta(\varepsilon)$	superpolynomial

Table 2.4: Summary of the hardness results for the bounded degree case $\mathsf{VertexCover}_\Delta$ achieved by a reduction from MaxCUT_Δ described in Theorem 2.3.3. The last line in this table holds assuming Conjecture 2.2.11.

	completeness	soundness	inapprox factor	size
fc_+	$1/2+\varepsilon$	$3/4-\varepsilon$	$3/2-\Theta(\varepsilon)$	$2^{m^{c(\varepsilon)}}$
$\mathrm{fc}_\oplus$	$1/2+\varepsilon$	$1-c_{GW}/2-\varepsilon$	$2-c_{GW}-\Theta(\varepsilon)$	superpolynomial

Theorem 2.3.3. *For every $\varepsilon > 0$ there is a Δ and a constant $c(\varepsilon)$ that for infinitely many m, there is a graph G with $|V(G)| = m$ such that* $\mathrm{fc}_+(\mathsf{VertexCover}(G)_\Delta, 1/2+$

Table 2.5: Summary of the hardness results for IndependentSet achieved by a reduction from MaxCUT which is described in Theorem 2.3.3. The last line in this table holds assuming Conjecture 2.2.11.

	completeness	soundness	inapprox factor	size
fc_+	$1/2-\varepsilon$	$1/4+\varepsilon$	$1/2+\Theta(\varepsilon)$	$2^{m^{c(\varepsilon)}}$
$\mathrm{fc}_\oplus$	exact case		1	$2^{\Omega\left(m^{2/13}\right)}$
	$4/10-\varepsilon$	$3/8+\varepsilon$	$15/16+\Theta(\varepsilon)$	$m^{\Omega(\log m/\log\log m)}$
	$1/2-\varepsilon$	$c_{GW}/2+\varepsilon$	$c_{GW}+\Theta(\varepsilon)$	superpolynomial

Table 2.6: Summary of the hardness results for the bounded degree case $\mathsf{IndependentSet}_\Delta$ achieved by a reduction from MaxCUT_Δ described in Theorem 2.3.3. The last line in this table holds assuming Conjecture 2.2.11.

	completeness	soundness	inapprox factor	size
fc_+	$1/2-\varepsilon$	$1/4+\varepsilon$	$1/2+\Theta(\varepsilon)$	$2^{m^{c(\varepsilon)}}$
$\mathrm{fc}_\oplus$	$1/2-\varepsilon$	$c_{GW}/2+\varepsilon$	$c_{GW}+\Theta(\varepsilon)$	superpolynomial

$\varepsilon, 3/4-\varepsilon) \geq 2^{m^{c(\varepsilon)}}$ *showing inapproximability within a factor of* $\frac{3}{2}-\Theta(\varepsilon)$*, and also* $\mathrm{fc}_+(\mathsf{IndependentSet}(G)_\Delta, 1/2-\varepsilon, 1/4+\varepsilon) \geq 2^{m^{c(\varepsilon)}}$ *establishing an inapproximability factor* $\frac{1}{2}+\Theta(\varepsilon)$*. Assuming Conjecture 2.2.11, we also have* $\mathrm{fc}_\oplus(\mathsf{VertexCover}(G)_\Delta, 1/2+\varepsilon, 1-c_{GW}/2-\varepsilon)$ *and* $\mathrm{fc}_\oplus(\mathsf{IndependentSet}(G)_\Delta, 1/2-\varepsilon, c_{GW}/2+\varepsilon)$ *are superpolynomial, achieving inapproximability factors* $2-c_{GW}-\Theta(\varepsilon)$ *and* $c_{GW}+\Theta(\varepsilon)$*, respectively. Additionally for the unbounded degree problems we have* $\mathrm{fc}_\oplus(\mathsf{VertexCover}(G),\ 6/10+\varepsilon,\ 5/8-\varepsilon) = m^{\Omega(\log m/\log\log m)}$ *and* $\mathrm{fc}_\oplus(\mathsf{IndependentSet}(G),\ 4/10-\varepsilon,\ 3/8+\varepsilon) = m^{\Omega(\log m/\log\log m)}$ *in the approximate case and* $\mathrm{fc}_\oplus(\mathsf{VertexCover}(G)) = 2^{\Omega\left(m^{2/13}\right)}$ *and* $\mathrm{fc}_\oplus(\mathsf{IndependentSet}(G)) = 2^{\Omega\left(m^{2/13}\right)}$ *for the exact problems.*

Proof. We shall use the graph G constructed above, which has $m = 2\binom{n}{2}$ vertices. We reduce MaxCUT_Δ to $\mathsf{VertexCover}(G)_{2\Delta-1}$ using Corollary 2.2.6 with $\alpha = -1$, $\mu = 2$ and $\eta = 2$ together with the values of τ_1 and σ_1 as they are given in Table 2.1 and Table 2.2. For demonstration purposes, we shall write out the explicit guarantees below for the linear case. Recall that for a graph K on $[n]$ with maximum degree at most Δ, the guarantees for

MaxCUT are $C_{\mathsf{MaxCUT}}(K) = (1-\varepsilon)\,|E(K)|$ and $S_{\mathsf{MaxCUT}}(K) = (1/2+\varepsilon)\,|E(K)|$. For $\mathsf{VertexCover}(G)_{2\Delta-1}$, we have the following explicit guarantees:

$$C_{\mathsf{VertexCover}(G)}(H) = (1/2+\varepsilon/2)\,|V(H)|\,,$$
$$S_{\mathsf{VertexCover}(G)}(H) = (3/4-\varepsilon/2)\,|V(H)|\,.$$

Let $H(K)$ be the induced subgraph of G on the set of all partial assignments σ which assign values to variables x_i, x_j corresponding to an edge $\{i,j\}$ of K, i.e., the vertex set is $V(H(K)) \coloneqq \left\{\sigma \,\middle|\, \{i,j\} \in E(K),\ \operatorname{dom}\sigma = \{x_i, x_j\}\right\}$. In particular, $|V(H(K))| = 2\,|E(K)|$, as there are two partial assignments per each edge $\{i,j\}$.

Note that for every partial assignment σ to x_i and x_j, there are $2\Delta-1$ partial assignments incompatible with it in $V(H(K))$: exactly one assignment for every edge of K incident to i or j. Thus the maximum degree of $H(K)$ is at most $2\Delta-1$.

We now define the two maps providing the reduction. Let $\beta(K) \coloneqq H(K)$. For a total assignment s, let $\gamma(s) \coloneqq \left\{\sigma \,\middle|\, \sigma \not\subseteq s\right\}$ be the set of partial assignments incompatible with s; this is clearly a vertex cover.

It remains to show that this is a reduction. For every edge $\{i,j\} \in K$, there are two partial assignments σ to x_i and x_j satisfying $x_i \oplus x_j = 1$. If s satisfies $x_i \oplus x_j = 1$, i.e., $\{i,j\}$ is in the cut induced by s, then exactly one of the σ is compatible with s, otherwise both of the assignments are incompatible. This provides

$$\mathrm{val}^{\mathsf{VertexCover}}_{H(K)}[\gamma(s)] = \left|\left\{\sigma \,\middle|\, \sigma \not\subseteq s\right\}\right| = 2\,|E(K)| - \mathrm{val}^{\mathsf{MaxCUT}}_{K}(s).$$

To compare optimum values, note that for any vertex cover S of G, the partial assignments $\{\sigma \,|\, \sigma \notin S\}$ occurring in the complement of S are compatible (as the complement forms a stable set), hence there is a global assignment s of $x_1, \ldots, x_n$ compatible with all of them.

In particular, $\gamma(s) \subseteq S$, hence $\mathrm{val}^{\mathsf{VertexCover}}_{H(K)}(S) \geq \mathrm{val}^{\mathsf{MaxCUT}}_{K}[\gamma(s)]$, so that we obtain

$$\begin{aligned}\min \mathrm{val}^{\mathsf{VertexCover}}_{H(K)} &= \min_s \mathrm{val}^{\mathsf{VertexCover}}_{H(K)}[\gamma(s)] = 2\,|E(K)| - \max \mathrm{val}^{\mathsf{MaxCUT}}_{K} \\ &\geq 2\,|E(K)| - (1/2+\varepsilon)\,|E(K)| = (3/4-\varepsilon/2)\,|V(H(K))| \\ &= S_{\mathsf{VertexCover}}(H(K)).\end{aligned}$$

Finally, it is easy to verify that $C_{\mathsf{VertexCover}}(H(K)) = 2|E(K)| - C_{\mathsf{MaxCUT}}(K)$. This finishes the proof that β and γ define a reduction to $\mathsf{VertexCover}(G)_\Delta$. Hence by Corollary 2.2.6 (or Proposition 2.2.2), using $m = 2\binom{n}{2}$, we have $\mathrm{fc}_+(\mathsf{VertexCover}(G)_{2\Delta-1}, 3/2-2\varepsilon, 1+2\varepsilon) = n^{\Omega(\frac{\log n}{\log\log n})} = m^{\Omega(\frac{\log m}{\log\log m})}$ for infinitely many n. The exact case follows with Corollary 2.2.5.

For IndependentSet, we apply a similar reduction from MaxCUT, in particular we reduce MaxCUT_Δ to $\mathsf{IndependentSet}(G)_{2\Delta-1}$. We define $\beta(K) \coloneqq H(K)$ as above and we set $\gamma(s) \coloneqq \{\sigma \mid \sigma \subseteq s\}$ to be the set of partial assignments compatible with the total assignment s, this is clearly an independent set, containing exactly one vertex per satisfied clause. In particular, $\mathrm{val}_{H(K)}[\gamma(s)] = \mathrm{val}_K(s)$. The rest of the argument is analogous to the case of $\mathsf{VertexCover}(G)_\Delta$, and hence omitted. Now the parameters for Corollary 2.2.6 are $\alpha = 1$, $\mu = 0$ and $\eta = 2$, again together with the values for τ_1 and σ_1 as given in Tables 2.1 and 2.2, e.g., $\tau_1 = 1-\varepsilon$, and $\sigma_1 = 1/2+\varepsilon$ in the linear case.

□

2.4 Inapproximability of CSPs

In this section we present example reductions for minimum and maximum constraint satisfaction problems. Some of the results for *binary* Max-CSP*s*, (for CSPs as defined in Definition 1.1.2) could also be obtained in the LP case from Kothari, Meka, and Raghavendra (2016) by combination with the respective Sherali–Adams/Lasserre gap instances . For simplicity of exposition, we reduce from Max-2-XOR, or sometimes MaxCUT, however by reducing from the subproblem MaxCUT_Δ, we immediately obtain the results for bounded

occurrence of literals, with Δ depending on the approximation factor.

2.4.1 Max-MULTI-k-CUT: a non-binary CSP

The Max-MULTI-k-CUT problem is interesting on its own being a CSP over a non-binary alphabet, thus the framework in Chan et al. (2013) does not readily apply. Note that Max-MULTI-k-CUT is APX-hard, as it contains MaxCUT. The current best PCP inapproximability bound $1 - 1/(34k) + \varepsilon$ is given by Kann et al. (1997).

Here we omit the definition of non-binary CSPs, where the feasible solutions are no longer two-valued assignments, and restrict to Max-MULTI-k-CUT.

Definition 2.4.1 (Max-MULTI-k-CUT)**.** For fixed positive integers n and k, the problem Max-MULTI-k-CUT has

(i) **feasible solutions:** all partitions of $[n]$ into k sets;

(ii) **instances:** all graphs G with $V(G) \subseteq [n]$.

(iii) **objective function:** for a graph G and a partition p of $[n]$, let $\mathrm{val}_G(p)$ be the number of edges of G whose end points lie in different cells of p.

This differs from a binary CSP only by having a different kind of feasible solutions. Hence it is still natural to define the size of an instance, i.e., graph G, as the number of clauses, i.e, number of edges $|E(G)|$.

We summarize the results in Table 2.7 and prove the statements in Corollary 2.4.2.

Corollary 2.4.2. *Let $k \geq 3$ be a fixed integer. Then for every $\varepsilon > 0$ there is a constant $\bar{c}(\varepsilon)$ such that for infinitely many n,*

$$\mathrm{fc}_+\left(\text{Max-MULTI-}k\text{-CUT}, \frac{c(k)+1}{c(k)} - \varepsilon, \frac{c(k)+1/2}{c(k)} + \varepsilon\right) \geq 2^{n^{\bar{c}(\varepsilon)}}$$

Table 2.7: Summary of the hardness results for Max-MULTI-k-CUT achieved by a reduction from MaxCUT which is described in Corollary 2.4.2. The last line in this table holds assuming Conjecture 2.2.11.

	completeness	soundness	inapprox factor	size
fc_+	$\frac{c(k)+1}{c(k)} - \varepsilon$	$\frac{2c(k)+1}{2c(k)} + \varepsilon$	$\frac{2c(k)+1}{2c(k)+2} + \Theta(\varepsilon)$	$2^{n^{\bar{\varepsilon}(\varepsilon)}}$
$\mathrm{fc}_\oplus$	exact case		1	$2^{\Omega\left(n^{2/13}\right)}$
	$\frac{5c(k)+4}{5c(k)} - \varepsilon$	$\frac{4c(k)+3}{4c(k)} + \varepsilon$	$\frac{20c(k)+15}{20c(k)+16} + \Theta(\varepsilon)$	$n^{\Omega(\log n/\log\log n)}$
	$\frac{c(k)+1}{c(k)} - \varepsilon$	$\frac{c(k)+c_{GW}}{c(k)} + \varepsilon$	$\frac{c(k)+c_{GW}}{c(k)+1} + \Theta(\varepsilon)$	superpolynomial

achieving inapproximability factor $\frac{2c(k)+1}{2c(k)+2} + \Theta(\varepsilon)$*, where*

$$c(k) := \binom{k-2}{2}\left(\binom{k+2}{2} - 3\right) + 2(k-2)\left(\binom{k+2}{2} - 3\right).$$

In the SDP case we have

$$\mathrm{fc}_\oplus\left(\text{Max-MULTI-}k\text{-CUT}, \frac{5c(k)+4}{5c(k)} - \varepsilon, \frac{4c(k)+3}{4c(k)} + \varepsilon\right) \geq n^{\Omega(\log n/\log\log n)}$$

achieving a factor of $\frac{20c(k)+15}{20c(k)+16} + \Theta(\varepsilon)$ *and in the exact case* $\mathrm{fc}_\oplus(\text{Max-MULTI-}k\text{-CUT}) \geq n^{\Omega(\log n/\log\log n)}$.

Further assuming Conjecture 2.2.11, we also have that $\mathrm{fc}_\oplus(\text{Max-MULTI-}k\text{-CUT}, c(k) + 1 - \varepsilon, c(k) + c_{GW} + \varepsilon)$ *is superpolynomial, showing inapproximability factor* $\frac{c(k)+c_{GW}}{c(k)+1} + \Theta(\varepsilon)$.

Proof. We reduce MaxCUT to Max-MULTI-k-CUT. The reduction is essentially identical to Papadimitriou and Yannakakis (1991), however we have to verify its compatibility with our reduction mechanism. To this end it will suffice to define the reduction maps β and γ.

Given a graph G, we construct a new graph $\beta(G)$ as illustrated in Figure 2.3. Consider the vertices of G together with $k-2$ new vertices $-k, \ldots, -3$. For every pair of vertices i, j we add n_{ij} copies of an almost complete graph on $k+2$ vertices, two of which are $(i), (j)$, as follows.

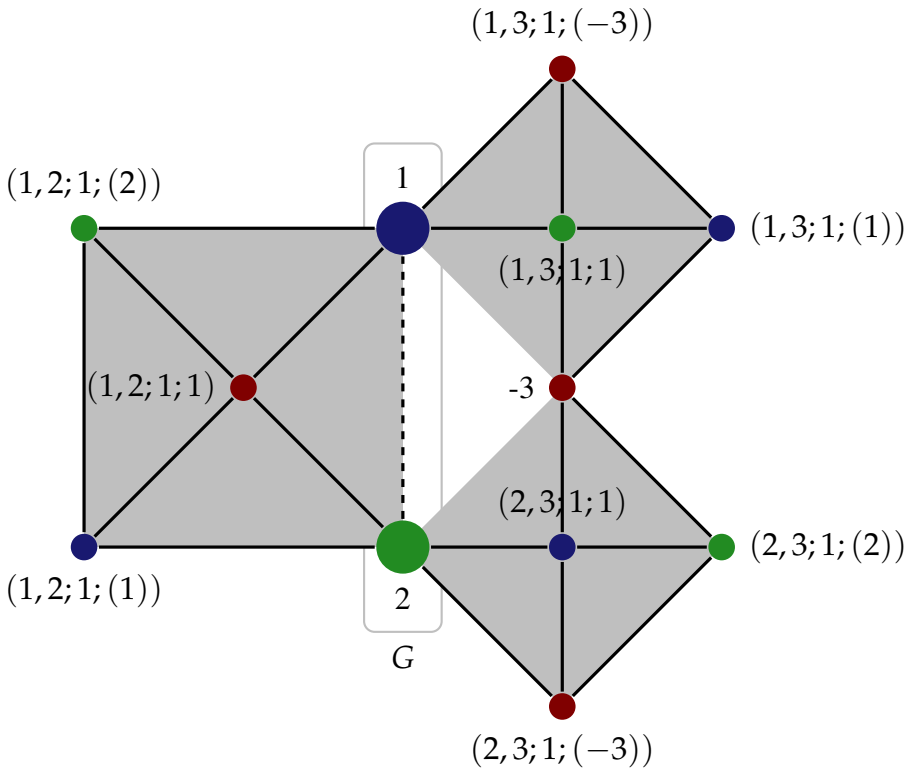

Figure 2.3: Reduction between MaxCUT and Max-MULTI-k-CUT. Here $k = 3$ and $G = K_2$. The dashed edge denotes the edge of G, which is *not* contained in the reduction. The squares are the copies of almost complete graphs added. The partition is represented by coloring the vertices: blue and green are the original cells on $[2]$, and red is an additional color.

First, let us determine the number n_{ij} of copies added. For $i, j \in [n]$, we add one copy if i and j are connected in G, and no copies otherwise. For $i, j \in \{-k, \ldots, -3\}$ we add $|E(G)|$ many copies. Finally, for $i \in [n]$ and $j \in \{-k, \ldots, -3\}$ we add $\deg_G(i)$ many copies.

Now let us describe the copies themselves. Let us fix i, j and let $x \in [n_{ij}]$ be an index of the copy. We add k new vertices $(i, j; x; (i))$, $(i, j; x; (j))$, and $(i, j; x; t)$ for $t = 1, \ldots, k-2$. We connect every pair of the $k+2$ vertices $i, j, (i, j; x; (i)), (i, j; x; (j)), (i, j; x; t)$ with an edge except the pairs $\{i, (i, j; x, (i))\}$, $\{i, j\}$, and $\{(i, j; x, j), (j)\}$. This is done for all i, j, x, and we let $\beta(G)$ be the graph so obtained.

By construction, $\beta(G)$ has

$$\begin{aligned} |E(\beta(G))| &= \left[\binom{k+2}{2} - 3\right] \cdot \sum_{ij} n_{ij} \\ &= \left[\binom{k+2}{2} - 3\right] \left(1 + 2(k-2) + \binom{k-2}{2}\right) |E(G)| \end{aligned}$$

many edges, and its vertex set $V(\beta(G))$ is contained in the set

$$[n] \cup \{-k, \ldots, -3\} \cup \left\{(i, j; x; t) \;\middle|\; -k \leq i < j \leq 3, x \in [n_{ij}], t \in [k]\right\}$$

having size polynomial in n:

$$m := n + (k-2) + k\left[\binom{k-2}{2} + 2(k-2) + 1\right]\binom{n}{2}.$$

We define γ to map every 2-partition p of $[n]$ into a k-partition of $\beta(G)$ by extending it as follows. Let p_1 and p_2 denote the cells of p. Elements of p_1 and p_2 go to the first and second cell of the $\gamma(p)$, respectively. The new vertices $-i$ added for $i = -k, \ldots, -3$ go to the i-th cell. Vertices $(i, j; x; (i))$ and $(i, j; x; (j))$ go to the cell of i and j, respectively. For fixed i, j, x the vertices $(i, j; x; t)$ for $t = 1, \ldots, k-2$ are put into $k-2$ different cells, which do not contain $(i, j; x; (i))$ or $(i, j; x; (j))$. This is possible as there are k different cells.

By Papadimitriou and Yannakakis (1991),

$$\begin{aligned}\mathrm{val}^{\text{Max-MULTI-}k\text{-CUT}}_{\beta(G)}[\gamma(p)] &= \mathrm{val}^{\text{MaxCUT}}_{G}(p) + \mu(G),\\ \max \mathrm{val}^{\text{Max-MULTI-}k\text{-CUT}}_{\beta(G)} &= \max \mathrm{val}^{\text{MaxCUT}}_{G} + \mu(G),\end{aligned}$$

where

$$\mu(G) = |E(\beta(G))| - |E(G)| = \underbrace{\left[\binom{k-2}{2} + 2(k-2)\right]\left[\binom{k+2}{2} - 3\right]}_{c(k)} |E(G)|\,.$$

Therefore we obtain a reduction from MaxCUT on n vertices to Max-MULTI-k-CUT on m vertices, with m polynomially bounded in n. Combining Corollary 2.2.6 with parameters $\alpha = 1$, $\mu = c(k)$ and $\eta = c(k)$ with the hardness results in Table 2.1, we get for example in the LP case with $\tau_1 = 1-\varepsilon$ and $\sigma_1 = 1/2+\varepsilon$, that there is a constant $\bar{c}(\varepsilon)$ such that $\mathrm{fc}_+\left(\text{Max-MULTI-}k\text{-CUT}, \frac{c(k)+1-\varepsilon}{c(k)}, \frac{c(k)+1/2+\varepsilon}{c(k)}\right) = 2^{m^{c(\varepsilon)}} = 2^{n^{\bar{c}(\varepsilon)}}$. The SDP cases follow analogously while we assume Conjecture 2.2.11 for the inapproximability factor of $\frac{c(k)+c_{GW}}{c(k)+1} + \Theta(\varepsilon)$. □

2.4.2 Inapproximability of general 2-CSPs

First we consider general CSPs with no restrictions on constraints, for which the exact approximation factor can be easily established. We present the hardness of LP approximation here. The LP with matching factor can be found in Trevisan (1998).

Definition 2.4.3 (Max-2-CSP and Max-2-CONJSAT)**.** The problem Max-2-CSP is the CSP on variables $x_1, \ldots, x_n$ with constraint family $\mathcal{C}_{2CSP}$ consisting of all possible constraints depending on at most two variables. The problem Max-2-CONJSAT is the CSP with constraint family consisting of all possible conjunctions of two literals.

Table 2.8: Summary of the hardness results for Max-2-CSP given in Corollary 2.4.4. The last line holds under the assumption of Conjecture 2.2.11.

	completeness	soundness	inapprox factor	size
fc_+	$1-\varepsilon$	$1/2+\varepsilon$	$1/2+\Theta(\varepsilon)$	$2^{n^{c(\varepsilon)}}$
$\mathrm{fc}_\oplus$	exact case		1	$2^{\Omega\left(n^{2/13}\right)}$
	$4/5-\varepsilon$	$3/4+\varepsilon$	$15/16+\Theta(\varepsilon)$	$n^{\Omega(\log n/\log\log n)}$
	$1-\varepsilon$	$c_{GW}+\varepsilon$	$c_{GW}+\Theta(\varepsilon)$	superpolynomial

Table 2.9: Summary of the hardness results for Max-2-CONJSAT given in Corollary 2.4.4. The last line holds under the assumption of Conjecture 2.2.11.

	completeness	soundness	inapprox factor	size
fc_+	$1/2-\varepsilon$	$1/4+\varepsilon$	$1/2+\Theta(\varepsilon)$	$2^{n^{c(\varepsilon)}}$
$\mathrm{fc}_\oplus$	exact case		1	$2^{\Omega\left(n^{2/13}\right)}$
	$4/10-\varepsilon$	$3/8+\varepsilon$	$15/16+\Theta(\varepsilon)$	$n^{\Omega(\log n/\log\log n)}$
	$1/2-\varepsilon$	$c_{GW}/2+\varepsilon$	$c_{GW}+\Theta(\varepsilon)$	superpolynomial

Corollary 2.4.4. *For every $\varepsilon > 0$ there is a constant $c(\varepsilon)$ such that for infinitely many n, we have* $\mathrm{fc}_+(\text{Max-2-CSP}, 1-\varepsilon, 1/2+\varepsilon) \geq 2^{n^{c(\varepsilon)}}$ *achieving inapproximability factor $\frac{1}{2}+\Theta(\varepsilon)$, where n is the number of variables of* Max-2-CSP. *Similarly,* $\mathrm{fc}_+(\text{Max-2-CONJSAT}, 1/2-\varepsilon, 1/4+\varepsilon) \geq 2^{n^{c(\varepsilon)}}$ *establishing inapproximability factor $\frac{1}{2}+\Theta(\varepsilon)$ for infinitely many n. Moreover, in the SDP case for the exact problems we have* $\mathrm{fc}_\oplus(\text{Max-2-CSP}) \geq 2^{\Omega\left(n^{2/13}\right)}$ *and* $\mathrm{fc}_\oplus(\text{Max-2-CONJSAT}) \geq 2^{\Omega\left(n^{2/13}\right)}$. *For approximation versions we have* $\mathrm{fc}_\oplus(\text{Max-2-CSP}, 4/5-\varepsilon, 3/4+\varepsilon) = n^{\Omega(\log n/\log\log n)}$ *and* $\mathrm{fc}_\oplus(\text{Max-2-CONJSAT}, 4/10-\varepsilon, 3/8+\varepsilon) = n^{\Omega(\log n/\log\log n)}$ *and, finally assuming Conjecture 2.2.11,* $\mathrm{fc}_\oplus(\text{Max-2-CSP}, 1-\varepsilon, c_{GW}+\varepsilon)$ *and* $\mathrm{fc}_\oplus(\text{Max-2-CONJSAT}, 1/2-\varepsilon, c_{GW}/2+\varepsilon)$ *are superpolynomial showing inapproximability factor $c_{GW}+\Theta(\varepsilon)$.*

Proof. We identify Max-2-XOR as a subproblem of Max-2-CSP: Every 2-XOR clause is evidently a boolean function of 2 variables. So restricting the instances of Max-2-CSP to 2-XOR clauses with 0/1 weights gives Max-2-XOR. Now with Theorem 2.2.8, 2.2.9 and

2.2.10, the results for Max-2-CSP follow.

The claim about Max-2-CONJSAT follows via the reduction from Max-2-CSP to Max-2-CONJSAT in Trevisan (1998). We prefer to reduce from Max-2-XOR instead for easier control over the approximation guarantees. The idea is to write each clause C in disjunctive normal form, and replace C with the set $S(C)$ of conjunctions in its normal form, one conjunction for every assignment satisfying C. In particular, for 2-XOR clauses $S(x_i \oplus x_j = 1) = \{x_i \wedge \neg x_j, \neg x_i \wedge x_j\}$ and $S(x_i \oplus x_j = 0) = \{x_i \wedge x_j, \neg x_i \wedge \neg x_j\}$. Therefore formally, a set of clauses L is mapped to $\beta(L) = \bigcup_{C \in L} S(C)$. Every assignment of variables is mapped to themselves, i.e., γ is the identity. We have $\mathrm{val}_{\beta(L)}(s) = \mathrm{val}_L(s)$ and $|\beta(L)| = 2\,|L|$. Now the claim follows with $\alpha = 1$ in Corollary 2.2.5 or with $\alpha = 1$, $\mu = 0$ and $\eta = 2$ in Corollary 2.2.6. □

2.4.3 Max-k-SAT inapproximability

We now establish an LP-inapproximability factor of $\frac{3}{4} + \varepsilon$ and an SDP-inapproximability factor of $35/36 + \varepsilon$ for Max-2-SAT via a direct reduction from MaxCUT. Note that Goemans and Williamson (1994) show the existence of an LP that achieves a factor of $\frac{3}{4}$, so that our estimation is tight in the LP case. Moreover, in Feige and Goemans (1995) it is shown that Max-2-SAT can be approximated with a small SDP within a factor of 0.931 leaving a gap of about 0.04.

Obviously, the same factor applies for Max-k-SAT with $k \geq 2$, too. We allow clauses with less than k literals in Max-k-SAT, which is in line with the definition in Schoenebeck (2008) to maintain compatibility. Note that Lee, Raghavendra, and Steurer (2014, Theorem 1.5) establishes $7/8 + \varepsilon$ inapproximability for Max-3-SAT even in the SDP case.

Definition 2.4.5 (Max-k-SAT)**.** For fixed $n, k \in \mathbb{N}$, the problem Max-k-SAT is the CSP on the set of variables $\{x_1, \ldots, x_n\}$, where the constraint family $\mathcal{C}$ is the set of all sat clauses which consist of at most k literals.

Table 2.10: Summary of the hardness results for Max-k-SAT with $k \geq 2$ given in Corollary 2.4.6. The last line holds under the assumption of Conjecture 2.2.11.

	completeness	soundness	inapprox factor	size
fc_+	$1-\varepsilon$	$3/4+\varepsilon$	$1/2+\Theta(\varepsilon)$	$2^{n^{c(\varepsilon)}}$
$\mathrm{fc}_\oplus$	exact case		1	$2^{\Omega\left(n^{2/13}\right)}$
	$9/10-\varepsilon$	$7/8+\varepsilon$	$35/36+\Theta(\varepsilon)$	$n^{\Omega(\log n/\log\log n)}$
	$1-\varepsilon$	$(1+c_{GW})/2+\varepsilon$	$(1+c_{GW})/2+\Theta(\varepsilon)$	superpolynomial

Corollary 2.4.6. *Let $k \geq 2$ and $\varepsilon > 0$. Then there is a constant $c(\varepsilon)$ such that for infinitely many n, $\mathrm{fc}_+(\text{Max-}k\text{-SAT}, 1-\varepsilon, 3/4+\varepsilon) \geq 2^{n^{c(\varepsilon)}}$ achieving an inapproximability factor $\frac{3}{4}+\Theta(\varepsilon)$, where n is the number of variables. In the case of SDPs we have that $\mathrm{fc}_\oplus(\text{Max-}k\text{-SAT}) \geq 2^{\Omega\left(n^{2/13}\right)}$ for the exact case, $\mathrm{fc}_\oplus(\text{Max-}k\text{-SAT}, 9/10-\varepsilon, 7/8+\varepsilon) \geq n^{\Omega(\log n/\log\log n)}$ achieving an inapproximability factor of $35/36+\Theta(\varepsilon)$ and, assuming Conjecture 2.2.11, that $\mathrm{fc}_\oplus(\text{Max-}k\text{-SAT}, 1-\varepsilon, (1+c_{GW})/2+\varepsilon)$ is superpolynomial establishing an inapproximability factor $\frac{1+c_{GW}}{2}+\Theta(\varepsilon)$.*

Proof. We reduce MaxCUT to Max-2-SAT. For a 2-XOR clause $l = (x_i \oplus x_j = 1)$ with $i, j \in [n]$, we define two auxiliary constraints $C_1(l) = (x_i \vee x_j)$ and $C_2(l) = (\bar{x}_i \vee \bar{x}_j)$. Let $\beta(L) := \{C_1(l), C_2(l) \mid l \in L\}$ for a set of 2-XOR clauses L. We choose γ to be the identity map. Observe that whenever l is satisfied by a partial assignment s then both $C_1(l)$ and $C_2(l)$ are also satisfied by s, otherwise exactly one of $C_1(l)$ and $C_2(l)$ is satisfied. Hence we obtain a reduction from MaxCUT to Max-2-SAT. Using the hardness results of MaxCUT given in Theorem 2.2.8, Theorem 2.2.9 and Theorem 2.2.10 together with the reductions of Corollary 2.2.5 and Corollary 2.2.6 with parameters $\alpha = 1$, $\mu = 1$ and $\eta = 2$ the results follow. The statement for general Max-k-SAT follows, as Max-2-SAT is a subproblem of Max-k-SAT for $k \geq 2$. □

2.4.4 Max-DICUT inapproximability

The problem Max-DICUT asks for a maximum sized cut in a directed graph G, i.e., partitioning the vertex set $V(G)$ into two parts V_0 and V_1, such that the number of directed edges $(i,j) \in E(G)$ going from V_0 to V_1, i.e., $i \in V_0$ and $j \in V_1$ are maximal. We use a formulation similar to MaxCUT.

Definition 2.4.7 (Directed Cut)**.** For a fixed $n \in \mathbb{N}$, the problem Max-DICUT is the CSP with constraint family $\mathcal{C}_{DICUT} = \{\neg x_i \wedge x_j \mid i,j \in [n], i \neq j\}$.

We obtain $(1/2 + \Theta(\varepsilon))$-inapproximability via the standard reduction from undirected graphs, by replacing every edge with two, namely, one edge in either direction (see Corollary 2.4.8). The inapproximability factor is tight as the LP in Trevisan (1998, Page 84, Eq. (DI)), is $\frac{1}{2}$-approximate for maximum weighted directed cut. In the SDP case we obtain $15/16 + \Theta(\varepsilon)$-inapproximability and in Feige and Goemans (1995) it is shown that Max-DICUT can be approximated with a small SDP within a factor of 0.859 leaving a gap of about 0.08.

Table 2.11: Summary of the hardness results for Max-DICUT given in Corollary 2.4.8. The last line holds under the assumption of Conjecture 2.2.11.

	completeness	soundness	inapprox factor	size
fc_+	$1/2-\varepsilon$	$1/4+\varepsilon$	$1/2+\Theta(\varepsilon)$	$2^{n^{c(\varepsilon)}}$
$\mathrm{fc}_\oplus$	exact case		1	$2^{\Omega\left(n^{2/13}\right)}$
	$4/10-\varepsilon$	$3/8+\varepsilon$	$15/16+\Theta(\varepsilon)$	$n^{\Omega(\log n/\log\log n)}$
	$1/2-\varepsilon$	$c_{GW}/2+\varepsilon$	$c_{GW}+\Theta(\varepsilon)$	superpolynomial

Corollary 2.4.8. *For every $\varepsilon > 0$ there is a constant $c(\varepsilon)$ such that for infinitely many n, we have* $\mathrm{fc}_+(\text{Max-DICUT}, 1/2-\varepsilon, 1/4+\varepsilon) \geq 2^{n^{c(\varepsilon)}}$ *achieving inapproximability factor* $1/2+\Theta(\varepsilon)$. *In the exact SDP case we have* $\mathrm{fc}_\oplus(\text{Max-DICUT}) \geq 2^{\Omega\left(n^{2/13}\right)}$, *while in the approximate SDP case* $\mathrm{fc}_\oplus(\text{Max-DICUT}, 4/10-\varepsilon, 3/8+\varepsilon) \geq n^{\Omega(\log n/\log\log n)}$ *holds*

achieving an inapproximability factor of $15/16 + \Theta(\varepsilon)$ *and assuming Conjecture 2.2.11* $\mathrm{fc}_{\oplus}(\textsf{Max-DICUT}, 1/2 - \varepsilon, c_{GW}/2 + \varepsilon)$ *is superpolynomial establishing inapproximability factor* $c_{GW} + \Theta(\varepsilon)$.

Proof. The proof is analogous to that of Corollaries 2.4.4 and 2.4.6, hence we point out only the differences. We reduce MaxCUT to Max-DICUT, but now replace every clause $l = (x_i \oplus x_j = 1)$ with $C_1(l) = \neg x_i \wedge x_j$ and $C_2(l) = x_i \wedge \neg x_j$. Observe that whenever $x_i \oplus x_j = 1$ is satisfied by a partial assignment s then exactly one of $\neg x_i \wedge x_j$ and $x_i \wedge \neg x_j$ is also satisfied by s. If on the other hand $x_i \oplus x_j$ is not fulfilled by s, then neither $\neg x_i \wedge x_j$ nor $x_i \wedge \neg x_j$ are fulfilled. The remainder of the proof is the same as in Corollary 2.4.6. □

2.4.5 Minimum constraint satisfaction

In this section we examine minimum constraint satisfaction problems, a variant of constraint satisfaction problems, where the objective is not to maximize the number of *satisfied* constraints, but to minimize the number of *unsatisfied* constraints. This is equivalent to maximizing the number of satisfied constraints, however, the changed objective function yields different approximation factors due to the change in the magnitude of the optimum value; this is in analogy to the algorithmic world. We consider only Min-2-CNFDeletion and MinUnCUT from Agarwal et al. (2005), which are complete in their class in the algorithmic hierarchy; our technique applies to many more problems in Papadimitriou and Yannakakis (1991). The problem Min-2-CNFDeletion is of particular interest here, as it is considered to be the hardest minimum CSP with nontrivial approximation guarantees (see Agarwal et al. 2005). We start with the general definition of minimum CSPs.

Definition 2.4.9 (Minimum CSPs)**.** The *minimum Constraint Satisfaction Problem* on variables $x_1, \ldots, x_n$ with constraint family $\mathcal{C} = \{C_1, \ldots, C_m\}$ is the minimization problem with

(i) **feasible solutions** all 0/1 assignments to $x_1, \ldots, x_n$;

(ii) **instances** all nonnegative weightings $w_1, \ldots, w_m$ of the constraints $C_1, \ldots, C_m$;

(iii) **objective functions** weighted sum of negated constraints, i.e. $\mathrm{val}_{w_1,\ldots,w_m}(x_1, \ldots, x_n) = \sum_i w_i[1 - C_i(x_1, \ldots, x_n)]$.

The goal is to minimize the objective function, i.e., the weight of unsatisfied constraints.

As mentioned above, we consider two examples.

Definition 2.4.10 (Min-2-CNFDeletion and MinUnCUT)**.** The problem Min-2-CNFDeletion is the minimum CSP with constraint family consisting of all disjunction of two literals, as in Max-2-SAT. The problem MinUnCUT is the minimum CSP with constraint family consisting of all equations $x_i \oplus x_j = b$ with $b \in \{0, 1\}$, as in Max-2-XOR or MaxCUT.

We are ready to prove inapproximability bounds for these problems. Instead of the reductions in Chlebík and Chlebíková (2004), we use direct, simpler reductions from MaxCUT and here we provide reductions for general weights. Note that the current best known algorithmic inapproximability for Min-2-CNFDeletion is $8\sqrt{5} - 15 - \varepsilon \approx 2.88854 - \varepsilon$ by Chlebík and Chlebíková (2004). Assuming the Unique Games Conjecture, Chawla et al. (2006) establishes that Min-2-CNFDeletion cannot be approximated within any constant factor and our LP inapproximability factor coincides with this one. The problem MinUnCUT is known to be SNP-hard (see Papadimitriou and Yannakakis 1991). We refer the reader to Khanna et al. (2001) for a classification of all minimum CSPs.

Table 2.12: Summary of the hardness results for Min-2-CNFDeletion achieved by a reduction from MaxCUT which is described in Theorem 2.4.11. The last line in this table holds assuming Conjecture 2.2.11.

	completeness	soundness	inapprox factor	size
fc_+	ε	$1/4 - \varepsilon$	$\rho \geq 1$	$2^{n^{c(\varepsilon)}}$
$\mathrm{fc}_\oplus$	exact case		1	$2^{\Omega\left(n^{2/13}\right)}$
	$1/10 + \varepsilon$	$1/8 - \varepsilon$	$5/4 - \Theta(\varepsilon)$	$n^{\Omega(\log n / \log\log n)}$
	ε	$(1 - c_{GW})/2 - \varepsilon$	$\rho \geq 1$	superpolynomial

Table 2.13: Summary of the hardness results for MinUnCUT achieved by a reduction from MaxCUT which is described in Theorem 2.4.11. The last line in this table holds assuming Conjecture 2.2.11.

	completeness	soundness	inapprox factor	size
fc_+	ε	$1/2-\varepsilon$	$\rho \geq 1$	$2^{n^{c(\varepsilon)}}$
$\mathrm{fc}_\oplus$	exact case		1	$2^{\Omega\left(n^{2/13}\right)}$
	$1/5+\varepsilon$	$1/4-\varepsilon$	$5/4-\Theta(\varepsilon)$	$n^{\Omega(\log n/\log\log n)}$
	ε	$1-c_{GW}-\varepsilon$	$\rho \geq 1$	superpolynomial

Corollary 2.4.11. *For every $\varepsilon > 0$ there exists a constant $c(\varepsilon)$ such that for infinitely many n, we have* $\mathrm{fc}_+(\text{Min-2-CNFDeletion}, \varepsilon, 1/4-\varepsilon) = 2^{n^{c(\varepsilon)}}$ *and* $\mathrm{fc}_+(\text{MinUnCUT}, \varepsilon, 1/2-\varepsilon) = 2^{n^{c(\varepsilon)}}$ *establishing inapproximability within any constant factor, where n is the number of variables. In the exact SDP case we have* $\mathrm{fc}_\oplus(\text{Min-2-CNFDeletion}) = 2^{\Omega\left(n^{2/13}\right)}$ *and* $\mathrm{fc}_\oplus(\text{MinUnCUT}) = 2^{\Omega\left(n^{2/13}\right)}$. *In the approximate SDP case we have* $\mathrm{fc}_\oplus(\text{Min-2-CNFDeletion}, 1/10+\varepsilon, 1/8-\varepsilon) = \mathrm{fc}_\oplus(\text{MinUnCUT}, 1/5+\varepsilon, 1/4-\varepsilon) = n^{\Omega(\log n/\log\log n)}$ *showing inapproximability within $5/4-\Theta(\varepsilon)$ and further assuming Conjecture 2.2.11, we even have that* $\mathrm{fc}_\oplus(\text{Min-2-CNFDeletion}, \varepsilon, (1-c_{GW})/2-\varepsilon)$ *and* $\mathrm{fc}_\oplus(\text{MinUnCUT}, \varepsilon, 1-c_{GW}-\varepsilon)$ *are superpolynomial showing inapproximability within any constant factor.*

Proof. We reduce from MaxCUT to Min-2-CNFDeletion similar to the previous reductions: assignments are mapped to themselves, i.e., γ is the identity. Under β every clause C_ℓ is replaced with two disjunctive clauses $C_\ell(1)$ and $C_\ell(2)$, both inheriting the weight w_ℓ of C_ℓ, i.e., $\mathrm{val}^{\text{Min-2-CNFDeletion}}_{\beta(w_1,\ldots,w_m)}(x_1,\ldots,x_n) = \sum_\ell w_\ell[(1-C_\ell(1))+(1-C_\ell(2))]$.

For $C_\ell = (x_i \oplus x_j = 1)$, we set $C_\ell(1) := x_i \vee \neg x_j$ and $C_\ell(2) := \neg x_i \vee x_j$. Note that if C_ℓ is unsatisfied, then both of $C_\ell(1)$ and $C_\ell(2)$ are satisfied, and if C_ℓ is satisfied, then exactly one of $C_\ell(1)$ and $C_\ell(2)$ is satisfied. Therefore $\mathrm{val}^{\text{Min-2-CNFDeletion}}_{\beta(w_1,\ldots,w_m)} = \sum_\ell w_\ell - \mathrm{val}^{\text{MaxCUT}}_{w_1,\ldots,w_m} w_\ell$. Corollaries 2.2.5 and 2.2.6 with parameters $\alpha = -1$, $\mu = 1$ and $\eta = 2$ provide the desired lower bounds.

For MinUnCUT the reduction is similar but simpler, as we replace every clause C with

itself. In this case the parameters for Corollaries 2.2.5 and 2.2.6 are $\alpha = -1$, $\mu = 1$ and $\eta = 1$. □

2.5 Inapproximability of Graph-Isomorphism

Graph-Isomorphism is of particularly interesting since it is besides Integer-Factorization the only problem from Garey and Johnson (1979), for which it is still unknown whether it is in P of NP-complete. Recently it has been shown that Graph-Isomorphism is solvable in quasi-polynomial time by Babai (2015). It is also known that Graph-Isomorphism is in the low hierarchy of the class NP (see Schöning 1988) and that it is equally hard to the isomorphism problem between many other objects, including finite automata, context-free grammars, Markov decision processes, and vertex-facet incidence relations of convex polytopes.

We discuss several different optimization versions of Graph Homomorphism and Graph Isomorphism problems in this section. All of them are special cases of the following Graph-Morphism problem.

Definition 2.5.1 (Graph-Morphism)**.** For two fixed positive integers n_1, n_2, the Graph-Morphism problem has

(i) **feasible solutions:** all maps π from $[n_1]$ to $[n_2]$;

(ii) **instances:** for every pair of graphs (G_1, G_2) with $V(G_1) = [n_1]$ and $V(G_2) = [n_2]$ consider the family $\{C_{uv} \mid (u,v) \in \mathcal{C}\}$ of constraints C_{uv} indexed by some set $\mathcal{C} \subseteq \{\{u,v\} \mid u,v \in [n_1], u \neq v\}$ where C_{uv} is the following constraint on π:

$$\{u,v\} \in E(G_1) \text{ if and only if } \{\pi(u), \pi(v)\} \in E(G_2);$$

(iii) **objective function:** In the maximization version the objective function $f_{(G_1,G_2)}$ represents the number of satisfied constraints. In the minimization version the objective function $f_{(G_1,G_2)}$ represents the number of unsatisfied constraints.

The size of the problem is measured in $m = \max\{n_1, n_2\}$.

Depending on the properties of π and the set of constraints $\mathcal{C}$ there are the following versions of the Graph-Morphism problem:

Table 2.14: Different versions of Graph Homomorphism and Graph Isomorphism problems depending on the constraint set and if π is a permutation or not. We write $|V(G_1)| = n_1$ and $|V(G_2)| = n_2$.

	$\mathcal{C} = E(G_1)$	$\mathcal{C} = \{\{u,v\} \mid u,v \in [n_1], u \neq v\}$
π permutation ($n_1 = n_2$)	Edge-Graph-Isomorphism (EGI)	Pair-Graph-Isomorphism (PGI)
π arbitrary	Graph-Homomorphism (GH)	

In the case where $n_1 = n_2$ and π is a permutation, we additionally define the Graph Isomorphism problems *with color class constraints*.

Definition 2.5.2 (Graph-Isomorphism with color class constraints)**.** Given $m, k \in \mathbb{N}$ and a partition $\biguplus V_i = [m]$ with $|V_i| \leq k$ the Graph-Isomorphism problem with color class constraints has

(i) **feasible solutions:** all permutations π of $[m]$ respecting the partition $\{V_i\}_i$, i.e., $\pi(V_i) = V_i$;

(ii) **instances:** the same as given in Definition 2.5.1;

(iii) **objective function:** the same as given in Definition 2.5.1.

We denote these problems by EGI_k and PGI_k.

Approximating both the colored and uncolored variants of EGI, PGI, and GH is shown to be NP-hard in Arvind et al. (2012).

While evidently e.g., $\mathrm{fc}_+(\text{Max-EGI}_l, C, S) \leq \mathrm{fc}_+(\text{Max-EGI}_k, C, S)$ for $l \leq k$, for the sake of exposition we shall provide direct arguments for Max-PGI_k and other colored problems for a general k instead of just $k = 2$.

2.5.1 Colored graph problems

We establish hardness results for the colored graph problems in this sections, first for the maximization then for the minimization problems.

Maximizing colored graph problems

We will now prove a lower bound on the formulation complexity of Max-PGI$_k$ with approximation guarantee $\frac{1}{2}+\varepsilon$. The proof is by reduction from MaxCUT, which was first used in Arvind et al. (2012, Lemma 11).

Table 2.15: Summary of the hardness results for Max-PGI$_l$ with $l \geq 2$ given in Corollary 2.5.3. The completeness and soundness guarantees are given as an absolute number as opposed to as a fraction of the size of the instance. G_2 is here the target graph in an Max-PGI$_l$ instance $f_{(G_1,G_2)}$. The last line holds under the assumption of Conjecture 2.2.11.

	completeness	soundness	inapprox factor	size
fc_+	$(2-\varepsilon)\,\lvert E(G_2)\rvert$	$(1+\varepsilon)\,\lvert E(G_2)\rvert$	$1/2+\Theta(\varepsilon)$	$2^{m^{c(\varepsilon)}}$
$\mathrm{fc}_\oplus$	exact case		1	$2^{\Omega\left(m^{2/13}\right)}$
	$(8/5-\varepsilon)\,\lvert E(G_2)\rvert$	$(6/4+\varepsilon)\,\lvert E(G_2)\rvert$	$15/16+\Theta(\varepsilon)$	$m^{\Omega(\log m/\log\log m)}$
	$(2-\varepsilon)\,\lvert E(G_2)\rvert$	$(2c_{GW}+\varepsilon)\,\lvert E(G_2)\rvert$	$c_{GW}+\Theta(\varepsilon)$	superpolynomial

Corollary 2.5.3. *Let $l \geq 2$ and $\varepsilon > 0$. Then there is a constant $c(\varepsilon)$ such that for infinitely many m, in the linear case* $\mathrm{fc}_+\left(\text{Max-PGI}_l, (2-\varepsilon)\,\lvert E(G_2)\rvert, (1+\varepsilon)\,\lvert E(G_2)\rvert\right) = 2^{m^{c(\varepsilon)}}$. *In the SDP case we have* $\mathrm{fc}_\oplus\left(\text{Max-PGI}_l\right) = 2^{\Omega\left(m^{2/13}\right)}$, $\mathrm{fc}_\oplus(\text{Max-PGI}_l, (8/5-\varepsilon)\,\lvert E(G_2)\rvert, (6/4+\varepsilon)\,\lvert E(G_2)\rvert) = m^{\Omega(\log m/\log\log m)}$ *and that* $\mathrm{fc}_\oplus(\text{Max-PGI}_l, (2-\varepsilon)\,\lvert E(G_2)\rvert, (2c_{GW}+\varepsilon)\,\lvert E(G_2)\rvert)$ *is superpolynomial under Conjecture 2.2.11.*

Proof. We shall apply Corollary 2.2.7 and thus define reduction maps from MaxCUT on n variables to Max-PGI$_{2k}$ on $[2nk]$ with n color classes V_i each of size $2k$. For the reduction we fix a partition $V_i = V_i^0 \cup V_i^1$ of every color class V_i into two sets V_i^0, V_i^1 each of size k.

Given an instance sat_L of MaxCUT for some set of constraints $L = \{x_i \oplus x_j = 1\}$, we construct colored graphs G_1 and G_2 as follows, which together define the instance $f_{(G_1,G_2)}$ of Max-PGI$_{2k}$:

Both graphs have vertex set $[2nk]$ with color classes V_i. The edges of G_1 are

- all edges (u,v) inside a color class V_i, i.e., $u,v \in V_i$
- all edges (u,v) between two color classes V_i and V_j (i.e., $u \in V_i$ and $v \in V_j$) with $(x_i \oplus x_j = 1) \notin L$ and $i \neq j$
- all edges (u,v) with either $u \in V_i^0$ and $v \in V_j^0$ or $u \in V_i^1$ and $v \in V_j^1$ with $(x_i \oplus x_j = 1) \in L$ and $i \neq j$

The edges of G_2 are all edges (u,v) with either $u \in V_i^0$ and $v \in V_j^1$ or $u \in V_i^1$ and $v \in V_j^0$ with $(x_i \oplus x_j = 1) \in L$ and $i \neq j$.

Now we define the reduction map γ between feasible solutions. Given a feasible solution of MaxCUT i.e. an assignment to the n variables. The isomorphism π is constructed as follows: If 0 is assigned to x_i, then π maps V_i^0 to V_i^0 and V_i^1 to V_i^1. (The exact way of mapping is irrelevant.) If 1 is assigned to x_i, then π maps V_i^0 of G_1 to V_i^1 of G_2 and V_i^1 of G_1 to V_i^0 of G_2.

We shall verify Eq. (2.10). We observe that for any vertices $u \in V_i$ and $v \in V_j$, the constraint C_{uv} is satisfied if and only if $i \neq j$, $(x_i \oplus x_j = 1) \in L$ and $x_i \oplus x_j = 1$ is satisfied. Thus $\mathrm{sat}_L(x) = \frac{1}{(2k)^2} f_{(G_1,G_2)}(\pi)$. To see exactness of this reduction, let π be a colored vertex map, and let $m_i = m_i(\pi)$ denote the number of vertices of V_i^0 which are mapped to V_i^1. Observe that if $m_i \in \{0,k\}$, then $\pi = \gamma(x)$ for the assignment given by $x_i := 0$ if $m_i = 0$ and $x_i := 1$ if $m_i = k$. In the general case, we subsequently modify π to make all the m_i either 0 or k without decreasing $f_{G_1,G_2}(\pi)$, which will clearly prove exactness.

Therefore let i be an index with $0 < m_i < k$. We define two permutations π_0 and π_1 of $[2nk]$, as candidates for modifications of π: They coincide with π outside of $V_i^0 \cup V_i^1$. The map π_0 maps V_i^0 to V_i^0 and V_i^1 to V_i^1, while π_1 maps V_i^0 to V_i^1 and V_i^1 to V_i^0. Obviously,

$m_j(\pi_0) = m_j(\pi_1) = m_j(\pi)$ for $i \neq j$, but $m_i(\pi_0) = 0$ and $m_i(\pi_1) = k$, where for clarity, the right permutation is added after m_i. We will show that

$$f_{(G_1,G_2)}(\pi) = \frac{m_i}{k} f_{(G_1,G_2)}(\pi_0) + \frac{k - m_i}{k} f_{(G_1,G_2)}(\pi_1), \tag{2.15}$$

which by convexity gives $\max\{f_{(G_1,G_2)}(\pi_0), f_{(G_1,G_2)}(\pi_1)\} \geq f_{(G_1,G_2)}(\pi)$. So by choosing the maximum we obtain a permutation with one more m_j being 0 or k, as claimed.

To show Equation (2.15). We count the correctly mapped pairs having one vertex in $V_i^0 \cup V_i^1$ and one vertex in $V_j^0 \cup V_j^1$, that

$$f_{(G_1,G_2)}(\pi) = \sum_{\substack{j<l \\ (x_j \oplus x_l = 1) \in L}} 4\left[m_j(k - m_l) + (k - m_j)m_l\right].$$

Since this is linear in m_i, Equation (2.15) follows.

Therefore we can apply Corollary 2.2.7 with $\alpha = 4k^2$ and together with the hardness of MaxCUT summarized in Table 2.1 the results follow. □

Now we prove an analogous result for colored edge graph isomorphisms.

Table 2.16: Summary of the hardness results for Max-EGI$_l$ with $l \geq 2$ given in Corollary 2.5.4. The completeness and soundness guarantees are given as absolute numbers, where G_2 is the target graph in an Max-EGI$_l$ instance $f_{(G_1,G_2)}$. The last line holds under the assumption of Conjecture 2.2.11.

	completeness	soundness	inapprox factor	size
fc_+	$(1-\varepsilon)\lvert E(G_2)\rvert$	$(1/2+\varepsilon)\lvert E(G_2)\rvert$	$1/2+\Theta(\varepsilon)$	$2^{m^{c(\varepsilon)}}$
$\mathrm{fc}_\oplus$	exact case		1	$2^{\Omega\left(m^{2/13}\right)}$
	$(4/5-\varepsilon)\lvert E(G_2)\rvert$	$(3/4+\varepsilon)\lvert E(G_2)\rvert$	$15/16+\Theta(\varepsilon)$	$m^{\Omega(\log m/\log\log m)}$
	$(1-\varepsilon)\lvert E(G_2)\rvert$	$(c_{GW}+\varepsilon)\lvert E(G_2)\rvert$	$c_{GW}+\Theta(\varepsilon)$	superpolynomial

Corollary 2.5.4. *Let $l \geq 2$ and $\varepsilon > 0$. Then there exists a constant $c(\varepsilon)$ such that for infinitely many m, we have* $\mathrm{fc}_+(\text{Max-EGI}_l, (1-\varepsilon)\lvert E(G_2)\rvert, (1/2+\varepsilon)\lvert E(G_2)\rvert) = 2^{m^{c(\varepsilon)}}$ *in*

the linear case and $\mathrm{fc}_{\oplus}(\textsf{Max-EGI}_l) = 2^{\Omega(m^{2/13})}$, $\mathrm{fc}_{\oplus}(\textsf{Max-EGI}_l, (4/5-\varepsilon)\,|E(G_2)|, (3/4+\varepsilon)\,|E(G_2)|) = m^{\Omega(\log m/\log\log m)}$ *and that assuming Conjecture 2.2.11* $\mathrm{fc}_{\oplus}(\textsf{Max-EGI}_l, (1-\varepsilon)\,|E(G_2)|, (c_{GW}+\varepsilon)\,|E(G_2)|)$ *is superpolynomial in the SDP case.*

Proof. We reduce MaxCUT to $\textsf{Max-EGI}_{2k}$ as in Arvind et al. (2012, Lemma 12). The maps β and γ are the same as defined in the proof of Corollary 2.5.3. The analysis is also identical except for the fact that we only count edges instead of all pairs. Equation (2.10) is in this case

$$\mathrm{sat}_L(x) = \frac{1}{2k^2} f_{(G_1,G_2)}(\pi).$$

Applying Corollary 2.2.7 with $\alpha = 2k^2$ together with the hardness of MaxCUT summarized in Table 2.1 and the results follow. □

Minimizing colored graph problems

Here we prove the corresponding results of Corollaries 2.5.3 and 2.5.4 for minimization problems, i.e., $\textsf{Min-PGI}_{2k}$ and $\textsf{Min-EGI}_{2k}$. While the reductions are analogous, they are not exactly the same: the constructions in Corollaries 2.5.3 and 2.5.4 resulted in many additional constraints which were never satisfied. In the minimization case we have to adjust them, so that they are always satisfied, i.e., they do not contribute to the objective function counting the *unsatisfied constraints*. Therefore some edges are inverted.

Table 2.17: Summary of the hardness results for $\textsf{Min-PGI}_l$ with $l \geq 2$ given in Corollary 2.5.5. The completeness and soundness guarantees are given as absolute numbers, where G_2 is the target graph in an $\textsf{Min-PGI}_l$ instance $f_{(G_1,G_2)}$. The last line holds under the assumption of Conjecture 2.2.11.

	completeness	soundness	inapprox factor	size
fc_{+}	$\varepsilon\,\lvert E(G_2)\rvert$	$(1-\varepsilon)\,\lvert E(G_2)\rvert$	$\rho \geq 1$	$2^{m^{c(\varepsilon)}}$
$\mathrm{fc}_{\oplus}$	exact case		1	$2^{\Omega\left(m^{2/13}\right)}$
	$(2/5+\varepsilon)\,\lvert E(G_2)\rvert$	$(2/4-\varepsilon)\,\lvert E(G_2)\rvert$	$5/4-\Theta(\varepsilon)$	$m^{\Omega(\log m/\log\log m)}$
	$\varepsilon\,\lvert E(G_2)\rvert$	$(2-2c_{GW}-\varepsilon)\,\lvert E(G_2)\rvert$	$\rho \geq 1$	superpolynomial

Corollary 2.5.5. *Let $\varepsilon \geq 0$ and $l \in \mathbb{N}$ with $l \geq 2$. Then there exists a constant $c(\varepsilon)$ such that for infinitely many m, we have* $\mathrm{fc}_{+}(\textsf{Min-PGI}_l, \varepsilon\,|E(G_2)|\,, (1-\varepsilon)\,|E(G_2)|) = 2^{m^{c(\varepsilon)}}$. *The exact SDP hardness is given by* $\mathrm{fc}_{\oplus}(\textsf{Min-PGI}_l) = 2^{\Omega\left(m^{2/13}\right)}$ *while in the approximate case we have* $\mathrm{fc}_{\oplus}(\textsf{Min-PGI}_l, (2/5+\varepsilon)\,|E(G_2)|\,, (2/4-\varepsilon)\,|E(G_2)|) = m^{\Omega(\log m/\log\log m)}$ *and that under Conjecture 2.2.11* $\mathrm{fc}_{\oplus}(\textsf{Min-PGI}_l, \varepsilon\,|E(G_2)|\,, (2-2c_{GW}-\varepsilon)\,|E(G_2)|)$ *is superpolynomial.*

Proof. We shall reduce MinUnCUT to $\textsf{Min-PGI}_l$ employing Corollary 2.2.7. The reduction is along the lines of the proof of Corollary 2.5.3, and comes from Arvind et al. (2012, Lemma 15). Given an objective function $\overline{\mathrm{sat}}_L(G)$ we construct $f_{(G_1,G_2)}$ as follows: For every variable x_i we have one color class $V_i^0(G_1) \cup V_i^1(G_1)$ in G_1 and one color class $V_i^0(G_2) \cup V_i^1(G_2)$ in G_2, each of size $2k$, i.e., $|V_i^0(G_1)| = |V_i^1(G_1)| = |V_i^0(G_2)| = |V_i^1(G_2)| = k$, and these are all vertices of G_1 and G_2. Within each color class there are neither edges in G_1 nor in G_2. Whenever $(x_i \oplus x_j = 0) \notin L(G)$ there are no edges between the color classes corresponding to x_i and x_j. If $(x_i \oplus x_j = 0) \in L(G)$, then all edges between $V_i^0(G_1)$ and $V_j^0(G_1)$ and between $V_i^1(G_1)$ and $V_j^1(G_1)$ are present in G_1. In G_2 all edges between $V_i^0(G_2)$ and $V_j^1(G_2)$ and between $V_i^1(G_2)$ and $V_j^0(G_2)$ are there.

An assignment s to the variables $x_1, \ldots, x_n$ is mapped to π_s by γ in the following way: If $s_i = 0$ then $\pi(V_i^0(G_1)) = V_i^0(G_2)$ and $\pi(V_i^1(G_1)) = V_i^1(G_2)$. If $s_i = 1$ then $\pi(V_i^0(G_1)) = V_i^1(G_2)$ and $\pi(V_i^1(G_1)) = V_i^0(G_2)$.

To see that these maps satisfy the requirements of Corollary 2.2.7, we observe that whenever a constraint $(x_i \oplus x_j = 0) \in L(G)$ is satisfied by an assignment s, then all $(2k)^2$ pairs between the color classes corresponding to i and j are *not* mapped correctly. Is $(x_i \oplus x_j = 0) \in L(G)$ on the other hand not satisfied, then all pairs between $V_i^0(G_1) \cup V_i^1(G_1)$ and $V_j^0(G_1) \cup V_j^1(G_1)$ are mapped correctly. Recall that $f_{(G_1,G_2)}$ as an objective function of $\textsf{Min-PGI}_{2k}$ counts the number of *not* correctly mapped pairs. Formally we get:

$$\overline{\mathrm{sat}}_{L(G)}(s) = \frac{1}{(2k)^2} f_{(G_1,G_2)}(\pi_s).$$

Equation 2.10 follows from a convexity argument analogous to the proof of Corollary 2.5.3, hence omitted. The only difference is that now we have $f_{G_1,G_2}(\pi) = \sum_{j \neq i} 4m_i m_j + 4(k - m_i)(k - m_j)$.

Corollary 2.2.7 with $\alpha = 4k^2$ and the hardness of MinUnCUT in Corollary 2.4.11 imply the desired results. □

The results for edge graph isomorphism are similar.

Table 2.18: Summary of the hardness results for Min-EGI$_l$ with $l \geq 2$ given in Corollary 2.5.6. Observe that the completeness and soundness guarantees are given as absolute numbers, where G_2 is the target graph in an Min-PGI$_l$ instance $f_{(G_1,G_2)}$. The last line holds under the assumption of Conjecture 2.2.11.

	completeness	soundness	inapprox factor	size
fc_+	$\varepsilon \lvert E(G_2)\rvert$	$(1/2 - \varepsilon) \lvert E(G_2)\rvert$	$\rho \geq 1$	$2^{m^{c(\varepsilon)}}$
$\mathrm{fc}_\oplus$	exact case		1	$2^{\Omega\left(m^{2/13}\right)}$
	$(1/5 + \varepsilon) \lvert E(G_2)\rvert$	$(1/4 - \varepsilon) \lvert E(G_2)\rvert$	$5/4 - \Theta(\varepsilon)$	$m^{\Omega(\log m / \log\log m)}$
	$\varepsilon \lvert E(G_2)\rvert$	$(1 - c_{GW} - \varepsilon) \lvert E(G_2)\rvert$	$\rho \geq 1$	superpolynomial

Corollary 2.5.6. *Let $\varepsilon \geq 0$ and $l \in \mathbb{N}$ with $l \geq 2$. Then there exists a constant $c(\varepsilon)$ such that for infinitely many m, we have in the linear case* $\mathrm{fc}_+(\text{Min-EGI}_l, \varepsilon \lvert E(G_2)\rvert, (1/2 - \varepsilon) \lvert E(G_2)\rvert) = 2^{m^{c(\varepsilon)}}$. *In the SDP case the hardness is given by* $\mathrm{fc}_\oplus(\text{Min-EGI}_l) = 2^{\Omega\left(m^{2/13}\right)}$, $\mathrm{fc}_\oplus(\text{Min-EGI}_l, (1/5 + \varepsilon) \lvert E(G_2)\rvert, (1/4 - \varepsilon) \lvert E(G_2)\rvert) = m^{\Omega(\log m / \log\log m)}$ *and, under Conjecture 2.2.11, by* $\mathrm{fc}_\oplus(\text{Min-EGI}_l, \varepsilon \lvert E(G_2)\rvert, (1 - c_{GW} - \varepsilon) \lvert E(G_2)\rvert)$ *being superpolynomial.*

Proof. The proof is analogous to the proof of Theorem 2.5.5 with the main difference that one writes $2k^2$ instead of $(2k)^2$, and hence omitted. The reduction is again from Arvind et al. (2012, Lemma 15). □

2.5.2 Uncolored graph problems

We summarize the hardness results of this section in Figure 2.19 for Max-EGI and in Figure 2.20 for Min-EGI. The results for the linear formulation complexity of Max-EGI look very similar, however there is no strict relation between the two: the result achieved through the reduction from MaxCUT has a better approximation factor, however the size is smaller than the one coming from Max-Matching.

Table 2.19: Summary of the hardness results for Max-EGI. The second line is shown in Corollary 2.5.7 by a reduction from Max-Matching while the other lines are shown Corollary 2.5.9 by a reduction from MaxCUT over Max-GH. Observe that the completeness and soundness guarantees are given as absolute numbers, where G_1 is the source graph in an Max-EGIinstance $f_{(G_1,G_2)}$. The last line holds under the assumption of Conjecture 2.2.11.

	completeness	soundness	inapprox factor	size
fc_+	$(1-\varepsilon)\lvert E(G_1)\rvert$	$(1/2+\varepsilon)\lvert E(G_1)\rvert$	$1/2+\Theta(\varepsilon)$	$2^{m^{c(\varepsilon)}}$
	$(m+\varepsilon)/2$	$m/2$	$1-\varepsilon/m$	$2^{\Theta(m)}$
$\mathrm{fc}_\oplus$	exact case		1	$2^{\Omega\left(m^{2/13}\right)}$
	$(4/5-\varepsilon)\lvert E(G_1)\rvert$	$(3/4+\varepsilon)\lvert E(G_1)\rvert$	$15/16+\Theta(\varepsilon)$	$m^{\Omega(\log m/\log\log m)}$
	$(1-\varepsilon)\lvert E(G_1)\rvert$	$(c_{GW}+\varepsilon)\lvert E(G_1)\rvert$	$c_{GW}+\Theta(\varepsilon)$	superpolynomial

Table 2.20: Summary of the hardness results for Min-EGI and Min-GH. The results for the latter problem are shown in Corollary 2.5.8 by a reduction from MinUnCUT. The results for Min-EGI follow by a reduction from Min-GH shown in Corollary 2.5.9. The completeness and soundness guarantees are given as absolute numbers, where G_1 is the source graph in an Max-EGIinstance $f_{(G_1,G_2)}$. The last line holds under the assumption of Conjecture 2.2.11.

	completeness	soundness	inapprox factor	size
fc_+	$\varepsilon\lvert E(G_1)\rvert$	$(1/2-\varepsilon)\lvert E(G_1)\rvert$	$\rho\geq 1$	$2^{m^{c(\varepsilon)}}$
$\mathrm{fc}_\oplus$	exact case		1	$2^{\Omega\left(m^{2/13}\right)}$
	$(1/5+\varepsilon)\lvert E(G_1)\rvert$	$(1/4-\varepsilon)\lvert E(G_1)\rvert$	$5/4-\Theta(\varepsilon)$	$m^{\Omega(\log m/\log\log m)}$
	$\varepsilon\lvert E(G_1)\rvert$	$(1-c_{GW}-\varepsilon)\lvert E(G_1)\rvert$	$\rho\geq 1$	superpolynomial

Matching and edge graph isomorphism

The Max-Matching problem has the remarkable property of having a polynomial time algorithm, and yet having large LP formulation complexity.

Corollary 2.5.7. *Let* $0 < \varepsilon < 1$ *be a fixed number and* $m \in \mathbb{N}$ *be even. Then* $\mathrm{fc}_+(\text{Max-EGI}, (m+\varepsilon)/2, m/2) = 2^{\Theta(m)}$.

Proof. We reduce from Max-Matching and employ Corollary 2.2.7 together with the hardness results established in Section 2.1.1. Let H be a graph on n vertices with edge set a perfect matching. The reduction on objective functions is given by $\beta(\mathrm{sat}_G) := f_{(G,H)}$, i.e., the matching problem for a graph G is reduced to the homomorphism problem for G and H. The reduction of feasible solutions is as follows. Given a perfect matching M of K_n, we consider the subgraph H_M of K_n with edge set M on all the vertices. We choose $\pi = \gamma(M)$ to be an arbitrary graph isomorphism between H_M and H and consider this as a vertex map between G and H using $V(G) = V(H_M)$ (see Figure 2.4).

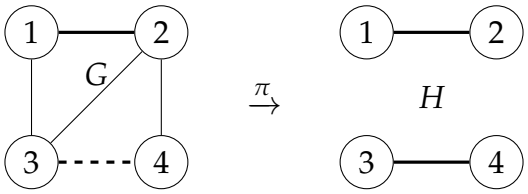

Figure 2.4: Reduction of the matching problem of a graph G to the homomorphism problem of G into H. The thick lines indicate a perfect matching, where the dashed line is not in G. The map π is the identity.

Clearly, $f_{(G,H)}(\pi) = |M \cap E(G)|$ is the number of edges preserved by π, showing that β, γ is a reduction. For exactness (Equation 2.10), observe that for any permutation σ of $[n]$, the perfect matching $M := \sigma^{-1}(E(H))$ of K_n clearly satisfies $|M \cap E(G)| = f_{(G,H)}(\sigma)$.

The two maps β and γ are compatible with Corollary 2.2.7 with $\alpha = 1$ and the results follow with the hardness results from Theorem 2.1.10. □

MaxCUT and Edge graph isomorphism

Now we prove hardness of Edge-Graph-Isomorphism using the hardness of MaxCUT and MinUnCUT for the maximization and minimization versions respectively. As a preparation, we first lower bound the formulation complexity of Graph-Homomorphism, serving as an intermediate problem in the reduction.

Corollary 2.5.8. *Let $\varepsilon > 0$. Then there exists a constant $c(\varepsilon)$ such that for infinitely many m, we have the lower bounds* $\mathrm{fc}_+(\text{Max-GH}, (1-\varepsilon)\,|E(G_1)|\,, (1/2+\varepsilon)\,|E(G_1)|) = 2^{m^{c(\varepsilon)}}$ *and* $\mathrm{fc}_+(\text{Min-GH}, \varepsilon\,|E(G_1)|\,, (1/2-\varepsilon)\,|E(G_1)|) = 2^{m^{c(\varepsilon)}}$ *for the linear case,* $\mathrm{fc}_\oplus(\text{Max-GH}) = 2^{\Omega\left(m^{2/13}\right)}$ *and* $\mathrm{fc}_\oplus(\text{Min-GH}) = 2^{\Omega\left(m^{2/13}\right)}$ *in the exact SDP case, and* $\mathrm{fc}_\oplus(\text{Max-GH}, (4/5-\varepsilon)\,|E(G_1)|, (3/4+\varepsilon)\,|E(G_1)|) = m^{\Omega(\log m/\log\log m)}$ *and* $\mathrm{fc}_\oplus(\text{Min-GH}, (1/5+\varepsilon)\,|E(G_1)|, (1/4-\varepsilon)\,|E(G_1)|) = m^{\Omega(\log m/\log\log m)}$ *in the approximate SDP case. Further assuming Conjecture 2.2.11 holds,* $\mathrm{fc}_\oplus(\text{Max-GH}, (1-\varepsilon)\,|E(G_1)|, (c_{GW}+\varepsilon)\,|E(G_1)|)$ *and* $\mathrm{fc}_\oplus(\text{Min-GH}, \varepsilon\,|E(G_1)|, (1-c_{GW}-\varepsilon)\,|E(G_1)|)$ *are superpolynomial.*

Proof. We reduce MaxCUT and MinUnCUT to Max-GH and Min-GH respectively, using the same reduction. Let $H = K_2$ be the graph with two vertices v_1 and v_2 and an edge between them. Then β maps the function $\mathrm{sat}_{L(G)}$ to $f_{(G,H)}$. γ maps an assignment $x \in \{0,1\}^n$ to the map that sends all vertices v with $x_v = 0$ to v_1 and those with $x_v = 1$ to v_2.

This is easily seen to satisfy the conditions of Corollary 2.2.7, as a constraint C_e is fulfilled for MaxCUT or MinUnCUT if and only if C_e is fulfilled for Max-GH or Min-GH, respectively. The exactness relation (Equation 2.10) holds, since γ is a bijection. The results follow with Corollary 2.2.7 with $\alpha = 1$ and the hardness results for MaxCUT (Table 2.1) and MinUnCUT (Table 2.13) together with Conjecture 2.2.11. □

Now we establish hardness of Edge-Graph-Isomorphism via a reduction from Graph-Homomorphism.

Corollary 2.5.9. *Let $\varepsilon > 0$. Then there exists a constant $c(\varepsilon)$ such that for infinitely many m, the hardness of* Max-EGI *and* Min-EGI *is given by* $\mathrm{fc}_+(\text{Max-EGI}, (1-\varepsilon)\,|E(G_1)|,$

$(1/2+\varepsilon)\,|E(G_1)|) = 2^{m^{c(\varepsilon)}}$ *and* $\mathrm{fc}_{+}(\textsf{Min-EGI}, \varepsilon\,|E(G_1)|, (1/2-\varepsilon)\,|E(G_1)|) = 2^{m^{c(\varepsilon)}}$ *for the linear case,* $\mathrm{fc}_{\oplus}(\textsf{Max-EGI} = 2^{\Omega\left(m^{2/13}\right)}$ *and* $\mathrm{fc}_{\oplus}(\textsf{Min-EGI} = 2^{\Omega\left(m^{2/13}\right)}$ *in the exact SDP case, and* $\mathrm{fc}_{\oplus}(\textsf{Max-EGI}, (4/5-\varepsilon)\,|E(G_1)|, (3/4+\varepsilon)\,|E(G_1)|) = m^{\Omega(\log m/\log\log m)}$ *and* $\mathrm{fc}_{\oplus}(\textsf{Min-EGI}, (1/5+\varepsilon)\,|E(G_1)|\,, (1/4-\varepsilon)\,|E(G_1)|) = m^{\Omega(\log m/\log\log m)}$ *in the approximate SDP case. Further under Conjecture 2.2.11,* $\mathrm{fc}_{\oplus}(\textsf{Max-EGI}, (1-\varepsilon)\,|E(G_1)|, (c_{GW}+\varepsilon)\,|E(G_1)|)$ *and* $\mathrm{fc}_{\oplus}(\textsf{Min-EGI}, \varepsilon\,|E(G_1)|, (1-c_{GW}-\varepsilon)\,|E(G_1)|)$ *are superpolynomial.*

Proof. We use the same reduction for the minimization and maximization problem, adapted from the proof of Arvind et al. (2012, Lemma 10). We define the map β to map a given objective function $f_{(G_1,G_2)}$ of Max-GH (Min-GH) to the objective function $f_{(H_1,H_2)}$ of Max-EGI (Min-EGI), where we define H_1 and H_2 as follows. The vertex sets are $V(H_1) = V(H_2) := V(G_1) \times V(G_2)$. Let $w_2 \in V(G_2)$ be any for the whole reduction fixed vertex of G_2. The edges of H_1 are in one-to-one correspondence with the edges of G_1: for an edge between vertices u_1 and v_1 of G_1, the graph H_1 has an edge between (u_1, w_2) and (v_1, w_2). The graph H_2 has an edge between two vertices (v_1, u_2) and (v_1', v_2) if and only if G_2 has an edge between u_2 and v_2.

We now define γ: Given $\phi\colon V_1 \to V_2$ we construct the permutation $\pi\colon V_1 \times V_2 \to V_1 \times V_2$ as follows.

$$\pi((u_1, w_2)) := (u_1, \phi(u_1)) \quad \forall u_1 \in V_1$$
$$\pi((u_1, u_2)) := \text{arbitrary} \quad \forall u_1 \in V_1, w_2 \neq u_2 \in V_2$$

We prove that this is a reduction: If an edge $\{u_1, v_1\} \in E_1$ is preserved by ϕ, the corresponding edge $\{(u_1, w_2), (v_1, w_2)\} \in E(H_1)$ is mapped to the edge $\{(u_1, \phi(u_1)), (v_1, \phi(v_1))\} \in E(H_2)$ by π and therefore also preserved. Thus ϕ and π preserve the same number of edges. Since H_1 has the same number of edges as G_1, also the number of not preserved edges of ϕ and π coincide, i.e., $f_{(G_1,G_2)}(\phi) = f_{(H_1,H_2)}(\pi)$, in both the maximization version and the

minimization version.

To show exactness, let π be any permutation between $V(H_1)$ and $V(H_2)$. We define $\phi\colon V(G_1) \to V(G_2)$ by letting $\phi(u_1)$ to be the second component of $\pi((u_1, w_2))$ for every $u_1 \in V(G_1)$. Clearly, $f_{G_1,G_2}(\phi) = f_{(H_1,H_2)}(\pi)$. Hence $\max f_{(G_1,G_2)} = \max f_{(H_1,H_2)}$ and $\min f_{(G_1,G_2)} = \min f_{(H_1,H_2)}$, (even though we need only one of them).

Applying Corollary 2.2.7 with $\alpha = 1$, Corollary 2.5.8 and Conjecture 2.2.11 finishes the proof. □

2.6 From matrix approximation to problem approximations

We will now explain how a nonnegative matrix with small nonnegative rank (or semidefinite rank) that is close to a slack matrix of a problem $\mathcal{P}$ of interest can be rounded to an actual slack matrix with a moderate increase in nonnegative rank (or semidefinite rank) and error. This argument is implicitly contained in Rothvoß (2013) and Briët, Dadush, and Pokutta (2015) for the linear and semidefinite case respectively. In some sense we might want to think of this approach as an interpolation between a slack matrix (which corresponds to $\mathcal{P}$) and a close-by matrix of low nonnegative rank (or semidefinite rank) that does not correspond to any optimization problem. The result is a low nonnegative rank (or semidefinite rank) approximation of $\mathcal{P}$ with small error.

We will need the following simple lemma. Recall that the exterior algebra of a vector space V is the $\mathbb{R}$-algebra generated by V subject to the relations $v^2 = 0$ for all $v \in V$. As is customary, the product in this algebra is denoted by $\wedge$. The subspace of homogeneous degree-k elements (i.e., linear combination of elements of the form $v_1 \wedge \cdots \wedge v_k$ with $v_1, \ldots, v_k \in V$) is denoted by $\bigwedge^k V$. Recall that for $k = \dim V$, the space $\bigwedge^k V$ is one dimensional and is generated by $v_1 \wedge \cdots \wedge v_k$ for any basis $v_1, \ldots, v_k$ of V.

Lemma 2.6.1. *Let $M \in \mathbb{R}^{m\times n}$ be a real matrix of rank r. Then there are column vectors $a_1, \ldots, a_r \in \mathbb{R}^m$ and row vectors $b_1, \ldots, b_r \in \mathbb{R}^n$ with $M = \sum_{i\in[r]} a_i b_i$. Moreover,*

$\|a_i\|_\infty \leq 1$ *and* $\|b_i\|_\infty \leq \|M\|_\infty$ *for all* $1 \leq i \leq r$.

Proof. Consider the r dimensional vector space V spanned by all the rows $M_1, \ldots, M_m$ of M and identify the one dimensional exterior product $\bigwedge^r V$ with $\mathbb{R}$. Now choose r rows $M_{i_1}, \ldots, M_{i_r}$ for which $M_{i_1} \wedge \cdots \wedge M_{i_r}$ is the largest in absolute value in $\mathbb{R}$. As the M_i together span V it follows that the largest value is non-zero. Hence $M_{i_1}, \ldots, M_{i_r}$ form a basis of V. Therefore any row M_k can be uniquely written as a linear combination of the basis elements:

$$M_k = \sum_{j \in [r]} a_{k,j} M_{i_j}. \tag{2.16}$$

Fixing $j \in [r]$ and taking exterior products with the M_{i_l} where $l \neq j$ and both side we obtain

$$M_{i_1} \wedge \ldots \wedge M_{i_{j-1}} \wedge M_k \wedge M_{i_{j+1}} \wedge \ldots \wedge M_{i_r} = a_{k,j} \cdot M_{i_1} \wedge \ldots \wedge M_{i_r},$$

using the vanishing property of the exterior product. By maximality of $M_{i_1} \wedge \ldots \wedge M_{i_r}$, it follows that $|a_{k,j}| \leq 1$. We choose $a_k := \begin{bmatrix} a_{k,1} \\ \cdots \\ a_{k,r} \end{bmatrix}$, and thus we have $\|a_k\|_\infty \leq 1$. Moreover choose $b_j := M_{i_j}$, so that $\|b_j\|_\infty \leq \|M\|_\infty$ holds. Finally, Eq. (2.16) can be rewritten to $M = \sum_{j \in [r]} a_j b_j$, finishing the proof. □

For a vector a we can decompose it into its positive and negative part so that $a = a^+ - a^-$ with $a^+ a^- = 0$. Let $|a|$ denote the vector obtained from a by replacing every entry with its absolute value. Note that a^+, a^-, and $|a|$ are nonnegative vectors and $|a| = a^+ + a^-$. Furthermore their ℓ_∞-norm is at most $\|a\|_\infty$.

Theorem 2.6.2. *Let $\mathcal{P}$ be an optimization problem with (C, S)-approximate slack matrix*

M and let $\widetilde{M}$ be a nonnegative matrix. Then for the adjusted guarantee C' for $\mathcal{P}$ defined as

$$C'(f) \coloneqq C(f) + (\operatorname{rank} M + \operatorname{rank} \widetilde{M})\|\widetilde{M} - M\|_\infty \quad \textit{if } \mathcal{P} \textit{ is a maximization problem, and} \tag{2.17}$$

$$C'(f) \coloneqq C(f) - (\operatorname{rank} M + \operatorname{rank} \widetilde{M})\|\widetilde{M} - M\|_\infty \quad \textit{if } \mathcal{P} \textit{ is a minimization problem,} \tag{2.18}$$

we have

$$\mathrm{fc}_+(\mathcal{P}, C', S) \leq \operatorname{rank}_{\mathrm{LP}} \widetilde{M} + 2(\operatorname{rank} M + \operatorname{rank} \widetilde{M}) \qquad \textit{and} \tag{2.19}$$

$$\mathrm{fc}_\oplus(\mathcal{P}, C', S) \leq \operatorname{rank}_{\mathrm{SDP}} \widetilde{M} + 2(\operatorname{rank} M + \operatorname{rank} \widetilde{M}). \tag{2.20}$$

Proof. We prove the statement for maximization problems; the minimization case follows similarly. The proof is based on the vector identity

$$\sum_{i\in[k]} |a_i| b - \sum_{i\in[k]} a_i b_i = \sum_{i\in[k]} a_i^+ (b - b_i) + \sum_{i\in[k]} a_i^- (b + b_i). \tag{2.21}$$

In our setting, the a_i, b_i with $i \in [k]$ will arise from the (not necessarily nonnegative) factorization of $\widetilde{M} - M$, obtained by applying Lemma 2.6.1, i.e., we have

$$\widetilde{M} - M = \sum_{i\in[k]} a_i b_i, \tag{2.22}$$

where $\|a_i\|_\infty \leq 1$ and $\|b_i\|_\infty \leq \|M\|_\infty$ for $i \in [k]$ with $k \leq \operatorname{rank}(\widetilde{M} - M) \leq \operatorname{rank} M + \operatorname{rank} \widetilde{M}$. Furthermore, define $b \coloneqq \|\widetilde{M} - M\|_\infty \mathbb{1}$ to be the row vector with all entries equal to $\|\widetilde{M} - M\|_\infty \mathbb{1}$.

Substituting these values into (2.21), using Eq. (2.22) we obtain after rearranging

$$N \coloneqq \sum_{i\in[k]} |a_i| b + M = \widetilde{M} + \sum_{i\in[k]} a_i^+ (b - b_i) + \sum_{i\in[k]} a_i^- (b + b_i), \tag{2.23}$$

so that we can conclude

$$\operatorname{rank}_{\mathrm{LP}} N \leq \operatorname{rank}_{\mathrm{LP}} \widetilde{M} + 2k \leq \operatorname{rank}_{\mathrm{LP}} \widetilde{M} + 2(\operatorname{rank} M + \operatorname{rank} \widetilde{M}), \tag{2.24}$$

and similarly

$$\operatorname{rank}_{\mathrm{SDP}} N \leq \operatorname{rank}_{\mathrm{SDP}} \widetilde{M} + 2k \leq \operatorname{rank}_{\mathrm{SDP}} \widetilde{M} + 2(\operatorname{rank} M + \operatorname{rank} \widetilde{M}). \tag{2.25}$$

It remains to relate N to the (C', S)-approximate slack matrix of $\mathcal{P}$. By definition, the entries of N are

$$N(f,s) = \underbrace{\sum_{i\in[k]} |a_i(f)| \cdot \|\widetilde{M} - M\|_\infty + C(f)}_{:= f^* \leq C'(f)} - \operatorname{val}_f(s),$$

where $a_i(f)$ is the f-entry of a_i. Furthermore, as $\|a_i\|_\infty \leq 1$ and $k \leq \operatorname{rank} M + \operatorname{rank} \widetilde{M}$, we have $f^* \leq C'(f)$. Thus the (C', S)-approximate slack matrix M' of $\mathcal{P}$ looks like

$$M'(f,s) = N(f,s) + (C'(f) - f^*),$$

and as $f^* \leq C'(f)$, we have $\operatorname{rank}_{\mathrm{LP}} M' \leq \operatorname{rank}_{\mathrm{LP}} N$ and $\operatorname{rank}_{\mathrm{SDP}} M' \leq \operatorname{rank}_{\mathrm{SDP}} N$, establishing the claimed complexity bounds due to (2.24) in the LP case and (2.25) in the SDP case. □

A possible application of Theorem 2.6.2 is to 'thin-out' a given factorization of a slack matrix to obtain an approximation with low nonnegative rank. The idea is that if a nonnegative matrix factorization contains a large number of factors that contribute only very little to each of the entries, then we can simply drop those factors, significantly reduce the nonnegative rank, and obtain a very good approximation of the original optimization problem. Theorem 2.6.2 is then used to turn the approximation of the matrix into an approximation of the original problem of interest.

Also, it is possible to obtain low rank approximations of combinatorial problems sampling rank-1 factors proportional to their ℓ_1-weight as done in the context of information-theoretic approximations in Braun et al. (2014b). However, the obtained approximations tend to be too weak to be of interest.

Chapter 3

THE MATCHING PROBLEM HAS NO SMALL SYMMETRIC SDP

Continuing the work from the previous chapter the main result in this part is that any symmetric semidefinite program for the matching problem has exponential size. Recall that size in this context is the dimension of the semidefinite cone of the program. The result we show is the semidefinite version of a result by Yannakakis from his seminal work Yannakakis (1988) and Yannakakis (1991) that every symmetric linear program for the matching problem has exponential size. Rothvoß recently showed that the symmetry requirement in the linear case is not necessary, i.e., any linear program for the matching problem has exponential size (see Rothvoß 2014).

The matching problem is of particular interest, since unlike all other problems considered in the previous chapter, there is a polynomial time algorithm solving the maximum matching matching problem. This is also the reason that we do not expect to find an easy reduction from one of the other problems, for which SDP hardness results are known so far, since that reduction would have to be constructed in a way that cannot be adopted to a computational complexity setting. Instead we use the sum of squares proof technique, in particular we show that if there is a small symmetric SDP formulation, then there is a low-degree sum of squares refutation of the existence of a perfect matching in an odd clique, contradicting a result in Grigoriev (2001).

Related work

Yannakakis's statement that all symmetric linear programs for the matching problem have exponential size raised the question how important the symmetry requirement is. Restricting the number of edges to $\log n$ in K_n leads to a separation between polynomial and superpolynomial size linear formulations between the symmetric and the general version as shown by Kaibel, Pashkovich, and Theis (2010). For the permutahedron Goemans (2015) and Pashkovich (2014) established a similar result that dropping the symmetry restriction can mean the difference between subquadratic and quadratic size linear formulations. For TSP on the other hand Fiorini et al. (2012) and Fiorini et al. (2015) showed a lower bound of $2^{\Omega(\sqrt{n})}$ for any linear formulation and similarly Rothvoß (2014) showed that any linear formulation of the matching problem has size at least $2^{\Omega(n)}$, so in these two cases both the symmetric and the general formulations have exponential size.

A successful strategy for proving lower bounds on the size of relaxations besides the reduction mechanism from Chapter 2 is to use hierarchies, which are systematic ways of constructing relaxations of a given program in an iterative fashion. The size and approximation guarantee of a general relaxation is connected to a certain level of a hierarchy and then lower bounds on the size of the hierarchy are used to show the general lower bounds. In the case of CSPs Chan et al. (2013) used this strategy, which was improved by Kothari, Meka, and Raghavendra (2016), to show that the d-round Sherali-Adams linear relaxation hierarchy (see Sherali and Adams 1990) yields an approximation that is at least as good as any linear relaxation of size $n^{d/2}$. We used some lower bounds resulting in this way as base hardness results in Chapter 2. In the semidefinite case for CSPs Lee et al. (2014) showed that the d-round Lasserre SDP relaxation hierarchy yields at least as good an approximation as any symmetric SDP relaxation of size $n^{d/2}$, leading in particular to an SDP relaxation lower bound for Max-3-SAT.

Contribution

The main contribution of this chapter is the lower bound on the size of symmetric semidefinite formulations for the matching problem. We summarize the contribution as follows:

(i) **The matching problem admits no small approximate SDP formulation.** The first result we show in this chapter is as follows: There exists an absolute constant $\alpha > 0$ such that for every $0 \leq \varepsilon < 1$, every symmetric SDP formulation approximating the perfect matching problem within a factor $1 - \frac{\varepsilon}{n-1}$ has size at least $2^{\alpha n}$ (see Theorem 3.2.10).

(ii) **The Lasserre relaxation is optimal among symmetric SDP relaxations for the asymmetric metric traveling salesperson problem.** The asymmetric metric TSP is the restriction of TSP where the costs of an edge u to v can be different from the costs of the edge v to u and additionally the edge costs obey triangle inequality. The second result proven in this chapter is a connection between a general symmetric SDP relaxation and the Lasserre hierarchy for the asymmetric metric TSP. In particular, we show that for every $\rho > 0$, if there exists a symmetric SDP relaxation of size $r < \sqrt{\binom{2n}{k}} - 1$ that achieves a ρ-approximation for asymmetric metric TSP instances on $2n$ vertices, then the $(2k-1)$-round Lasserre relaxation achieves a ρ-approximation for asymmetric metric TSP instances on n vertices.

3.1 Symmetric formulations as sum of squares

We first introduce some notation that we will use specifically in this chapter. Let $[n]$ denote the set $\{1, \ldots, n\}$. Let $\mathbb{R}[x]$ denote the set of polynomials in n real variables $x = (x_1, \ldots, x_n)$ with real coefficients. For a set $\mathcal{H} \subseteq \mathbb{R}[x]$ let $\langle \mathcal{H} \rangle$ denote the vector space spanned by $\mathcal{H}$ and let $\langle \mathcal{H} \rangle_I$ denote the ideal $\mathbb{R}[x]$ generated by $\mathcal{H}$. Recall that a polynomial *ideal* in a polynomial ring R is a set that is closed under addition of polynomials in the ideal and closed

under multiplication by polynomials in the ring.

We now turn a G-coordinate-symmetric SDP formulation into a symmetric sum of squares representation over a small set of basis functions.

Lemma 3.1.1 (Sum of squares for a symmetric SDP formulation)**.** *If a G-symmetric maximization problem $\mathcal{P} = (\mathcal{S}, \mathcal{F})$ admits a G-coordinate-symmetric (C, S)-approximate SDP formulation of size d, then there is a set $\mathcal{H}$ of at most $\binom{d+1}{2}$ functions $h\colon \mathcal{S} \to \mathbb{R}$ such that for any $f \in \mathcal{F}$ with $\max f \leq S(f)$ we have $C(f) - f = \sum_j h_j^2 + \mu_f$ for some $h_j \in \langle \mathcal{H} \rangle$ and constant $\mu_f \geq 0$. Furthermore the set $\mathcal{H}$ is invariant under the action of G given by $(g \cdot h)(s) = h(g^{-1} \cdot s)$ for $g \in G$, $h \in H$ and $s \in S$.*

Proof. For any psd matrix M let $\sqrt{M}$ denote the unique psd matrix with $\sqrt{M}^2 = M$. Note that $\sqrt{M}\sqrt{M}^\mathsf{T} = M$ also, since $\sqrt{M}$ is symmetric.

Let $\mathcal{A}, b, \{X^s \mid s \in \mathcal{S}\}, \{w^f \mid f \in \mathcal{F}\}$ comprise a G-coordinate-symmetric SDP formulation of size d. We define the set $\mathcal{H} \coloneqq \{h_{ij} \mid i, j \in [d]\}$ via $h_{ij}(s) \coloneqq \sqrt{X^s}_{ij}$. By the action of G and the uniqueness of the square root, we have $g \cdot h_{ij} = h_{g \cdot i, g \cdot j}$, so $\mathcal{H}$ is G-symmetric. As $h_{ij} = h_{ji}$, the set $\mathcal{H}$ has at most $\binom{d+1}{2}$ elements.

By standard strong duality arguments as for example used in the proof of Theorem 2.1.5, for every $f \in \mathcal{F}$ with $\max f \leq S(f)$, there is a $U^f \in \mathbb{S}^d_+$ and $\mu_f \geq 0$ such that for all $s \in \mathcal{S}$,

$$C(f) - f(s) = \operatorname{Tr}[U^f X^s] + \mu_f.$$

Again by standard arguments the trace can be rewritten as a sum of squares:

$$\operatorname{Tr}[U^f X^s] = \operatorname{Tr}\left[\left(\sqrt{U^f}\sqrt{X^s}\right)^\mathsf{T}\left(\sqrt{U^f}\sqrt{X^s}\right)\right] = \sum_{i,j \in [d]} \left(\sum_{k \in [d]} \sqrt{U^f}_{ik} \cdot \sqrt{X^s}_{kj}\right)^2.$$

Therefore $C(f) - f = \sum_{i,j \in [d]} \left(\sum_{k \in [d]} \sqrt{U^f}_{ik} \cdot h_{kj}\right)^2 + \mu_f$, as claimed. □

3.2 The perfect matching problem

We recall the definition of the *perfect matching problem* $\mathrm{PM}(n)$ from Definition 2.1.9 where we established a lower bound on the linear formulation complexity and show that any symmetric SDP formulation has exponential size.

Let n be an even positive integer, and let K_n denote the complete graph on n vertices. The feasible solutions of $\mathrm{PM}(n)$ are all the perfect matchings M on K_n. The instances are indexed by the edge sets F of K_n and the objective function computes $\mathrm{val}_F(M) = |M \cap F|$ the number of edges contained in both the edge set F and the perfect matching M. For approximation guarantees we use $S(f) \coloneqq \max f$ and $C(f) \coloneqq \max f + \varepsilon/2$ for some fixed $0 \leq \varepsilon < 1$ as before and in Braun and Pokutta (2015a); see also Braun and Pokutta (2015b) for a more in-depth discussion.

3.2.1 Symmetric functions on matchings are juntas

In this section we show that functions on perfect matchings with high symmetry are actually *juntas*: they depend only on the edges of a small vertex set. The key is the following lemma stating that perfect matchings coinciding on a vertex set belong to the same orbit of the pointwise stabilizer of the vertex set. Let A_n denote the alternating group on n letters, and for any subset $X \subseteq [n]$ let $A(X)$ denote the alternating group that operates on the elements of X and fixes the remaining elements of $[n]$. For any set $W \subseteq [n]$ let $E[W]$ denote the edges of K_n with both endpoints in W.

Lemma 3.2.1. *Let $S \subseteq [n]$ with $|S| < n/2$ and let M_1 and M_2 be perfect matchings in K_n. If $M_1 \cap E[S] = M_2 \cap E[S]$ then there exists $\sigma \in A([n] \setminus S)$ such that $\sigma \cdot M_1 = M_2$.*

Proof. Let $\delta(S)$ denote the edges with exactly one endpoint in S. There are three kinds of edges: those in $E[S]$, those in $\delta(S)$, and those disjoint from S. We construct σ to handle each type of edge, then fix σ to be even.

To handle the edges in $E[S]$ we set σ to the identity on S, since $M_1 \cap E[S] = M_2 \cap E[S]$.

To handle the edges in $\delta(S)$ we note that for each edge $(s,v) \in M_1$ with $s \in S$ and $v \notin S$ there is a unique edge $(s,w) \in M_2$ with $w \notin S$. We extend σ to map v to w for each such s.

To handle the edges disjoint from S, we again use the fact that M_1 and M_2 are perfect matchings, so the number of edges in each that are disjoint from S is the same. We extend σ to be an arbitrary bijection on those edges.

We now show that we can choose σ to be even. Since $|S| < n/2$ there is an edge $(u,v) \in M_2$ disjoint from S. Let $\tau_{u,v}$ denote the transposition of u and v and let $\sigma' := \tau_{u,v} \circ \sigma$. We have $\sigma' \cdot M_1 = \sigma \cdot M_1 = M_2$, and either σ or σ' is even. □

We also need the following lemma, which has been used extensively for symmetric linear extended formulations. See references Yannakakis (1988), Yannakakis (1991), Kaibel, Pashkovich, and Theis (2010), Braun and Pokutta (2011), and Lee et al. (2014) for examples.

Lemma 3.2.2 (Dixon and Mortimer 1996, Theorems 5.2A and 5.2B)**.** *Let $n \geq 10$ and let $G \leq A_n$ be a group. If $|A_n : G| < \binom{n}{k}$ for some $k < n/2$, then there is a subset $W \subseteq [n]$ such that $|W| < k$, W is G-invariant, and $A([n] \setminus W)$ is a subgroup of G.*

We now formally state and prove the claim about juntas:

Proposition 3.2.3. *Let $n \geq 10$, let $k < n/2$ and let $\mathcal{H}$ be an A_n-symmetric set of functions on the set of perfect matchings of K_n of size less than $\binom{n}{k}$. Then for every $h \in \mathcal{H}$ there is a vertex set $W \subseteq [n]$ of size less than k such that h depends only on the (at most $\binom{k-1}{2}$) edges in W.*

Proof. Let $h \in \mathcal{H}$, let $\mathrm{Stab}(h)$ denote the stabilizer of h, and let $\mathrm{Orb}(h)$ denote the orbit of h. Since $\mathcal{H}$ is A_n-symmetric we have $|\mathrm{Orb}(h)| < \binom{n}{k}$. By the orbit-stabilizer theorem it follows that $|A_n : \mathrm{Stab}(h)| < \binom{n}{k}$. Applying Lemma 3.2.2 to the stabilizer of h, we obtain

a subset $W \subseteq [n]$ of size less than k such that h is stabilized by $A([n] \setminus W)$, i.e.,

$$h(M) = (g \cdot h)(M) = h(g^{-1} \cdot M)$$

for all $g \in A([n] \setminus W)$.

Therefore for every perfect matching M the function h is constant on the $A([n] \setminus W)$-orbit of M. As the orbit is determined by $M \cap E[W]$ by Lemma 3.2.1, so is the function value $h(M)$. Therefore h depends only on the edges in $E[W]$. □

3.2.2 The matching polynomials

A key step in proving our lower bound is obtaining low-degree derivations of approximation guarantees for objective functions of $\mathrm{PM}(n)$. Therefore we start with a standard representation of functions as polynomials. We define the *matching constraint polynomials* $\mathcal{P}_n$ as:

$$\begin{aligned} \mathcal{P}_n := &\{x_{uv}x_{uw} \mid u,v,w \in [n] \text{ distinct}\} \\ &\cup \left\{ \sum_{u \in [n], u \neq v} x_{uv} - 1 \;\middle|\; v \in [n] \right\} \\ &\cup \left\{ x_{uv}^2 - x_{uv} \mid u,v \in [n] \text{ distinct} \right\}. \end{aligned} \tag{3.1}$$

We observe that the ring of real valued functions on perfect matchings is isomorphic to $\mathbb{R}[\{x_{uv}\}_{\{u,v\} \in \binom{[n]}{2}}] / \langle \mathcal{P}_n \rangle_I$ with x_{uv} representing the indicator function of the edge uv being contained in a perfect matching. Intuitively, under this representation the vanishing of the first set of polynomials ensures that no vertex is matched more than once, the vanishing of the second set ensures that each vertex is matched, and the vanishing of the third set ensures that each edge coordinate is 0-1 valued.

Now we formulate low-degree derivations. Let $\mathcal{P}$ denote a set of polynomials in $\mathbb{R}[x]$. For polynomials F and G, we write $F \simeq_{(\mathcal{P},d)} G$, or F *is congruent to* G *from* $\mathcal{P}$ *in degree* d,

if and only if there exist polynomials $\{q(p) : p \in \mathcal{P}\}$ such that

$$F + \sum_{p \in \mathcal{P}} q(p) \cdot p = G$$

and $\max_p \deg(q(p) \cdot p) \leq d$. We often drop the dependence on $\mathcal{P}$ when it is clear from context. We shall write $F \equiv G$ for two polynomials F and G defining the same function on perfect matchings, i.e., $F - G \in \langle \mathcal{P}_n \rangle_I$. (Note that as $\mathcal{P}_n$ contains $x_{uv}^2 - x_{uv}$ for all variables x_{uv}, the ideal generated by $\mathcal{P}_n$ is automatically radical.)

3.2.3 Deriving that symmetrized polynomials are constant

Averaging any polynomial on matchings over the symmetric group gives a constant. In this section we show that this fact has a low-degree derivation.

For a partial matching M, let $x_M \coloneqq \prod_{e \in M} x_e$ denote the product of edge variables for the edges in M. The first step is to reduce every polynomial to a linear combination of the x_M.

Lemma 3.2.4. *For every polynomial F there is a polynomial F' with $\deg F' \leq \deg F$ and $F \simeq_{(\mathcal{P}_n, \deg F)} F'$, where all monomials of F' have the form x_M for some partial matching M.*

Proof. It suffices to prove the lemma when F is a monomial. Let $F = \prod_{e \in A} x_e^{k_e}$ for a set A of edges with multiplicities $k_e \geq 1$. From $x_e^2 \simeq_2 x_e$ it follows that $x_e^k \simeq_k x_e$ for all $k \geq 1$, hence $F \simeq_{\deg F} \prod_{e \in A} x_e$. If A is a partial matching we are done, otherwise there are distinct $e, f \in A$ with a common vertex, hence $x_e x_f \simeq_2 0$ and $F \simeq_{\deg F} 0$. □

Lemma 3.2.5. *For any partial matching M on $2d$ vertices and a vertex a not covered by M, we have*

$$x_M \simeq_{(\mathcal{P}_n, d+1)} \sum_{\substack{M_1 = M \cup \{a,u\} \\ u \in K_n \setminus (M \cup \{a\})}} x_{M_1}. \tag{3.2}$$

Proof. We use the generators $\sum_u x_{au} - 1$ to add variables corresponding to edges at a, and

then use $x_{au}x_{uv}$ to remove monomials not corresponding to a partial matching:

$$x_M \simeq_{(\mathcal{P}_n,d+1)} x_M \sum_{u\in K_n} x_{au} \simeq_{(\mathcal{P}_n,d+1)} \sum_{\substack{M_1=M\cup\{a,u\}\\ u\in K_n\setminus(M\cup\{a\})}} x_{M_1}.$$

□

This leads to a similar congruence using all containing matchings of a larger size:

Lemma 3.2.6. *For any partial matching M of $2d$ vertices and $d \le k \le n/2$, we have*

$$x_M \simeq_{(\mathcal{P}_n,k)} \frac{1}{\binom{n/2-d}{k-d}} \sum_{\substack{M'\supset M\\ |M'|=k}} x_{M'} \tag{3.3}$$

Proof. We use induction on $k-d$. The start of the induction is with $k=d$, when the sides of (3.3) are actually equal. If $k > d$, let a be a fixed vertex not covered by M. Applying Lemma 3.2.5 to M and a followed by the inductive hypothesis gives:

$$x_M \simeq_{(\mathcal{P}_n,d+1)} \sum_{\substack{M_1=M\cup\{a,u\}\\ u\in K_n\setminus(M\cup\{a\})}} x_{M_1} \simeq_{(\mathcal{P}_n,k)} \frac{1}{\binom{n/2-d-1}{k-d-1}} \sum_{\substack{M'\supset M_1\\ |M'|=k\\ M_1=M\cup\{a,u\}\\ u\in K_n\setminus(M\cup\{a\})}} x_{M'}.$$

Averaging over all vertices a not covered by M, we obtain:

$$\begin{aligned} x_M &\simeq_{(\mathcal{P}_n,k)} \frac{1}{n-2d}\frac{1}{\binom{n/2-d-1}{k-d-1}} \sum_{\substack{M'\supset M_1\\ |M'|=k\\ M_1=M\cup\{a,u\}\\ a,u\in K_n\setminus M}} x_{M'} \\ &= \frac{1}{n-2d}\frac{1}{\binom{n/2-d-1}{k-d-1}} 2(k-d) \sum_{\substack{M'\supset M\\ |M'|=k}} x_{M'} \\ &= \frac{1}{\binom{n/2-d}{k-d}} \sum_{\substack{M'\supset M\\ |M'|=k}} x_{M'}. \end{aligned}$$

where in the second step the factor $2(k-d)$ accounts for the number of ways to choose a and u. □

We are now ready to state and prove the claim about symmetrized polynomials:

Lemma 3.2.7. *For any polynomial F, there is a constant c_F with $\sum_{\sigma\in S_n}\sigma F \simeq_{(\mathcal{P}_n,\deg F)} c_F$.*

Proof. Given Lemma 3.2.4, it suffices to prove the claim for $F = x_M$ for some partial matching M. Note that if $|M| = k$ then (using the orbit-stabilizer theorem) the size of the stabilizer of M is $2^k k!(n-2k)!$. Now apply Lemma 3.2.6 with $d = 0$:

$$\sum_{\sigma\in S_n}\sigma x_M = 2^k k!(n-2k)! \sum_{M' :\, |M'|=k} x_{M'} \simeq_k 2^k k!(n-2k)!\binom{n/2}{k}.$$

□

3.2.4 Low-degree certificates for matching ideal membership

In this section we present a crucial part of our argument, namely that every degree-d polynomial that is identically zero over perfect matchings has a derivation of this fact whose degree is $O(d)$.

The following lemma will allow us to apply induction:

Lemma 3.2.8. *If L is a polynomial with $L \simeq_{(\mathcal{P}_{n-2},d)} 0$ for some d, and a,b are the two additional vertices in K_n, then $Lx_{ab} \simeq_{(\mathcal{P}_n,d+1)} 0$.*

Proof. It is enough to prove the claim for $L \in \mathcal{P}_{n-2}$. For $L = x_e^2 - x_e$ and $L = x_{uv}x_{uw}$ the claim is trivial since $L \in \mathcal{P}_n$ also. The remaining case is $L = \sum_{u\in K_{n-2}} x_{uv} - 1$ for some $v \in K_{n-2}$. Then

$$Lx_{ab} = \left(\sum_{u\in K_{n-2}} x_{uv} - 1\right) x_{ab} = \left(\sum_{u\in K_n} x_{uv} - 1\right) x_{ab} - x_{av}x_{ab} - x_{bv}x_{ab} \simeq_{d+1} 0.$$

The degree of the derivation is at most $d+1$ since we can simply multiply the degree-d derivation for $L \simeq 0$ by x_{ab}. □

We now show that any $F \in \langle \mathcal{P}_n \rangle_I$ can be generated by low-degree coefficients from $\mathcal{P}_n$:

Theorem 3.2.9. *For every $F \in \mathbb{R}[\{x_{uv}\}_{\{u,v\} \in \binom{n}{2}}]$, if $F \in \langle \mathcal{P}_n \rangle_I$ then $F \simeq_{(\mathcal{P}_n, 2\deg F - 1)} 0$.*

Proof. We use induction on the degree d of F. If $d = 0$ then $F = 0$ and the statement holds trivially. (Note that $\simeq_{-1}$ is just equality.) The case $d = 1$ rephrased means that the affine space spanned by the characteristic vectors of all perfect matchings is defined by the $\sum_v x_{uv} - 1$ for all vertices u. This follows from Edmonds's description of the perfect matching polytope by linear inequalities in Edmonds (1965).

For the case $d \geq 2$ we first prove the following claim:

Claim. If $F \in \langle \mathcal{P}_n \rangle_I$ is a degree-d polynomial and $\sigma \in S_n$ is a permutation of vertices, then

$$F \simeq_{(\mathcal{P}_n, 2d-1)} \sigma F.$$

We use induction on the degree. If $d = 0$ or $d = 1$ the claim follows from the corresponding cases $d = 0$ and $d = 1$ of the theorem. For $d \geq 2$ it is enough to prove the claim when σ is a transposition of two vertices a and u. Note that in $F - \sigma F$ all monomials which are independent of both a and u cancel:

$$F - \sigma F = \sum_{e\colon\, a \in e \text{ or } u \in e} L_e x_e \tag{3.4}$$

where each L_e has degree at most $d-1$. We now show that every summand is congruent to a sum of monomials containing edges incident to both a and u. For example, for $e = \{a, b\}$ in (3.4) we apply the generator $\sum_v x_{uv} - 1$ to find:

$$L_{ab} x_{ab} \simeq_{d+1} L_{ab} x_{ab} \sum_v x_{uv} \simeq_{d+1} \sum_v L_{ab} x_{ab} x_{uv}.$$

Therefore

$$F - \sigma F \simeq_{d+1} \sum_{bv} L'_{bv} x_{ab} x_{uv}$$

for some polynomials L'_{bv} of degree at most $d-1$. We may assume that L'_{bv} does not contain variables x_e with e incident to a, b, u, v, as these can be removed using generators like $x_{ab}x_{ac}$ or $x_{ab}^2 - x_{ab}$. Moreover, it can be checked that L'_{bv} is zero on all perfect matchings containing $\{a, b\}$ and $\{u, v\}$. By induction, $L'_{bv} \simeq_{(\mathcal{P}_{n-4}, 2d-3)} 0$ (identifying K_{n-4} with the graph $K_n \setminus \{a, b, u, v\}$), from which $L'_{bv} \simeq_{(\mathcal{P}_n, 2d-1)} 0$ follows by two applications of Lemma 3.2.8. (The special case $a = v, b = u$ is also handled by induction and one application of Lemma 3.2.8.) This concludes the proof of the claim.

We now apply the claim followed by Lemma 3.2.7:

$$F \simeq_{2d-1} \frac{1}{n!} \sum_{\sigma \in S_n} \sigma F \simeq_d \frac{c_F}{n!}$$

for a constant c_F. As $F \in \langle \mathcal{P}_n \rangle_I$, it must be that $c_F = 0$, and therefore $F \simeq_{2d-1} 0$. □

3.2.5 The lower bound on the size of symmetric SDP extensions

We now have all the ingredients to prove the main theorem of this chapter. Note that the alternating group A_n acts naturally on $\mathrm{PM}(n)$ via permutation of vertices. Recall that we set $S(f) := \max f$ and $C(f) := \max f + \varepsilon/2$, where the functions f are indexed by edge sets and ε is a parameter. It follows that the guarantees C and S are A_n-symmetric in the sense defined in Section 1.1.3. Our main theorem is an exponential lower bound on the size of any A_n-coordinate-symmetric SDP extension of $\mathrm{PM}(n)$.

Theorem 3.2.10. *There exists an absolute constant $\alpha > 0$ such that for all even n and every $0 \leq \varepsilon < 1$, every A_n-coordinate-symmetric SDP extended formulation approximating the perfect matching problem $\mathrm{PM}(n)$ within a factor of $1 - \varepsilon/(n-1)$ has size at least $2^{\alpha n}$.*

Proof. Fix an integer $n \geq 10$ and let $k = \lceil \beta n \rceil$ for some small enough constant $0 < \beta <$

$1/2$ chosen later. Suppose for a contradiction that $\mathrm{PM}(n)$ admits a symmetric SDP extended formulation of size $d < \sqrt{\binom{n}{k}} - 1$.

Let m equal $n/2$ or $n/2-1$, whichever is odd. Let $S = [m]$ and let $T = \{m+1, \ldots, 2m\}$. If $m = n/2$ then let $U = \{2m+1, 2m+2\}$, otherwise let $U = \varnothing$. Note that $S \cup T \cup U = [n]$ and $|S| = |T| = m = \Theta(n)$. Consider the objective function for the set of edges $E[S]$, namely $f_{E[S]}(M) \coloneqq |M \cap E[S]|$. Since $|S|$ is odd we have $\max f_{E[S]} = (|S|-1)/2$, from which we obtain:

$$f(x) \coloneqq C(f_{E[S]}) - f_{E[S]}(x) = \frac{|S|-1}{2} + \frac{\varepsilon}{2} - \sum_{u,v \in S} x_{uv} \equiv \frac{1}{2} \sum_{u \in S, v \in T \cup U} x_{uv} - \frac{1-\varepsilon}{2}. \tag{3.5}$$

By Lemma 3.1.1, as $\binom{d+1}{2} < \binom{n}{k}$, there is a constant $\mu_f \geq 0$ and an A_n-symmetric set $\mathcal{H}$ of functions of size at most $\binom{n}{k}$ on the set of perfect matchings with

$$f \equiv \sum_g g^2 + \mu_f \qquad \text{with each } g \in \langle \mathcal{H} \rangle.$$

By Proposition 3.2.3, every $h \in \mathcal{H}$ depends only on the edges within a vertex set of size less than k, and hence can be represented by a polynomial of degree less than $k/2$ over perfect matchings. As the g are linear combinations of the $h \in \mathcal{H}$, they can also be represented by polynomials of degree less than $k/2$, which we assume for the rest of the proof.

Applying Theorem 3.2.9 with (3.5), we conclude

$$\frac{1}{2} \sum_{u \in S, v \in T \cup U} x_{uv} - \frac{1-\varepsilon}{2} \simeq_{(\mathcal{P}_n, 2k-1)} \sum_g g^2 + \mu_f.$$

We now apply the following substitution: set $x_{2m+1,2m+2} \coloneqq 1$ if U is not empty, set $x_{u+m,v+m} \coloneqq x_{uv}$ for each $uv \in E[S]$, and set $x_{uv} \coloneqq 0$ otherwise. Intuitively, the substitution ensures that U is matched, ensures the matching on T is identical to the matching on S, and ensures every edge is entirely within S, T, or U. The main point is that the substitution maps every polynomial in $\mathcal{P}_n$ either to 0 or into $\mathcal{P}_m$.

Applying this substitution we obtain a new polynomial identity on the variables $\{x_{uv}\}_{\{u,v\}\in\binom{S}{2}}$:

$$-\frac{1-\varepsilon}{2} \simeq_{(\mathcal{P}_m,2k-1)} \sum_g g^2 + \mu_f. \tag{3.6}$$

This equation is a sum of squares proof that an odd clique of size m cannot have a perfect matching. To complete our argument we appeal to a theorem from Grigoriev (2001) which shows that any such proof must have high degree. Since the degree of the proof in (3.6) is $2k-1$, our conclusion will be that k must be large.

The theorem from Grigoriev (2001) uses different terminology from what we have developed here. It is phrased in terms of Positivstellensatz Calculus ($PC_>$) proof systems and the MOD_2 principle. We first present the theorem as originally stated and then relate it to our setting.

Theorem 3.2.11 (Grigoriev 2001, Corollary 2)**.** *The degree of any $PC_>$ refutation of MOD_2^k is greater than $\Omega(k)$.*

The MOD_p^k principle states that it is not possible to partition a set of size k into groups of size p if k is congruent to 1 modulo p. In our case, with $p=2$ and k odd, this is equivalent to the statement that no perfect matching exists in an odd clique.

Likewise, via Grigoriev (2001, Definition 2) one checks that (3.6) constitutes a $PC_>$ proof; we refer the reader to Buss et al. (1999) for further discussion.

Applying Theorem 3.2.11 to (3.6), we find that $2k-1=\Omega(m)=\Omega(n)$, a contradiction when β is chosen small enough, establishing inapproximability with guarantees $C(f_{E[S]}) = \max f_{E[S]} + \varepsilon/2$ and $S(f_{E[S]}) = \max f_{E[S]}$. The inapproximability factor follows as we have seen in Theorem 2.1.10 from

$$\max f_{E[S]} = \frac{m-1}{2} \leq \min\left\{\frac{\frac{n-1}{2}-1}{2}, \frac{\frac{n}{2}-1}{2}\right\} = \frac{n-3}{4}$$

and thus

$$\frac{S(f_{E[S]})}{C(f_{E[S]})} = \frac{\max f_{E[S]}}{\max f_{E[S]} + \varepsilon/2} = \frac{1}{1 + \frac{\varepsilon/2}{\max f_{E[S]}}} \leq \frac{1}{1 + \frac{\varepsilon/2}{(n-3)/4}} = \frac{1}{1 + \frac{2\varepsilon}{n-3}} \leq 1 - \frac{\varepsilon}{n-1},$$

which finishes the proof. □

3.3 The Metric Traveling Salesperson Problem (TSP) revisited

In this section, we prove that a particular Lasserre SDP is optimal among all symmetric SDP relaxations for the asymmetric metric traveling salesperson problem on K_n. The *feasible solutions* of the problem are all permutations $\sigma \in S_n$. A permutation σ corresponds to the tour in K_n in which vertex i is the $\sigma(i)$-th vertex visited. An *instance* $\mathcal{I}$ of TSP is a set of non-negative distances $d_{\mathcal{I}}(i,j)$ for each edge $(i,j) \in K_n$, obeying the triangle inequality. The value of a tour σ is just the sum of the distances of edges traversed $\mathrm{val}_{\mathcal{I}}(\sigma) = \sum_i d_{\mathcal{I}}(\sigma^{-1}(i), \sigma^{-1}(i+1))$. The *objective functions* are all the $\mathrm{val}_{\mathcal{I}}$. For approximation guarantees we will use $S(f) = \min f$ and $C(f) = \min f/\rho$ for some factor $\rho \geq 1$. As a reminder instead of referring to a "(C,S)-approximate formulation" we will refer to a "formulation within a factor ρ."

The natural action of A_n on TSP is by permutation of vertices, which means here that A_n acts on S_n by composition from the left: $(\sigma_1 \cdot \sigma_2)(i) = \sigma_1(\sigma_2(i))$. Obviously, the problem TSP is A_n-symmetric.

The ring of real-valued functions on the set S_n of feasible solutions is isomorphic to $\mathbb{R}[\{x_{ij}\}_{\{i,j\}\in[n]}]/\langle \mathcal{Q}_n \rangle_I$, with x_{ij} being the indicator of $\sigma(i) = j$, and $\mathcal{Q}_n$ is the set of *TSP*

constraints:

$$\mathcal{Q}_n = \left\{ \sum_{i\in[n]} x_{ij} - 1 \,\middle|\, j \in [n] \right\} \cup \left\{ \sum_{j\in[n]} x_{ij} - 1 \,\middle|\, i \in [n] \right\}$$
$$\cup \left\{ x_{ij}x_{ik} \,\middle|\, i,j,k \in [n] \right\} \cup \left\{ x_{ij}x_{kj} \,\middle|\, i,j,k \in [n] \right\}$$
$$\cup \left\{ x_{ij}^2 - x_{ij} \,\middle|\, i,j \in [n] \right\}.$$

We emphasize that our description of the TSP constraints is different from the TSP polytope treated in Yannakakis (1991), Yannakakis (1988), Fiorini et al. (2012), and Fiorini et al. (2015): the variables x_{ij} do not directly encode the edges of a Hamiltonian cycle but instead specify a permutation of n vertices, encoded as a perfect bipartite matching on $K_{n,n}$.

Following the framework presented in Lee et al. (2014), we define the Lasserre hierarchy for TSP as follows. The (dual of) the k-th level Lasserre SDP relaxation for a TSP instance $\mathcal{I}$ is given by

$$\begin{aligned} &\text{Maximize } c \\ &\text{subject to } \mathrm{val}_{\mathcal{I}} - c \simeq_{(\mathcal{Q}_n,k)} \sum_p p^2 \qquad \text{for some polynomials } p. \end{aligned}$$

We now state our main theorem regarding optimal SDP relaxations for TSP.

Theorem 3.3.1. *Suppose that there is some coordinate A_{2n}-symmetric SDP relaxation of size $r < \sqrt{\binom{n}{k}} - 1$ approximating TSP within some factor $\rho \geq 1$ for instances on $2n$ vertices. Then the $(2k-1)$-level Lasserre relaxation approximates TSP within the factor of ρ on instances on n vertices.*

To prove Theorem 3.3.1 there is an equivalent of Proposition 3.2.3 we need for TSP tours, so that a small set of invariant functions depends only on the positions of a small number of indices. We start with the following proposition.

Proposition 3.3.2. *Let $\mathcal{H}$ be an A_n-symmetric set of functions of size $\binom{n}{k}$ on the set of TSP*

tours $\sigma \in S_n$. Then for every $h \in \mathcal{H}$ there is a set $W \subseteq [n]$ of size less than k, such that $h(\sigma)$ depends only on the positions of the vertices in W in the tour σ, and the sign of σ as a permutation.

Proof. For every $h \in \mathcal{H}$ we can apply Lemma 3.2.2 to the stabilizer of h to obtain a subset $W \subseteq [n]$ of size at most k such that h is stabilized by $A([n] \setminus W)$. Thus for every tour σ, h is constant on the $A([n] \setminus W)$-orbit of σ. This orbit is clearly determined by the positions of the vertices in W and, since $A([n] \setminus W)$ preserves signs, the sign of the permutation σ. □

Next we give a reduction which allows us to eliminate the dependence of the functions $h \in \mathcal{H}$ on the sign of the permutation σ. In particular we encode every TSP tour σ on an n-vertex graph as some new tour $\Phi(\sigma)$ in a $2n$-vertex graph, such that $\Phi(\sigma)$ is always an even permutation in S_{2n}.

Lemma 3.3.3. *Let $\mathcal{I}$ be an instance of TSP on K_n. Then there exists an instance $\mathcal{I}'$ of TSP on K_{2n} and an injective map $\Phi : S_n \to S_{2n}$ such that*

(i) $\mathrm{val}_{\mathcal{I}}(\sigma) = \mathrm{val}_{\mathcal{I}'}(\Phi(\sigma))$ *for all* $\sigma \in S_n$.

(ii) For every tour $\tau \in S_{2n}$ *there exists* $\sigma \in S_n$ *such that* $\mathrm{val}_{\mathcal{I}'}(\Phi(\sigma)) \leq \mathrm{val}_{\mathcal{I}'}(\tau)$

(iii) For all $\sigma \in S_n$ *the permutation* $\Phi(\sigma)$ *is even.*

Proof. Given a TSP instance $\mathcal{I}$ on K_n we construct a new instance $\mathcal{I}'$ on K_{2n} as follows:

- For every vertex $i \in \mathcal{I}$ add a pair of vertices i and i' to $\mathcal{I}'$.
- For every distance $d(i,j)$ in $\mathcal{I}$ add 4 edges all with the same distance $d(i,j) = d(i',j) = d(i,j') = d(i',j')$ to $\mathcal{I}'$.
- For every pair of vertices $i, i' \in \mathcal{I}'$ add an edge of distance zero, i.e. set $d(i,i') = 0$.

We will call a tour $\tau \in S_{2n}$ *canonical* if it visits i' immediately after i, i.e. $\sigma(i') = \sigma(i) + 1$. We will write T for the set of canonical tours in S_{2n}. It is easy to check using the triangle

inequality that for every tour τ there is a canonical tour with no larger value. For every tour σ in $\mathcal{I}$ define $\Phi(\sigma)$ to be the corresponding canonical tour in $\mathcal{I}'$. That is $\Phi(\sigma)(i) = 2\sigma(i) - 1$ and $\Phi(\sigma)(i') = 2\sigma(i)$. Note that $\Phi : S_n \to S_{2n}$ is an injective map whose image is all of T. By construction we have:

$$\mathrm{val}_{\mathcal{I}}(\sigma) \equiv \mathrm{val}_{\mathcal{I}'}(\Phi(\sigma))$$

which proves property (1). Property (2) follows from the fact that every tour $\tau \in S_{2n}$ has a canonical tour with no larger value, and that T is the image of Φ.

For property (3), note that every canonical tour is an even permutation. To see why, suppose $\sigma \in S_n$ is given by $\sigma = (i_1, j_1)(i_2, j_2), \ldots, (i_m, j_m)$ where (i, j) denotes the permutation that swaps i and j. Then $\Phi(\sigma) = (i_1, j_1)(i'_1, j'_1), \ldots, (i_m, j_m)(i'_m, j'_m)$ is comprised of $2m$ swap permutations, and is therefore even. □

The last ingredient we need is a version of Theorem 3.2.9 for the TSP.

Theorem 3.3.4. *If F is a multilinear polynomial whose monomials are partial matchings on $K_{n,n}$ and $F \in \langle \mathcal{Q}_n \rangle_I$, then $F \simeq_{(\mathcal{Q}_n, 2\deg F - 1)} 0$.*

Because $\mathcal{Q}_n$ is so similar to $\mathcal{P}_n$, it should come as no surprise that the proof of the above theorem is extremely similar to the proof of Theorem 3.2.9. We include the full proof for completeness, but defer it to Section 3.3.1. We now have all the tools necessary to prove Theorem 3.3.1.

Proof of Theorem 3.3.1. First let $\mathcal{I}$ be an instance of TSP on K_n. Use Lemma 3.3.3 to construct a TSP instance $\mathcal{I}'$ on K_{2n} and the corresponding map Φ. Now assume we have an arbitrary A_{2n}-symmetric SDP relaxation of size $d < \sqrt{\binom{2n}{k}} - 1$ for TSP on K_{2n}. By Lemma 3.1.1 there is a corresponding A_{2n}-symmetric family of functions $\mathcal{H}'$ of size $\binom{d+1}{2}$

such that whenever $\min_\tau \mathrm{val}_{\mathcal{I}'}(\tau) \geq S(\mathrm{val}_{\mathcal{I}'})$ we have:

$$\mathrm{val}_{\mathcal{I}'}(\tau) - C(\mathrm{val}_{\mathcal{I}'}) \equiv \sum_j h_j(\tau)^2 + \mu_{\mathcal{I}'} \qquad \text{where } h_j \in \langle \mathcal{H}' \rangle \text{ and } \mu_{\mathcal{I}'} \geq 0\,.$$

Let $h' \in \mathcal{H}'$. By Proposition 3.3.2 $h'(\tau)$ depends only on some subset W' of size at most k, and possibly on the sign of τ.

Now we restrict the above relaxation to the image of Φ. By Lemma 3.3.3 this does not change the optimum. Using the fact that $\mathrm{val}_{\mathcal{I}}(\sigma) \equiv \mathrm{val}_{\mathcal{I}'}(\Phi(\sigma))$ and setting $\mu_{\mathcal{I}} = \mu_{\mathcal{I}'}$ then gives rise to a new relaxation where whenever $\min_\sigma \mathrm{val}_{\mathcal{I}}(\sigma) \geq S(\mathrm{val}_{\mathcal{I}})$ we have:

$$\mathrm{val}_{\mathcal{I}}(\sigma) - C(\mathrm{val}_{\mathcal{I}}) \equiv \sum_j h_j(\Phi(\sigma))^2 + \mu_{\mathcal{I}} \qquad \text{where } h_j \in \langle \mathcal{H}' \rangle \text{ and } \mu_{\mathcal{I}} \geq 0$$

as $S(\mathrm{val}_{\mathcal{I}}) = S(\mathrm{val}_{\mathcal{I}'})$ and $C(\mathrm{val}_{\mathcal{I}}) = C(\mathrm{val}_{\mathcal{I}'})$ by Lemma 3.3.3. Next for each $h' \in \mathcal{H}'$ define $h : S_n \to \mathbb{R}$ by $h(\sigma) = h'(\Phi(\sigma))$. Since $\Phi(\sigma)$ is even, we then have that each h depends only on the position of some subset $W \subseteq [n]$ of size at most k. Such a function can be written as a degree-k polynomial p in the variables x_{ij} so that $p(x^\sigma) \equiv f(\sigma)$ on the vertices of $P_{TSP}(n)$. Now by Theorem 3.3.4 we have that $p \simeq_{(Q_n, 2k-1)} h$. Since $\mu_{\mathcal{I}} \geq 0$ it is clearly the square of a (constant) polynomial, and we conclude that whenever $\min_\sigma \mathrm{val}_{\mathcal{I}}(\sigma) \leq S(\mathrm{val}_{\mathcal{I}})$ we have:

$$f_{\mathcal{I}}(x) - \min f_{\mathcal{I}}/\rho \simeq_{(\mathcal{Q}_n, 2k-1)} \sum_p p(x)^2$$

which is precisely the statement that the $(2k-1)$-level Lasserre relaxation for $P_{TSP}(n)$ is a ρ-approximation. □

3.3.1 Low-degree certificates for tour ideal membership

In this section we prove Theorem 3.3.4 showing that every degree-d polynomial identically zero over *TSP tours* is congruent to 0 within degree $O(d)$.

Note that any partial tour τ can be thought of as a partial matching M in $K_{n,n}$, namely if $\tau(i) = j$, then M includes the edge (i, j). Because of this, it is not surprising that the proof proceeds in a very similar manner to Section 3.2.4, and hereafter we shall always refer to partial matchings on $K_{n,n}$ rather than on K_n.

For a partial matching M, let $x_M \coloneqq \prod_{e \in M} x_e$ denote the product of edge variables for the edges in M. The first step is to reduce every polynomial to a linear combination of the x_M.

Lemma 3.3.5. *For every polynomial F there is a polynomial F' with $\deg F' \leq \deg F$ and $F \simeq_{(\mathcal{Q}_n, \deg F)} F'$, where all monomials of F have the form x_M for some partial matching M.*

Proof. It is enough to prove the lemma when F is a monomial: $F = \prod_{e \in A} x_e^{k_e}$ for a set $A \subseteq E[K_{n,n}]$ of edges with multiplicities $k_e \geq 1$. From $x_e^2 \simeq_2 x_e$ it follows that $x_e^k \simeq_k x_e$ for all $k \geq 1$, hence $F \simeq_{\deg F} \prod_{e \in A} x_e$, proving the claim if A is a partial matching. If A is not a partial matching, then there are distinct $e, f \in A$ with a common vertex, hence $x_e x_f \simeq_2 0$ and $F \simeq_{\deg F} 0$. □

The rest of the proof proceeds identically to Theorem 3.2.9, but we let the symmetric group act on polynomials slightly differently. If $K_{n,n} = U_n \cup V_n$ is the bipartite decomposition of $K_{n,n}$, then we only let the permutation group act on the labels of vertices of U_n, i.e. $\sigma x_{(a,b)} = x_{(\sigma(a),b)}$. We show that under this action, symmetrized polynomials are congruent to a constant, which can again be seen in the same sequence of lemmas:

Lemma 3.3.6. *For any partial matching M on $2d$ vertices and a vertex $a \in U_n$ not covered by M, we have*

$$x_M \simeq_{(\mathcal{Q}_n, d+1)} \sum_{\substack{M_1 = M \cup \{a,u\} \\ v \in V_n \setminus (M \cap V_n)}} x_{M_1}. \tag{3.7}$$

Proof. We use the generators $\sum_v x_{av} - 1$ to add variables corresponding to edges at a, and

then use $x_{av}x_{bv}$ to remove monomials not corresponding to a partial matching:

$$x_M \simeq_{(\mathcal{Q}_n,d+1)} x_M \sum_{v\in V_n} x_{av} \simeq_{(\mathcal{Q}_n,d+1)} \sum_{\substack{M_1=M\cup\{a,v\}\\ v\in V_n\setminus(M\cap V_n)}} x_{M_1}.$$

□

This leads to a similar congruence using all containing matchings of a larger size:

Lemma 3.3.7. *For any partial matching M of $2d$ vertices and $d \le k \le n$, we have*

$$x_M \simeq_{(\mathcal{Q}_n,k)} \frac{1}{\binom{n-d}{k-d}} \sum_{\substack{M'\supset M\\ |M'|=k}} x_{M'} \tag{3.8}$$

Proof. We use induction on $k-d$. The start of the induction is when $k=d$, when the sides of Equation (3.8) are equal.

If $k > d$, let $a \in U_n$ be a fixed vertex not covered by M. Applying Lemma 3.3.6 to M and a followed by the inductive hypothesis gives:

$$x_M \simeq_{(\mathcal{Q}_n,d+1)} \sum_{\substack{M_1=M\cup\{a,u\}\\ u\in V_n\setminus(M\cap V_n)}} x_{M_1} \simeq_{(\mathcal{Q}_n,k)} \frac{1}{\binom{n-d-1}{k-d-1}} \sum_{\substack{M'\supset M_1\\ |M'|=k\\ M_1=M\cup\{a,u\}\\ u\in V_n\setminus(M\cap V_n)}} x_{M'}.$$

Averaging over all vertices $a \in U_n$ not covered by M, we obtain

$$\begin{aligned} x_M &\simeq_{(\mathcal{Q}_n,k)} \frac{1}{n-d}\frac{1}{\binom{n-d-1}{k-d-1}} \sum_{\substack{M' \supset M_1 \\ |M'|=k \\ M_1 = M \cup \{a,u\} \\ a \in U_n \setminus (M \cap U_n) \\ u \in V_n \setminus (M \cap V_n)}} x_{M'} \\ &= \frac{1}{n-d}\frac{1}{\binom{n-d-1}{k-d-1}}(k-d) \sum_{\substack{M' \supset M \\ |M'|=k}} x_{M'} \\ &= \frac{1}{\binom{n-d}{k-d}} \sum_{\substack{M' \supset M \\ |M'|=k}} x_{M'}. \end{aligned}$$

□

Corollary 3.3.8. *For any polynomial F, there is a constant c_F with $\sum_{\sigma \in S_n} \sigma F \simeq_{(\mathcal{Q}_n, \deg F)} c_F$.*

Proof. In view of Lemma 3.3.5, it is enough to prove the claim for $F = x_M$ for some partial matching M on $2k$ vertices, which is an easy application of Lemma 3.3.7 with $d = 0$:

$$\sum_{\sigma \in S_n} \sigma x_M = (n-k)! \sum_{M' :\, |M'|=k} x_{M'} \simeq_k (n-k)! \binom{n}{k}.$$

□

The next lemma will allow us to apply induction:

Lemma 3.3.9. *If L is a polynomial with $L \simeq_{(\mathcal{Q}_{n-2}, d)} 0$ and a, b are the additional vertices in $\mathcal{Q}_n$ then $Lx_{ab}x_{ba} \simeq_{(\mathcal{Q}_n, d+2)} 0$.*

Proof. It is enough to prove the claim when L is from $\mathcal{Q}_{n-2}$. For $L = x_e^2 - x_e$, $L = x_{uv}x_{uw}$, and $L = x_{uv}x_{wv}$ the claim is trivial, as then $L \in \mathcal{Q}_n$. The remaining cases are (1) $L = \sum_{u \in U_{n-2}} x_{uv} - 1$ for some $v \in V_{n-2}$, or (2) $L = \sum_{v \in V_{n-2}} x_{uv} - 1$ for some $u \in U_{n-2}$.. We only deal with the first case, as the second one is analogous. Then

$$Lx_{ab}x_{ba} = \left(\sum_{u \in U_n} x_{uv} - 1 \right) x_{ab}x_{ba} - x_{av}x_{ab}x_{ba} - x_{bv}x_{ab}x_{ba} \simeq_{(\mathcal{Q}_n, d+1)} 0.$$

□

We are now ready to prove Theorem 3.3.4.

Proof of Theorem 3.3.4. We use induction on the degree d of F. The case $d = 0$ is obvious, as then clearly $F = 0$. (Note that $\simeq_{-1}$ is just equality.) The case $d = 1$ rephrased means that the affine space spanned by the characteristic vectors of all perfect matchings is defined by the $\sum_v x_{uv} - 1$ for all vertices u. This follows again from Edmonds's description of the perfect matching polytope by linear inequalities in Edmonds (1965) (valid for any graph in addition to K_{2n} and $K_{n,n}$).

For the case $d \geq 2$ we first prove the following claim:

Claim. If $F \in \langle \mathcal{Q}_n \rangle_I$ is a degree-d polynomial and $\sigma \in S_n$ is a permutation of vertices, then

$$F \simeq_{(\mathcal{Q}_n, 2d-1)} \sigma F.$$

We use induction on the degree. If $d = 0$ or $d = 1$ the claim follows from the corresponding cases $d = 0$ and $d = 1$ of the theorem. For $d \geq 2$ it is enough to prove the claim when σ is a transposition of two vertices a and u. Note that in $F - \sigma F$ all monomials which do not contain an x_e with e incident to a or u on the left cancel:

$$F - \sigma F = \sum_{e\colon e=(a,r) \text{ or } e=(u,r)} L_e x_e \tag{3.9}$$

where each L_e has degree at most $d - 1$. We now show that every summand is congruent to a sum of monomials containing edges incident to both a and u on the left. For example, for $e = \{a, b\}$ in (3.9), we apply the generator $\sum_v x_{uv} - 1$ to find:

$$L_{ab} x_{ab} \simeq_{d+1} L_{ab} x_{ab} \sum_v x_{uv} \simeq_{d+1} \sum_v L_{ab} x_{ab} x_{uv}.$$

Therefore

$$F - \sigma F \simeq_{d+1} \sum_{bv} L'_{bv} x_{ab} x_{uv}$$

for some polynomials L'_{bv} of degree at most $d-1$. We may assume that L'_{bv} does not contain variables x_e with e incident to a, u on the left or b, v on the right, as these can be removed using generators like $x_{ab}x_{ac}$ or $x_{ab}^2 - x_{ab}$. Moreover, since F is zero on all perfect matchings, it can be checked that L'_{bv} is zero on all perfect matchings containing $\{a, b\}$ and $\{u, v\}$. By induction, $L'_{bv} \simeq_{(\mathcal{Q}_{n-4}, 2d-3)} 0$ (identifying K_{n-4} with the graph $K_n \setminus \{a, b, u, v\}$), from which $L'_{bv} \simeq_{(\mathcal{Q}_n, 2d-1)} 0$ follows by two applications of Lemma 3.3.9. (The special case $a = v, b = u$ is also handled by induction and one application of Lemma 3.3.9.) This concludes the proof of the claim.

We now apply the claim followed by Corollary 3.3.8:

$$F \simeq_{2d-1} \frac{1}{n!} \sum_{\sigma \in S_n} \sigma F \simeq_d \frac{c_F}{n!}$$

for a constant c_F. As $F \in \langle \mathcal{Q}_n \rangle_I$, it must be that $c_F = 0$, and therefore $F \simeq_{2d-1} 0$. □

Chapter 4

LAZIFYING CONDITIONAL GRADIENT ALGORITHMS

In light of the lower bounds shown so far, we present in this chapter a modification of conditional gradient algorithms that eschew (approximate) linear optimization. We can summarize the contribution as follows, where the general lazification technique for conditional gradient type algorithms is the main contribution.

(i) **Lazification technique.** Our lazification mechanism allows conditional gradient algorithms to employ the weak separation oracle instead of the linear optimization oracle, returning only a good enough solution or a certificate that such a solution does not exists. This leads to an improved performance in wall clock time, while the asymptotic convergence rates of the non-lazy counterparts are maintained up to small constant factors.

(ii) **Lazified conditional gradient algorithms.** We show our lazification technique on several examples: in Section 4.1.1 we show the most basic example by lazifying Algorithm 1, in Section 4.1.2 we apply the method to the Pairwise Conditional Gradient algorithm (PCG) from Garber and Meshi (2016), in Section 4.1.3 to the Local Linear Optimization Oracle based method from Garber and Hazan (2013) and finally in Section 4.2 to the Online Conditional Gradient method (OCG) from Hazan and Kale (2012).

(iii) **Weak separation through augmentation.** In Section 4.4 we show that in the case when P is a 0/1 polytope we can implement the weak separation oracle (Oracle 1) by

using at most k calls to an even weaker augmentation oracle, where k is the ℓ_1 diameter (or sparsity) of P. An augmentation oracle returns on the same input as the linear optimization oracle either an improving point $\bar{x}$, i.e., $c\bar{x} < cx$ or certifies that such a point does not exist.

(iv) **Experimental results.** We show the computational advantages of different lazified conditional gradient methods over their non-lazy counterparts. We consider several problems of real world relevance such as video colocalization (see Joulin, Tang, and Fei-Fei (2014)), matrix completion and structured regression (see Section 4.5).

Related work

The Frank-Wolfe method was introduced by Frank and Wolfe (1956) and is also known as Conditional Gradient Descent (see Levitin and Polyak 1966). Many different variations have been considered since then and we will only discuss the ones most relevant to this chapter here. Jaggi (2013) shows convergence of the Frank-Wolfe method using approximate optimization, which we presented in Section 1.2.2. After showing a lazified version of the vanilla Frank-Wolfe method we further show how lazification works for more advanced variants, the first such method being the Local Linear Optimization Oracle based method by Garber and Hazan (2013) that achieves linear order of convergence by finding an optimal point for the gradient in a neighborhood of the current iterate. Unfortunately, the large constants in the convergence rate of this method make it impractical to use. Another variant we show a lazified version of is the Pairwise Conditional Gradient method of Garber and Meshi (2016), which uses two linear optimization calls per iteration, one to find a good point to add to the convex combination of the current point and one to remove points or at least decrease their contribution. Our method also applies to the *Block-Coordinate Frank-Wolfe* algorithm, the *Fully-Corrective Frank-Wolfe* algorithm, as well as the *Block-Coordinate Frank-Wolfe* method, however we do not provide the proofs for these methods in this work. For a more detailed overview in general we refer the reader to Jaggi (2013) and for an

overview specifically for versions with global linear convergence to Lacoste-Julien and Jaggi (2015).

The Frank-Wolfe method was also successfully applied in the online learning setting (see Hazan and Kale 2012). We also show a lazified version of this algorithm maintaining the same regret bounds as the original version. Combinatorial convex online optimization has been investigated in a long line of work (see e.g., Kalai and Vempala 2005; Audibert, Bubeck, and Lugosi 2013; Neu and Bartók 2013) and we want to point out that our regret bounds hold in the structured learning setting, i.e., they depend on the ℓ_1-diameter of the feasible region instead of the dimension (see e.g., Cohen and Hazan 2015; Gupta, Goemans, and Jaillet 2016). A good overview of online convex optimization can be found in Hazan (2016).

We want to stress again that our lazified algorithms inherit all requirements, assumptions, and properties of the non-lazy counterparts, including restrictions on the feasible region P as well as smoothness and convexity requirements for the considered objective functions. We will state these requirements in each section for the particular algorithm.

4.1 Lazy Conditional Gradients

We start with the most basic Frank-Wolfe algorithm as a simple example how a conditional gradient algorithm can be lazified by means of a *weak separation oracle*. We will also use the basic variant to discuss various properties and implications. We then show how the more complex Frank-Wolfe algorithms in Garber and Hazan (2013) and Garber and Meshi (2016) can be lazified. Throughout this section $\|\cdot\|$ denotes the ℓ_2-norm.

4.1.1 Lazy Conditional Gradients: a basic example

We start with lazifying the original Frank-Wolfe algorithm (arguably the simplest conditional gradient algorithm), adapting the baseline argument from Jaggi (2013, Theorem 1). While

the vanilla version has suboptimal convergence rate $O(1/T)$, its simplicity makes it an illustrative example of the main idea of lazification. The lazy algorithm (Algorithm 2) maintains an upper bound Φ_t on the convergence rate, guiding its eagerness for progress when searching for an improving vertex v_t. If the oracle provides an improving vertex v_t we refer to this as a *positive call* and we call it a *negative call* otherwise.

Algorithm 2 Lazy Conditional Gradients (LCG)

Require: smooth convex f function with curvature C, $x_1 \in P$ start vertex, LPsep_P weak linear separation oracle, accuracy $K > 1$, initial upper bound Φ_0
Ensure: x_t points in P
1: **for** $t = 1$ **to** $T - 1$ **do**
2: $\quad \Phi_t \leftarrow \frac{\Phi_{t-1} + \frac{C\gamma_t^2}{2}}{1+\frac{\gamma_t}{K}}$
3: $\quad v_t \leftarrow \text{LPsep}_P(\nabla f(x_t), x_t, \Phi_t, K)$
4: $\quad$ **if** $v_t =$ **false then**
5: $\quad\quad x_{t+1} \leftarrow x_t$
6: $\quad$ **else**
7: $\quad\quad x_{t+1} \leftarrow (1-\gamma_t)x_t + \gamma_t v_t$
8: $\quad$ **end if**
9: **end for**

The step size γ_t is chosen to (approximately) minimize Φ_t in Line 2; roughly Φ_{t-1}/KC.

Theorem 4.1.1. *Assume f is convex and smooth with curvature C. Then Algorithm 2 with $\gamma_t = \frac{2(K^2+1)}{K(t+K^2+2)}$ has convergence rate*

$$f(x_t) - f(x^*) \leq \frac{2\max\{C, \Phi_0\}(K^2+1)}{t+K^2+2}, \tag{4.1}$$

where x^* is a minimum point of f over P.

Proof. We prove by induction that

$$f(x_t) - f(x^*) \leq \Phi_{t-1}.$$

The claim is clear for $t = 1$ by the choice of Φ_0. Assuming the claim is true for t, we prove it for $t+1$. We distinguish two cases depending on the return value of the weak separation

oracle in Line 3.

When the oracle returns an improving solution v_t, which we call the positive case, then $\nabla f(x_t)(x_t - v_t) \geq \Phi_t / K$, which is used in the second inequality below. The first inequality follows by smoothness of f, and the third inequality by the induction hypothesis:

$$\begin{aligned} f(x_{t+1}) - f(x^*) &\leq f(x_t) - f(x^*) + \gamma_t \nabla f(x_t)(v_t - x_t) + \frac{C\gamma_t^2}{2} \\ &\leq f(x_t) - f(x^*) - \gamma_t \frac{\Phi_t}{K} + \frac{C\gamma_t^2}{2} \\ &\leq \Phi_{t-1} - \gamma_t \frac{\Phi_t}{K} + \frac{C\gamma_t^2}{2} \\ &= \Phi_t, \end{aligned}$$

When the oracle returns no improving solution, then in particular $\nabla f(x_t)(x_t - x^*) \leq \Phi_t$, hence by Line 5

$$f(x_{t+1}) - f(x^*) = f(x_t) - f(x^*) \leq \nabla f(x_t)(x_t - x^*) = \Phi_t. \tag{4.2}$$

Finally, using the specific values of γ_t we prove the upper bound

$$\Phi_{t-1} \leq \frac{2\max\{C, \Phi_0\}(K^2+1)}{t + K^2 + 2} \tag{4.3}$$

by induction on t. The claim is obvious for $t = 1$. The induction step is an easy computation relying on the definition of Φ_t on Line 2:

$$\begin{aligned} \Phi_t = \frac{\Phi_{t-1} + \frac{C\gamma_t^2}{2}}{1 + \frac{\gamma_t}{K}} &\leq \frac{\frac{2\max\{C,\Phi_0\}(K^2+1)}{t+K^2+2} + \frac{\max\{C,\Phi_0\}\gamma_t^2}{2}}{1 + \frac{\gamma_t}{K}} \\ = 2\max\{C, \Phi_0\}(K^2+1) \frac{1 + \frac{\gamma_t}{2K}}{\left(1 + \frac{\gamma_t}{K}\right)(t + K^2 + 2)} &\leq \frac{2\max\{C, \Phi_0\}(K^2+1)}{t + K^2 + 3}. \end{aligned} \tag{4.4}$$

Here the second equation follows via plugging-in the choice for γ_t for one of the γ_t in the quadratic term and last inequality follows from $t \geq 1$ and the concrete choice of γ_t. □

Remark 4.1.2 (Discussion of the weak separation oracle). A few remarks are in order:

(i) *Interpretation of weak separation oracle.* The weak separation oracle provides new *extreme points* (or vertices) v_t that ensure necessary progress to converge at the proposed rate Φ_t or it certifies that we are already Φ_t-close to the optimal solution. It is important to note that the two cases in Oracle 1 are not mutually exclusive: the oracle might return $y \in P$ with $c(x-y) > \Phi/K$ (positive call: returning a vertex y with improvement Φ/K), while still $c(x-z) \leq \Phi$ for all $z \in P$ (negative call: certifying that there is no vertex z that can improve by Φ). This a desirable property as it makes the separation problem much easier and the algorithm works with either answer in the ambiguous case.

(ii) *Choice of K.* The K parameter can be used to bias the oracle towards positive calls, i.e., returning improving directions. We would also like to point out that the algorithm above as well as those below will also work for $K = 1$, however we later show that we can use an even weaker oracle to realize a weak separation oracle if $K > 1$ in Section 4.4 and for consistency, we require $K > 1$ throughout. In the case $K = 1$ the two cases in the oracle are mutually exclusive.

(iii) *Effect of caching and early termination.* When realizing the weak separation oracle, the actual linear optimization oracle has to be only called if none of the previously seen vertices (or atoms) satisfies the separation condition. Moreover, the weak separation oracle has to only produce a satisfactory solution and not an approximately optimal one. These two properties are responsible for the observed speedup (see Figure 4.26). Moreover, the convex combinations of vertices of P that represent the solutions x_t are extremely sparse as we reuse (cached) vertices whenever possible.

(iv) *Dual certificates.* By not computing an approximately optimal solution, we give up dual optimality certificates. For a given point $x \in P$, let $g(x) := \max_{v \in P} \nabla f(x)(x-v)$ denote the *Wolfe gap*. We have $f(x) - f(x^*) \leq g(x)$ where $x^* = \operatorname{argmin}_{x \in P} f(x)$

by convexity. In those rounds t where we obtain an improving vertex we have no information about $g(x_t)$. However, if the oracle returns *false* in round t, then we obtain the dual certificate $f(x_t) - f(x^*) \leq g(x_t) \leq \Phi_t$.

(v) *Rate of convergence.* A close inspection of the algorithm utilizing the weak separation oracle suggests that the algorithm converges only at the worst-case convergence rate that we propose with the Φ_t sequence. This however is only an artifact of the simplified presentation for the proof of the worst-case rate. We can easily adjust the algorithm to implicitly perform a search over the rate Φ_t combined with line search for γ. This leads to a parameter-free variant of Algorithm 2 and comes at the expense of a (small!) constant factor deterioration of the worst-case rate guarantee; see Remark 4.1.3.(iii) as well as Section 4.3.

Remark 4.1.3 (Implementation improvements). Note that there are various obvious improvements to Algorithm 2 for actual implementations. These improvements do not affect the theoretical (worst-case) performance and for the sake of clarity of the exposition we did not include them in Algorithm 2; see Section 4.3 for the parameter-free version including (most of) these improvements.

(i) First of all, we can improve the update of Φ_t, updating it with the actual gap closed, rather than the pessimistic update via the lower bound on gap closed, i.e., we can update $\Phi_t \leftarrow \Phi_t - (f(x_t) - f(x_{t+1}))$, whenever we calculated a new point x_{t+1}.

(ii) Moreover, we can better utilize information from negative oracle calls (i.e., when the oracle returns **false**): if the oracle utilizes linear optimization at its core, then a negative oracle call will certify $\nabla f(x_t)(x_t - v) \leq \Phi_t$ via maximizing $\nabla f(x_t)v$ with $v \in P$, i.e., the linear optimization oracle computes $g(x_t)$ and we can reset $\Phi_t \leftarrow g(x_t)$. If v^* realizes the Wolfe gap, which is obtained as a byproduct of the above linear maximization, we can further use v^* to make a step: rather than executing line 5, we

can execute line 7 with $v_t = v^*$. By doing so we maximize the use of information obtained from a negative oracle call.

(iii) Finally, we can optimize the management of Φ_t. To obtain a better upper bound Φ_0, we can solve $v^* := \operatorname{argmax}_{v \in P} \nabla f(x_1)v$ at the expense of one extra LP call and set $\Phi_0 := \nabla f(x_1)(x_1 - v^*) = g(x_1)$. Alternatively, we can perform binary search over Φ_0 until the weak separation oracle produces an actual step. If $\bar{\Phi}$ is the value of the search for which we observe the first step, we can reset $\Phi_0 := 2\bar{\Phi}$ and we have $f(x_1) - f(x^*) \leq g(x_1) \leq 2\bar{\Phi}$.

Furthermore, we can change the strategy for managing Φ_t as follows: we keep the value of Φ_t fixed in line 2 and perform line search for γ. Whenever, we observe a negative oracle call, we set the current Φ_t to $\frac{1}{2}g(x_t)$ obtained from the negative call. As such, we ensure $\Phi_t < g(x_t) \leq 2\Phi_t$, which biases the algorithm towards (much cheaper) positive calls. Convergence is ensured by observing that an $\text{LPsep}_P(\cdot, \cdot, \Phi/2, K)$ oracle is an $\text{LPsep}_P(\cdot, \cdot, \Phi, K/2)$ oracle for which the theorem directly applies. With this strategy we maintain the same theoretical (worst-case) convergence up to a constant factor, however in case a faster convergence is possible, we adapt to that rate.

4.1.2 Lazy Pairwise Conditional Gradients

In this section we provide a lazy variant (Algorithm 3) of the Pairwise Conditional Gradient algorithm from Garber and Meshi (2016), using separation instead of linear optimization. We make identical assumptions: the feasible region is a $0/1$ polytope given in the form $P = \{x \in \mathbb{R}^n \mid 0 \leq x \leq \mathbf{1}, Ax = b\}$, where $\mathbf{1}$ denotes the all-one vector of compatible dimension; in particular all vertices of P have only $0/1$ entries.

Observe that Algorithm 3 calls the linear separation oracle LPsep on the Cartesian product of P with itself. Choosing the objective function as in Line 5 allows us to simultaneously find an improving direction and an away-step direction.

Algorithm 3 Lazy Pairwise Conditional Gradients (LPCG)

Require: polytope P, smooth and S-strongly convex function f with curvature C, accuracy $K > 1$, η_t non-increasing step-sizes

Ensure: x_t points

1: $x_1 \in P$ arbitrary and $\Phi_0 \geq f(x_1) - f(x^*)$
2: **for** $t = 1, \ldots, T$ **do**
3: define $\tilde{\nabla} f(x_t) \in \mathbb{R}^m$ as follows:

$$\tilde{\nabla} f(x_t)_i := \begin{cases} \nabla f(x_t)_i & \text{if } (x_t)_i > 0 \\ -\infty & \text{if } (x_t)_i = 0 \end{cases}$$

4: $\Phi_t \leftarrow \frac{2\Phi_{t-1} + \eta_t^2 C}{2 + \frac{\eta_t}{K\Delta_t}}$
5: $c_t \leftarrow \left(\nabla f(x_t), -\tilde{\nabla} f(x_t)\right)$
6: $(v_t^+, v_t^-) \leftarrow \text{LPsep}_{P \times P}\left(c_t, (x_t, x_t), \frac{\Phi_t}{\Delta_t}, K\right)$
7: **if** $(v_t^+, v_t^-) =$ **false then**
8: $x_{t+1} \leftarrow x_t$
9: **else**
10: $\tilde{\eta}_t \leftarrow \max\{2^{-\delta} \mid \delta \in \mathbb{Z}_{\geq 0},\ 2^{-\delta} \leq \eta_t\}$
11: $x_{t+1} \leftarrow x_t + \tilde{\eta}_t (v_t^+ - v_t^-)$
12: **end if**
13: **end for**

Theorem 4.1.4. *Let x^* be a minimum point of f in P, and Φ_0 an upper bound of $f(x_1) - f(x^*)$. Furthermore, let $M_1 \coloneqq \sqrt{\frac{S}{8\,\mathrm{card}(x^*)}}$, $M_2 \coloneqq KC/2$, $\kappa \coloneqq \min\{\frac{M_1}{2M_2}, 1/\sqrt{\Phi_0}\}$, $\eta_t \coloneqq \kappa\sqrt{\Phi_{t-1}}$ and $\Delta_t \coloneqq \sqrt{\frac{2\,\mathrm{card}(x^*)\Phi_{t-1}}{S}}$, then Algorithm 3 has convergence rate*

$$f(x_{t+1}) - f(x^*) \leq \Phi_t \leq \Phi_0 \left(\frac{1+B}{1+2B}\right)^t,$$

where $B \coloneqq \kappa \cdot \frac{M_1}{2K}$.

We recall a technical lemma for the proof.

Lemma 4.1.5 (Garber and Meshi 2016, Lemma 2)**.** *Let $x, y \in P$. There exists vertices v_i of P such that $x = \sum_{i=1}^k \lambda_i v_i$ and $y = \sum_{i=1}^k (\lambda_i - \gamma_i) v_i + \left(\sum_{i=1}^k \gamma_i\right) z$ with $\gamma_i \in [0, \lambda_i]$, $z \in P$ and $\sum_{i=1}^k \gamma_i \leq \sqrt{\mathrm{card}(y)}\|x - y\|$.*

Proof of Theorem 4.1.4. The feasibility of the iterates x_t is ensured by Line 10 and the monotonicity of the sequence $\{\eta_t\}_{t\geq 1}$ with the same argument as in Garber and Meshi (2016, Lemma 1 and Observation 2).

We first show by induction that

$$f(x_{t+1}) - f(x^*) \leq \Phi_t.$$

For $t = 0$ we have $\Phi_0 \geq f(x_1) - f(x^*)$. Now assume the statement for some $t \geq 0$. In the negative case (Line 8), we use the guarantee of Oracle 1 to get

$$c_t((x_t, x_t) - (z_1, z_2)) \leq \frac{\Phi_t}{\Delta_t}$$

for all $z_1, z_2 \in P$, which is equivalent to (as $c_t(x_t, x_t) = 0$)

$$\tilde{\nabla} f(x_t) z_2 - \nabla f(x_t) z_1 \leq \frac{\Phi_t}{\Delta_t}$$

and therefore

$$\nabla f(x_t)(\tilde{z}_2 - z_1) \leq \frac{\Phi_t}{\Delta_t}, \tag{4.5}$$

for all $\tilde{z}_2, z_1 \in P$ with $\operatorname{supp}(\tilde{z}_2) \subseteq \operatorname{supp}(x_t)$. We further use Lemma 4.1.5 to write $x_t = \sum_{i=1}^k \lambda_i v_i$ and $x^* = \sum_{i=1}^k (\lambda_i - \gamma_i) v_i + \sum_{i=1}^k \gamma_i z$ with $\gamma_i \in [0, \lambda_i]$, $z \in P$ and $\sum_{i=1}^k \gamma_i \leq \sqrt{\operatorname{card}(x^*)} \|x_t - x^*\| \leq \sqrt{\frac{2 \operatorname{card}(x^*) \Phi_{t-1}}{S}} = \Delta_t$, using the induction hypothesis and the strong convexity in the second inequality. Then

$$f(x_{t+1}) - f(x^*) = f(x_t) - f(x^*) \leq \nabla f(x_t)(x_t - x^*) = \sum_{i=1}^k \gamma_i (v_i - z) \cdot \nabla f(x_t) \leq \Phi_t,$$

where we used Equation 4.5 for the last inequality.

For the positive case (Lines 10 and 11) we get, using first smoothness of f, then $\eta_t/2 < \tilde{\eta}_t \leq \eta_t$ and $\nabla f(x_t)(v_t^+ - v_t^-) \leq -\Phi_t/(\Delta_t K)$, and finally the definition of Φ_t:

$$\begin{aligned} f(x_{t+1}) - f(x^*) &= f(x_t) - f(x^*) + f(x_t + \tilde{\eta}_t (v_t^+ - v_t^-)) - f(x_t) \\ &\leq \Phi_{t-1} + \tilde{\eta}_t \nabla f(x_t)(v_t^+ - v_t^-) + \frac{\tilde{\eta}_t^2 C}{2} \\ &\leq \Phi_{t-1} - \frac{\eta_t}{2} \cdot \frac{\Phi_t}{\Delta_t K} + \frac{\eta_t^2 C}{2} = \Phi_t. \end{aligned}$$

Plugging in the values of η_t and Δ_t to the definition of Φ_t gives the desired bound.

$$\Phi_t = \frac{2\Phi_{t-1} + \eta_t^2 C}{2 + \frac{\eta_t}{K \Delta_t}} = \Phi_{t-1} \frac{1 + \kappa^2 M_2 / K}{1 + \kappa M_1 / K} \leq \Phi_{t-1} \frac{1+B}{1+2B} \leq \Phi_0 \left(\frac{1+B}{1+2B} \right)^t. \quad \square$$

4.1.3 Lazy Local Conditional Gradients

In this section we provide a lazy version (Algorithm 4) of the conditional gradient algorithm from Garber and Hazan (2013). Let $P \subseteq \mathbb{R}^n$ be any polytope, D denote an upper bound on the ℓ_2-diameter of P, and $\mu \geq 1$ be the affine invariant of P from Garber and Hazan (2013). As the algorithm is not affine invariant by nature, we need a non-invariant version of

smoothness: Recall that a convex function f is β-*smooth* if

$$f(y) - f(x) \leq \nabla f(x)(y - x) + \beta \|y - x\|^2/2.$$

Algorithm 4 Lazy Local Conditional Gradients (LLCG)

Require: feasible polytope P, β-smooth and S-strongly convex function f, parameters K, S, β, μ; diameter D
Ensure: x_t points

$x_1 \in P$ arbitrary and $\Phi_0 \geq f(x_1) - f(x^*)$
$\alpha \leftarrow \frac{S}{2K\beta n\mu^2}$
for $t = 1, \ldots, T$ **do**
  $\Phi_t \leftarrow \frac{\Phi_{t-1} + \frac{\beta}{2}\alpha^2 \min\{n\mu^2 r_t^2, D^2\}}{1+\alpha/K}$
  $r_t \leftarrow \sqrt{\frac{2\Phi_{t-1}}{S}}$
  $p_t \leftarrow \text{LLPsep}_P(\nabla f(x_t), x_t, r_t, \Phi_t, K)$
  if $p_t =$ **false then**
    $x_{t+1} \leftarrow x_t$
  else
    $x_{t+1} \leftarrow x_t + \alpha(p_t - x_t)$
  end if
end for

As an intermediary step, we first implement a *local weak separation oracle* in Algorithm 5, a *local* version of Oracle 1, analogously to the local linear optimization oracle in Garber and Hazan (2013). To this end, we recall a technical lemma from Garber and Hazan (2013).

Lemma 4.1.6 (Garber and Hazan 2013, Lemma 7)**.** *Let $P \subseteq \mathbb{R}^n$ be a polytope and $v_1, \ldots, v_N$ be its vertices. Let $x, y \in P$ and $x = \sum_{i=1}^N \lambda_i v_i$ a convex combination of the vertices of P. Then there are numbers $0 \leq \gamma_i \leq \lambda_i$ and $z \in P$ satisfying*

$$y - x = -\sum_{i \in [N]} \gamma_i v_i + \left(\sum_{i \in [N]} \gamma_i\right) z \tag{4.6}$$

$$\sum_{i \in [N]} \gamma_i \leq \frac{\sqrt{n}\mu}{D} \|x - y\|. \tag{4.7}$$

Now we prove the correctness of the weak local separation algorithm.

Algorithm 5 Weak Local Separation $\mathrm{LLPsep}_P(c, x, r, \Phi, K)$

Require: $c \in \mathbb{R}^n$ linear objective, $x \in P$ point, $r > 0$ radius, $\Phi > 0$ objective value
Ensure: Either (1) $y \in P$ with $\|x - y\| \leq \sqrt{n}\mu r$ and $c(x - y) > \Phi/K$, or (2) **false**: $c(x - z) \leq \Phi$ for all $z \in P \cap \mathbb{B}_r(x)$.

1: $\Delta \leftarrow \min\left\{\frac{\sqrt{n}\mu}{D} r, 1\right\}$
2: Decompose x: $x = \sum_{j=1}^{M} \lambda_j v_j,\ \lambda_j > 0, \sum_j \lambda_j = 1$.
3: Sort vertices: $i_1, \ldots, i_M \quad cv_{i_1} \geq \cdots \geq cv_{i_M}$.
4: $k \leftarrow \min\{k : \sum_{j=1}^{k} \lambda_{i_j} \geq \Delta\}$
5: $p_- \leftarrow \sum_{j=1}^{k-1} \lambda_{i_j} v_{i_j} + \left(\Delta - \sum_{j=1}^{k-1} \lambda_{i_j}\right) v_{i_k}$
6: $v^* \leftarrow \mathrm{LPsep}_P\left(c, \frac{p_-}{\Delta}, \frac{\Phi}{\Delta}\right)$
7: **if** $v^* =$ **false then**
8: **return false**
9: **else**
10: **return** $y \leftarrow x - p_- + \Delta v^*$
11: **end if**

Lemma 4.1.7. *Algorithm 5 is correct. In particular* $\mathrm{LLPsep}_P(c, x, r, \Phi, K)$

(i) returns either an $y \in P$ with $\|x - y\| \leq \sqrt{n}\mu r$ and $c(x - y) > \Phi/K$,

(ii) or establishes $c(x - z) \leq \Phi$ for all $z \in P \cap \mathbb{B}_r(x)$.

Proof. We first consider the case when the algorithm exits in Line 10. Observe that $y \in P$ since y is a convex combination of vertices of P. Moreover by construction of y we can write $y = \sum_{j=1}^{M} (\lambda_{i_j} - \gamma_j) v_{i_j} + \Delta v^*$ with $\Delta = \sum_{j=1}^{M} \gamma_j \leq \frac{\sqrt{n}\mu}{D} r$. Therefore

$$\|x - y\| = \left\| \sum_{j=1}^{M} \gamma_j v_{i_j} - \Delta v^* \right\| \leq \sum_{j=1}^{M} \gamma_j \|v_{i_j} - v^*\| \leq \sqrt{n}\mu r.$$

Finally using the guarantee of LPsep_P we get

$$c(x - y) = \Delta c\left(\frac{p_-}{\Delta} - v^*\right) \geq \frac{\Phi}{K}.$$

If the algorithm exits in Line 8, we use Lemma 4.1.6 to decompose any $y \in P \cap \mathbb{B}_r(x)$

in the following way:

$$y = \sum_{i=1}^{N} (\lambda_i - \gamma_i) v_i + \left(\sum_{i=1}^{N} \gamma_i \right) z,$$

with $z \in P$ and $\sum_{i=1}^{N} \gamma_i \leq \frac{\sqrt{n}\mu}{D} \|x - y\| \leq \Delta$. Since $\sum_{i=1}^{N} \lambda_i = 1 \geq \Delta$, there are numbers $\gamma_i \leq \eta_i^- \leq \lambda_i$ with $\sum_{i=1}^{N} \eta_i^- = \Delta$. Let

$$\tilde{p}_- := \sum_{i=1}^{N} \eta_i^- v_i, \tag{4.8}$$

$$\tilde{p}_+ := y - x + \tilde{p}_- = \sum_{i=1}^{N} (\eta_i^- - \gamma_i) v_i + \sum_{i=1}^{N} \gamma_i z, \tag{4.9}$$

so that $\tilde{p}_+ / \Delta \in P$. To bound the function value we first observe that the choice of p_- in the algorithm assures that $cu \leq cp_-$ for all $u = \sum_{i=1}^{N} \eta_i v_i$ with $\sum_{i=1}^{N} \eta_i = \Delta$ and all $\eta_i \geq 0$. In particular, $c\tilde{p}_- \leq cp_-$. The function value of the positive part $\tilde{p}_+$ can be bounded with the guarantee of LPsep_P:

$$c \left(\frac{p_-}{\Delta} - \frac{\tilde{p}_+}{\Delta} \right) \leq \frac{\Phi}{\Delta},$$

i.e., $c(p_- - \tilde{p}_+) \leq \Phi$. Finally combining these bounds gives

$$c(x - y) = c\left(\tilde{p}_- - \tilde{p}_+\right) \leq c(p_- - \tilde{p}_+) \leq \Phi$$

as desired. □

We are ready to examine the Conditional Gradient algorithm based on LLPsep_P:

Theorem 4.1.8. *Algorithm 4 converges with the following rate:*

$$f(x_{t+1}) - f(x^*) \leq \Phi_t \leq \Phi_0 \left(\frac{1 + \alpha/(2K)}{1 + \alpha/K} \right)^t.$$

Proof. The proof is similar to the proof of Theorem 4.1.4. We prove this rate by induction. For $t = 0$ the choice of Φ_0 guarantees that $f(x_1) - f(x^*) \leq \Phi_0$. Now assume the theorem

holds for $t \geq 0$. With strong convexity and the induction hypothesis we get

$$\|x_t - x^*\|^2 \leq \frac{2}{S}(f(x_t) - f(x^*)) \leq \frac{2}{S}\Phi_{t-1} = r_t^2,$$

i.e., $x^* \in P \cap \mathbb{B}_{r_t}(x_t)$. In the negative case, i.e., when $p_t = \textbf{false}$, then case (ii) of Lemma 4.1.7 applies:

$$f(x_{t+1}) - f(x^*) = f(x_t) - f(x^*) \leq \nabla f(x_t)(x_t - x^*) \leq \Phi_t.$$

In the positive case, i.e., when Line 10 is executed, we get the same inequality via:

$$\begin{aligned} f(x_{t+1}) - f(x^*) &\leq \Phi_{t-1} + \alpha \nabla f(x_t)(p_t - x_t) + \frac{\beta}{2}\alpha^2 \|x - p_t\|^2 \\ &\leq \Phi_{t-1} - \alpha\frac{\Phi_t}{K} + \frac{\beta}{2}\alpha^2 \min\{n\mu^2 r_t^2, D^2\} \\ &= \Phi_t. \end{aligned}$$

Therefore using the definition of α and r_t we get the desired bound:

$$\Phi_t \leq \frac{\Phi_{t-1} + \frac{\beta}{2}\alpha^2 r_t^2 n\mu^2}{1 + \alpha/K} = \Phi_{t-1}\left(\frac{1 + \alpha/(2K)}{1 + \alpha/K}\right) \leq \Phi_0\left(\frac{1 + \alpha/(2K)}{1 + \alpha/K}\right)^t. \qquad \square$$

4.2 Lazy Online Conditional Gradients

In this section we lazify the Online Conditional Gradient algorithm (OCG) of Hazan and Kale (2012) over arbitrary polytopes $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$, resulting in Algorithm 6. We slightly improve constant factors by replacing Hazan and Kale (2012, Lemma 3.1) with a better estimation via solving a quadratic inequality arising from strong convexity. In this section the norm $\|\cdot\|$ can be arbitrary.

Theorem 4.2.1. *Let $0 \leq b, s < 1$. Let $K > 1$ be an accuracy parameter. Assume f_t is L-Lipschitz, and smooth with curvature at most Ct^{-b}. Let $D := \max_{y_1, y_2 \in P}\|y_1 - y_2\|$*

Algorithm 6 Lazy Online Conditional Gradients (LOCG)

Require: f_t functions, $x_1 \in P$ start vertex, LPsep_P weak linear separation oracle, parameters K, C, b, S, s; diameter D

Ensure: x_t points

for $t = 1$ **to** $T - 1$ **do**
  $\nabla_t \leftarrow \nabla f_t(x_t)$
  if $t = 1$ **then**
    $h_1 \leftarrow \min\{\|\nabla_1\|^* D, 2\|\nabla_1\|^{*2}/S\}$
  else
    $h_t \leftarrow \Phi_{t-1} + \min\left\{\|\nabla_t\|^* D, \frac{\|\nabla_t\|^{*2}}{St^{1-s}} + 2\sqrt{\frac{\|\nabla_t\|^{*2}}{2St^{1-s}}\left(\frac{\|\nabla_t\|^{*2}}{2St^{1-s}} + \Phi_{t-1}\right)}\right\}$
  end if
  $\Phi_t \leftarrow \frac{h_t + \frac{Ct^{1-b}\gamma_t^2}{2(1-b)}}{1+\frac{\gamma_t}{K}}$
  $v_t \leftarrow \text{LPsep}_P(\sum_{i=1}^t \nabla f_i(x_t), x_t, \Phi_t, K)$
  if $v_t =$ **false then**
    $x_{t+1} \leftarrow x_t$
  else
    $x_{t+1} \leftarrow (1-\gamma_t)x_t + \gamma_t v_t$
    $\Phi_t \leftarrow h_t - \sum_{i=1}^t f_i(x_t) + \sum_{i=1}^t f_i(x_{t+1})$
  end if
end for

denote the diameter of P in norm $\|\cdot\|$. Then the following hold for the points x_t computed by Algorithm 6 where x_T^* is the minimizer of $\sum_{t=1}^T f_t$:

(i) With the choice

$$\gamma_t = t^{-(1-b)/2},$$

the x_t satisfy

$$\frac{1}{T}\sum_{t=1}^{T}(f_t(x_T) - f_t(x_T^*)) \leq AT^{-(1-b)/2}, \tag{4.10}$$

where

$$A := \frac{CK}{2(1-b)} + L(K+1)D.$$

(ii) Moreover, if all the f_t are St^{-s}-strongly convex, then with the choice

$$\gamma_t = t^{(b+s-2)/3},$$

the x_t satisfy

$$\frac{1}{T}\sum_{t=1}^{T}(f_t(x_T) - f_t(x_T^*)) \leq AT^{-(2(1+b)-s)/3}, \tag{4.11}$$

where

$$A := 2\left((K+1)(K+2)\frac{L^2}{S} + \frac{CK}{2(1-b)}\right).$$

Proof. We prove only Claim (ii), as the proof of Claim (i) is similar and simpler. Let $F_T := \sum_{t=1}^T f_t$. Furthermore, let $\overline{h}_T := AT^{1-(2(1+b)-s)/3}$ be T times the right-hand side of Equation (4.11). In particular, F_T is S_T-strongly convex, and smooth with curvature at most C_{F_T} where

$$C_{F_T} := \frac{CT^{1-b}}{1-b} \geq C\sum_{t=1}^{T} t^{-b}, \qquad S_T := ST^{1-s} \leq S\sum_{t=1}^{T} t^{-s}. \tag{4.12}$$

We prove $F_t(x_t) - F_t(x_t^*) \leq h_t \leq \overline{h}_t$ by induction on t. The case $t = 1$ is clear. Let $\overline{\Phi}_t$ denote the value of Φ_t in Line 8, while we reserve Φ_t to denote its value as used in Line 6.

We start by showing $F_t(x_{t+1}) - F_t(x_t^*) \leq \Phi_t \leq \overline{\Phi}_t$. We distinguish two cases depending on v_t from Line 9. If v_t is false, then $\Phi_t = \overline{\Phi}_t$ and the weak separation oracle asserts $\max_{y \in P} \nabla F_t(x_t)(x_t - y) \leq \Phi_t$, which combined with the convexity of F_t provides

$$F_t(x_{t+1}) - F_t(x_t^*) = F_t(x_t) - F_t(x_t^*) \leq \nabla F_t(x_t)(x_t - x_{t^*}) \leq \Phi_t = \overline{\Phi}_t.$$

Otherwise v_t is a vertex of P, then Line 14 and the induction hypothesis provides $F_t(x_{t+1}) - F_t(x_t^*) \leq h_t + F_t(x_{t+1}) - F_t(x_t) = \Phi_t$. To prove $\Phi_t \leq \overline{\Phi}_t$, we apply the smoothness of F_t followed by the inequality provided by the choice of v_t:

$$F_t(x_{t+1}) - F_t(x_t) - \frac{C_{F_t}\gamma_t^2}{2} \leq \nabla F_t(x_t)(x_{t+1} - x_t) = \gamma_t \nabla F_t(x_t)(v_t - x_t) \leq -\frac{\gamma_t \overline{\Phi}_t}{K}.$$

Rearranging provides the inequality below.

$$\Phi_t = h_t + F_t(x_{t+1}) - F_t(x_t) \leq h_t - \frac{\gamma_t \overline{\Phi}_t}{K} + \frac{C_{F_t}\gamma_t^2}{2} = \overline{\Phi}_t.$$

For later use, we bound the difference between $\overline{h}_t$ and $\overline{\Phi}_t$ using the value of parameters, $h_t \leq \overline{h}_t$, and $\gamma_t \leq 1$:

$$\overline{h}_t - \overline{\Phi}_t \geq \overline{h}_t - \frac{\overline{h}_t + \frac{C_{F_t}\gamma_t^2}{2}}{1 + \frac{\gamma_t}{K}} = \frac{\frac{\overline{h}_t \gamma_t}{K} - \frac{C_{F_t}\gamma_t^2}{2}}{1 + \frac{\gamma_t}{K}} \geq \frac{\frac{\overline{h}_t \gamma_t}{K} - \frac{C_{F_t}\gamma_t^2}{2}}{1 + \frac{1}{K}} = \frac{A - \frac{CK}{2(1-b)}}{K+1} t^{[2s-(1+b)]/3}.$$

We now apply $F_t(x_{t+1}) - F_t(x_t^*) \leq \Phi_t$, together with convexity of f_{t+1}, and the minimality $F_t(x_t^*) \leq F_t(x_{t+1}^*)$ of x_t^*, followed by strong convexity of F_{t+1}:

$$\begin{aligned} F_{t+1}(x_{t+1}) - F_{t+1}(x_{t+1}^*) &\leq (F_t(x_{t+1}) - F_t(x_t^*)) + (f_{t+1}(x_{t+1}) - f_{t+1}(x_{t+1}^*)) \\ &\leq \Phi_t + \|\nabla_{t+1}\|^* \cdot \|x_{t+1} - x_{t+1}^*\| \\ &\leq \Phi_t + \|\nabla_{t+1}\|^* \sqrt{\frac{2}{S_{t+1}}(F_{t+1}(x_{t+1}) - F_{t+1}(x_{t+1}^*))}. \end{aligned} \tag{4.13}$$

Solving the quadratic inequality provides

$$F_{t+1}(x_{t+1}) - F_{t+1}(x^*_{t+1}) \leq \Phi_t + \frac{\|\nabla_{t+1}\|^{*2}}{S_{t+1}} + 2\sqrt{\frac{\|\nabla_{t+1}\|^{*2}}{2S_{t+1}}\left(\frac{\|\nabla_{t+1}\|^{*2}}{2S_{t+1}} + \Phi_t\right)}. \tag{4.14}$$

From Equation (4.13), ignoring the last line, we also obtain $F_{t+1}(x_{t+1}) - F_{t+1}(x^*_{t+1}) \leq \Phi_t + \|\nabla_{t+1}\|^* D$ via the estimate $\|x_{t+1} - x^*_{t+1}\| \leq D$. Thus $F_{t+1}(x_{t+1}) - F_{t+1}(x^*_{t+1}) \leq h_{t+1}$, by Line 6, as claimed.

Now we estimate the right-hand side of Equation (4.14) by using the actual value of parameters, the estimate $\|\nabla_{t+1}\|^* \leq L$ and the inequality $s + b \leq 2$. Actually, we estimate a proxy for the right-hand side. Note that A was chosen to satisfy the second inequality.

$$\begin{aligned}
\frac{L^2}{S_{t+1}} + 2\sqrt{\frac{L^2}{2S_{t+1}}\overline{h}_t} &\leq \frac{L^2}{St^{1-s}} + 2\sqrt{\frac{L^2}{2St^{1-s}}\overline{h}_t} \leq \frac{L^2}{S}t^{[2s-(1+b)]/3} + 2\sqrt{\frac{L^2}{2St^{1-s}}\overline{h}_t} \\
&= \left(\frac{L^2}{S} + \sqrt{2\frac{L^2}{S}A}\right)t^{[2s-(1+b)]/3} \leq \frac{A - \frac{CK}{2(1-b)}}{K+1}t^{[2s-(1+b)]/3} \\
&\leq \overline{h}_t - \overline{\Phi}_t \leq \overline{h}_t - \Phi_t.
\end{aligned}$$

In particular, $\frac{L^2}{2S_{t+1}} + \Phi_t \leq \overline{h}_t$ hence combining with Equation (4.14) we obtain

$$\begin{aligned}
h_{t+1} &\leq \Phi_t + \frac{L^2}{S_{t+1}} + 2\sqrt{\frac{L^2}{2S_{t+1}}\left(\frac{L^2}{2S_{t+1}} + \Phi_t\right)} \\
&\leq \Phi_t + \frac{L^2}{S_{t+1}} + 2\sqrt{\frac{L^2}{2S_{t+1}}\overline{h}_t} \\
&\leq \overline{h}_t \leq \overline{h}_{t+1}.
\end{aligned}$$

□

4.2.1 Stochastic and Adversarial Versions

Complementing the offline algorithms from Section 4.1, we will now derive various versions from the online case. The presented cases here are similar to those in Hazan and Kale (2012) and thus we state them without proof.

For stochastic cost functions f_t, we obtain bounds from Theorem 4.2.1 (i) similar to Hazan and Kale (2012, Theorems 4.1 and 4.3) (with δ replaced by δ/T in the bound to correct an inaccuracy in the original argument). The proof is analogous and hence omitted, but note that $\|y_1 - y_2\|_2 \leq \sqrt{\|y_1 - y_2\|_1 \|y_1 - y_2\|_\infty} \leq \sqrt{k}$ for all $y_1, y_2 \in P$.

Corollary 4.2.2. *Let f_t be convex functions sampled i.i.d. with expectation $\mathbb{E}[f_t] = f^*$, and $\delta > 0$. Assume that the f_t are L-Lipschitz in the 2-norm.*

(i) If all the f_t are smooth with curvature at most C, then Algorithm 6 applied to the f_t (with $b = 0$) yields with probability $1 - \delta$

$$\sum_{t=1}^{T} f^*(x_t) - \min_{x \in P} \sum_{t=1}^{T} f^*(x) \leq O\left(C\sqrt{T} + Lk\sqrt{nT \log(nT^2/\delta) \log T}\right). \quad (4.15)$$

(ii) Without any smoothness assumption, Algorithm 6 (applied to smoothenings of the f_t) provides with probability $1 - \delta$

$$\sum_{t=1}^{T} f^*(x_t) - \min_{x \in P} \sum_{t=1}^{T} f^*(x) \leq O\left(\sqrt{n}LkT^{2/3} + Lk\sqrt{nT \log(nT^2/\delta) \log T}\right). \quad (4.16)$$

Similar to Hazan and Kale (2012, Theorem 4.4), from Theorem 4.2.1 (ii) we obtain the following regret bound for adversarial cost functions with an analogous proof.

Corollary 4.2.3. *For any L-Lipschitz convex cost functions f_t, Algorithm 6 applied to the functions $\tilde{f}_t(x) := \nabla f_t(x_t)x + \frac{2L}{\sqrt{k}} t^{-1/4} \|x - x_1\|_2^2$ (with $b = s = 1/4$, $C = L\sqrt{k}$, $S = L/\sqrt{k}$, and Lipschitz constant $3L$) achieving regret*

$$\sum_{t=1}^{T} f_t(x_t) - \min_{x \in P} \sum_{t=1}^{T} f_t(x) \leq O(L\sqrt{k}T^{3/4}) \quad (4.17)$$

with at most T calls to the weak separation oracle.

Note that the gradient of the $\tilde{f}_t$ are easily computed via the formula $\nabla \tilde{f}_t(x) = \nabla f_t(x_t) + 4Lt^{-1/4}(x - x_1)/\sqrt{k}$, particularly because the gradient of the f_t need not be recomputed,

so that we obtain a weak separation-based stochastic gradient descent algorithm, where we only have access to the f_t through a stochastic gradient oracle, while retaining all the favorable properties of the Frank-Wolfe algorithm with a convergence rate $O(T^{-1/4})$ (c.f., Garber and Hazan 2013).

4.3 Parameter-free Conditional Gradients via Weak Separation

In this section we provide the Conditional Gradient algorithm with the implementation improvements that we sketched in Section 4.1.1 for completeness. In particular the obtained algorithm is parameter-free; note that K is a parameter of the oracle and not the algorithms. Similar improvements apply to the other algorithms and the adaptation of those is straightforward and left to the interested reader.

If the weak separation oracle is realized via a linear minimization oracle at its core, then we can slightly change the specification of the oracle. It still maintains the advantage of caching and early termination, however we utilize the information obtained from the linear optimization calls better. We present the adjusted oracle in Oracle 2. The major difference is that in the negative case, where a Φ/K-improving vertex does not exist, the oracle does not just return *false* as before but returns also a maximizing vertex. Note that this information is obtained from the linear optimization oracle as a byproduct in the negative case. The negative case (2) in Oracle 2 can be replaced by an approximate upper bound if desired; see Remark 4.3.4.

Oracle 2 Weak Separation Oracle $\mathrm{LPsep}_P(c, x, \Phi, K)$

Require: $c \in \mathbb{R}^n$ linear objective, $x \in P$ point, $K \geq 1$ accuracy, $\Phi > 0$ objective value;
Ensure: $y \in P$ vertex with either (1) $c(x - y) > \Phi/K$, or (2) $y = \operatorname{argmax}_{y \in P} c(x - z) \leq \Phi$.

We now present the *Lazy Conditional Gradient* algorithm (see Algorithm 2) with im-

provements in Algorithm 7 below. We stress that the worst-case convergence rate is identical up to a small constant factor and at the same time the algorithm becomes parameter-free.

Here we perform the initial bound tightening of Φ_0 with a single extra LP call, which can be also done approximately as long as Φ_0 is a valid upper bound. Alternative one can perform binary search via the weak separation oracle as described earlier. Note that Algorithm 7 does not include the primal tightening $\Phi_t \leftarrow \Phi_t - (f(x_t) - f(x_{t+1}))$ as outlined in Remark 4.1.3.(i), as we want to keep the Φ_t unchanged in the case of a positive oracle call to promote more aggressive improvements: the guaranteed improvement is quadratic or linear in Φ_t for positive calls depending on the magnitude of Φ_t.

Algorithm 7 Parameter-free Lazy Conditional Gradients (LCG)

Require: smooth convex f function, $x_1 \in P$ start vertex, LPsep_P weak linear separation oracle, accuracy $K > 1$
Ensure: x_t points in P
1: $\Phi \leftarrow \max_{x \in P} \nabla f(x_1)(x_1 - x)$ {Initial bound tightening}
2: **for** $t = 1$ **to** $T - 1$ **do**
3: $v_t \leftarrow \text{LPsep}_P(\nabla f(x_t), x_t, \Phi, K)$
4: Line search over γ_t to minimize $f((1 - \gamma_t)x_t + \gamma_t v_t)$
5: $x_{t+1} \leftarrow (1 - \gamma_t)x_t + \gamma_t v_t$
6: **if not** $\nabla f(x_t)(x_t - v_t) > \Phi/K$ **then**
7: $\Phi \leftarrow \frac{\nabla f(x_t)(x_t - v_t)}{2}$ {Update Φ via dual gap and halve}
8: **end if**
9: **end for**

Theorem 4.3.1 shows that Algorithm 7 converges in the worst-case at a rate identical to Algorithm 2 (up to a small constant factor).

Theorem 4.3.1. *Algorithm 7 converges at a rate proportional to* $1/t$. *In particular to achieve a bound* $f(x_t) - f(x^*) \leq \varepsilon$, *the number of required steps is upper bounded by*

$$t \leq 4K\lceil \log \Phi_0 / KC \rceil + \frac{4K^2 C}{\varepsilon}.$$

Proof. Let C be the curvature of the smooth convex function f and Φ_t be Φ at the end of iteration t. Let us first establish that $f(x_{t+1}) - f(x^*) \leq 2\Phi_t$ at the end of each iteration:

Initially, for $t = 0$, we have by Line 1 and convexity $\Phi_0 = \operatorname{argmax}_{x \in P} \nabla f(x_1)(x_1 - x) \geq \nabla f(x_1)(x_1 - x^*) \geq f(x_1) - f(x^*)$. This establishes the case $t = 0$, so that for induction we assume that $f(x_t) - f(x^*) \leq 2\Phi_{t-1}$ holds at the beginning of iteration t.

In the positive case (case (1) in Oracle 2). Via line search it follows that in any iteration the primal gap is non-increasing, i.e.,

$$f(x_{t+1}) - f(x^*) \leq f(x_t) - f(x^*) \leq 2\Phi_{t-1} = 2\Phi_t \tag{4.18}$$

using the induction hypothesis and the fact that Φ does not change in the case of positive oracle calls. For the negative case (case (2) in Oracle 2) by an argument similar to the positive case we get

$$f(x_{t+1}) - f(x^*) \leq f(x_t) - f(x^*) \leq \nabla f(x_t)(x_t - x^*) \leq \nabla f(x_t)(x_t - v_t) = 2\Phi_t,$$

using convexity for the second inequality and $2\Phi_t = \nabla f(x_t)(x_t - v_t)$ by Line 7 for the equality, which completes the induction.

It is left to show that Φ_t decreases fast enough, which is done by bounding the number of steps in which Φ is not halved. First observe that in an iteration with a positive oracle call we have

$$f(x_t) - f(x_{t+1}) \geq \gamma_t \frac{\Phi_{t-1}}{K} - \frac{C}{2}\gamma_t^2 \geq \begin{cases} \frac{\Phi_{t-1}^2}{2CK^2} & \text{if } \frac{\Phi_{t-1}}{KC} < 1 \\ \frac{\Phi_{t-1}}{K} - \frac{C}{2} \geq \frac{C}{2} & \text{if } \frac{\Phi_{t-1}}{KC} \geq 1 \end{cases}, \tag{4.19}$$

via smoothness and optimality of γ_t from the line search; in both cases a non-negative change. Let t' be the number of consecutive steps in which Φ is not halved, then lower bounding the progress in each step via (4.19), we have

$$2\Phi_{t-1} \geq f(x_t) - f(x^*) \geq \sum_{\tau=t}^{t+t'-1} f(x_\tau) - f(x_{\tau+1}) \geq \begin{cases} t' \frac{\Phi_{t-1}^2}{2CK^2} & \text{if } \frac{\Phi_{t-1}}{KC} < 1 \\ t' \left(\frac{\Phi_{t-1}}{K} - \frac{C}{2} \right) & \text{if } \frac{\Phi_{t-1}}{KC} \geq 1 \end{cases},$$

which gives in the case $\Phi_{t-1} < KC$ that $t' \leq \frac{4CK^2}{\Phi_{t-1}}$ and in the case $\Phi_{t-1} \geq KC$ that

$$t' \leq \frac{2\Phi_{t-1}}{\frac{\Phi_{t-1}}{K} - \frac{C}{2}} = \frac{4K\Phi_{t-1}}{2\Phi_{t-1} - KC} \leq \frac{4K\Phi_{t-1}}{2\Phi_{t-1} - \Phi_{t-1}} = 4K.$$

Adding up the number of steps gives the desired rate: we have at most $\log(\Phi_0/\varepsilon)$ scaling phases, where $\log(\cdot)$ is the binary logarithm and ε is the (additive) accuracy. Further in each scaling phase with scaling parameter Φ we make at most $\frac{4CK^2}{\Phi}$ positive steps if $\Phi \leq KC$ and at most $4K$ positive steps if $\Phi \geq KC$. Thus after a total of

$$\begin{aligned}
\bar{t} &:= \sum_{\ell \in [\lceil \log \Phi_0/KC \rceil]} 4K + \sum_{\ell \in [\lceil \log KC/\varepsilon \rceil]} \frac{2^{\ell-1} \cdot 4CK^2}{KC} \\
&= 4K\lceil \log \Phi_0/KC \rceil + 4K \sum_{\ell \in [\lceil \log KC/\varepsilon \rceil]} 2^{\ell-1} \\
&\leq 4K\lceil \log \Phi_0/KC \rceil + \frac{4K^2C}{\varepsilon}
\end{aligned}$$

steps we have $f(x_{\bar{t}}) - f(x^*) \leq \varepsilon$. □

Remark 4.3.2. Observe that Algorithm 7 might converge much faster due to the aggressive halving of the rate. In fact, Algorithm 7 convergences at a rate that is at most a factor $4K^2$ slower than the rate that the vanilla (non-lazy) Frank-Wolfe algorithm would realize for the same problem. In actual wall-clock time Algorithm 7 is much faster though due to the use of the weaker oracle; see Figure 4.1 for a comparison of the bounds on the Wolfe gap and Section 4.5.3 for more experimental results.

Negative oracle calls tend to be significantly more expensive time-wise than positive oracle calls due to proving dual bounds. The following corollary is an immediate consequence of the argumentation from above:

Corollary 4.3.3. *Algorithm 7 makes at most* $\log(\Phi_0/\varepsilon)$ *negative oracle calls.*

Remark 4.3.4 (Approximate negative calls). In Oracle 2 in the negative call case the oracle returns $y = \operatorname{argmax}_{y \in P} c(x - z) \leq \Phi$. We can relax the exact linear optimization and

replace it by a (weaker) upper bound τ satisfying

$$\text{argmax}_{y \in P}\, c(x - z) \leq \tau \leq \Phi,$$

and adjust Algorithm 7 appropriately; in particular in the negative case the algorithm simply does not update the iterate.

4.4 Weak Separation through Augmentation

So far we realized the weak separation oracle via lazy optimization. We will now create a (weak) separation oracle for integral polytopes, employing an even weaker, so-called augmentation oracle, which only provides an improving solution but provides no guarantee with respect to optimality. We call this approach *lazy augmentation*. This is especially useful when a fast augmentation oracle is available or the vertices of the underlying polytope P are particularly sparse. As before theoretical convergence rates are maintained.

For simplicity of exposition we restrict to $0/1$ polytopes P here. For general integral polytopes, one considers a so-called *directed augmentation oracle*, which can be similarly linearized after splitting variables in positive and negative parts; we refer the interested reader to see Schulz and Weismantel (2002) and Bodic et al. (2015) for an in-depth discussion.

Let k denote the ℓ_1-diameter of P. Upon presentation with a 0/1 solution x and a linear objective $c \in \mathbb{R}^n$, an augmentation oracle either provides an improving 0/1 solution $\bar{x}$ with $c\bar{x} < cx$ or asserts optimality for c:

Oracle 3 Linear Augmentation Oracle AUG_P(c, x)

Require: $c \in \mathbb{R}^n$ linear objective, $x \in P$ vertex,

Ensure: $\bar{x} \in P$ vertex with $c\bar{x} < cx$ when exists, otherwise $\bar{x} = x$

Such an oracle is significantly weaker than a linear optimization oracle but also significantly easier to implement and much faster; we refer the interested reader to Grötschel and

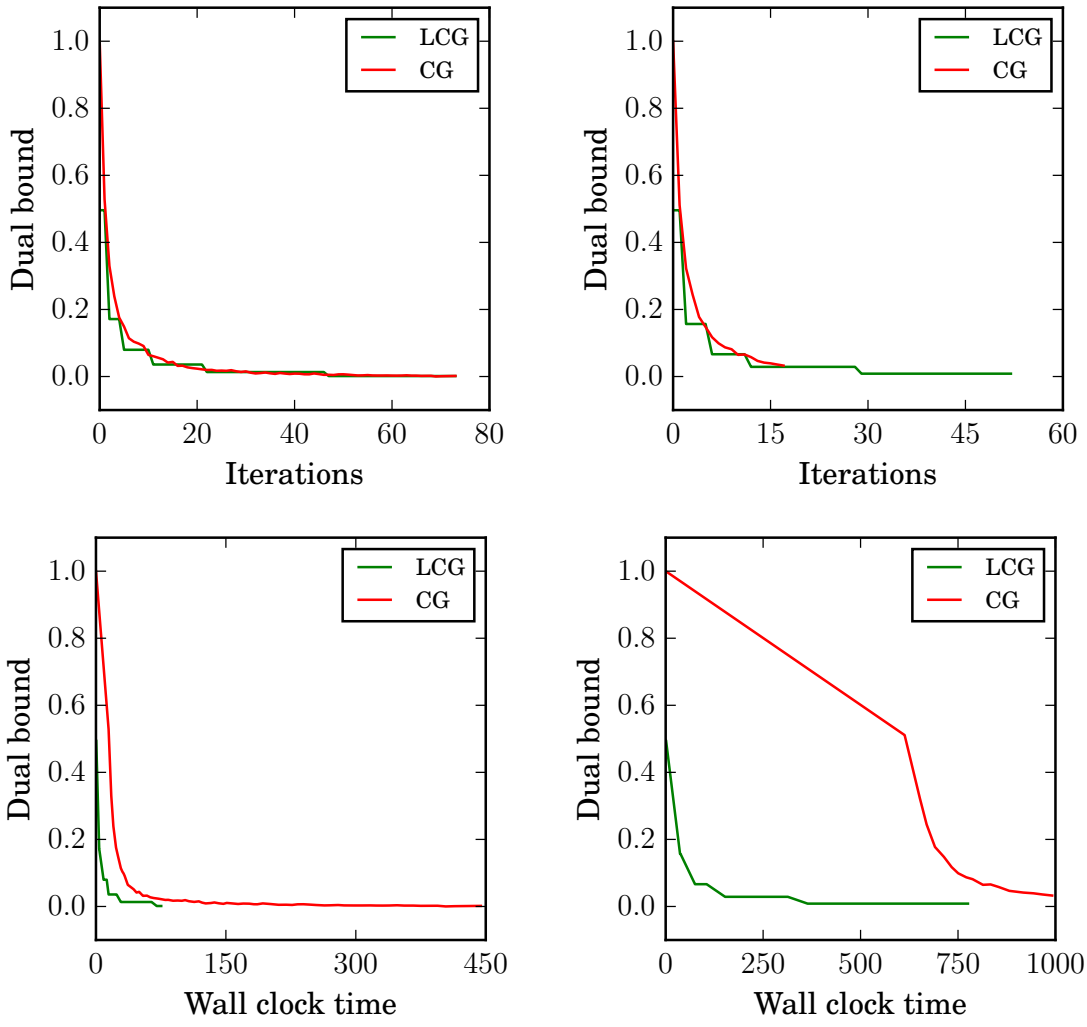

Figure 4.1: Example behavior of parameter-free variant of the Lazy CG algorithm (Algorithm 7) compared to the vanilla Frank-Wolfe algorithm with approximate optimization (denoted by CG). We depict the behavior of the dual bound, i.e., the bound on Wolfe gap. In the top row we report dual bound vs. iterations and in the lower row dual bound vs. wall-clock time. We can see in the top row how the halving of Φ in Algorithm 7 approximates the 'true' Φ from the vanilla Frank-Wolfe. However in walk-clock time the lazy variant achieves significantly better dual bounds in the same time. The example instances here are two maxcut instances and we observe the same behavior across most instances. The time limit was 450s on the left and 1000s on the right.

Lovász (1993), Schulz, Weismantel, and Ziegler (1995), and Schulz and Weismantel (2002) for an extensive list of examples. While augmentation and optimization are polynomially equivalent (even for convex integer programming, see Oertel, Wagner, and Weismantel 2014) the current best linear optimization algorithms based on an augmentation oracle are slow for general objectives. While optimizing an *integral* objective $c \in \mathbb{R}^n$ needs $O(k \log \|c\|_\infty)$ calls to an augmentation oracle (see Schulz, Weismantel, and Ziegler 1995; Schulz and Weismantel 2002; Bodic et al. 2015), a general objective function, such as the gradient in Frank–Wolfe algorithms has only an $O(kn^3)$ guarantee in terms of required oracle calls (e.g., via simultaneous Diophantine approximations Frank and Tardos 1987), which is not desirable for large n. In contrast, here we use an augmentation oracle to perform separation, without finding the optimal solution. Allowing a multiplicative error $K > 1$, we realize an augmentation-based weak separation oracle (see Algorithm 8), which decides given a linear objective function $c \in \mathbb{R}^n$, an objective value $\Phi > 0$, and a starting point $x \in P$, whether there is a $y \in P$ with $c(x-y) > \Phi/K$ or $c(x-y) \leq \Phi$ for all $y \in P$. In the former case, it actually provides a certifying $y \in P$, i.e., with $c(x-y) > \Phi/K$. Note that a constant accuracy K requires a linear number of oracle calls in the diameter k, e.g., $K = (1-1/e)^{-1} \approx 1.582$ needs at most $N \leq k$ oracle calls.

At the beginning, in Line 2, the algorithm has to replace the input point x with an integral point x_0. If the point x is given as a convex combination of integral points, then a possible solution is to evaluate the objective c on these integral points, and choose x_0 the first one with $cx_0 \leq cx$. This can be easily arranged for Frank–Wolfe algorithms as they maintain convex combinations.

Proposition 4.4.1. *Assume $\|y_1 - y_2\|_1 \leq k$ for all $y_1, y_2 \in P$. Then Algorithm 8 is correct, i.e., it outputs either (1) $y \in P$ with $c(x-y) > \Phi/K$, or (2)* **false**. *In the latter case $c(x-y) \leq \Phi$ for all $y \in P$ holds. The algorithm calls AUG_P at most $N \leq \lceil \log(1-1/K) / \log(1-1/k) \rceil$ many times.*

Proof. First note that $(\mathbf{1}-2x)v + \|x\|_1 = \|v-x\|_1$ for $x, v \in \{0,1\}^n$, hence Line 7 is

Algorithm 8 Augmenting Weak Separation $\text{LPsep}_P(c, x, \Phi, K)$

Require: $c \in \mathbb{R}^n$ linear objective, $x \in P$ point, $\Phi > 0$ objective value; $K > 1$ accuracy
Ensure: Either (1) $y \in P$ vertex with $c(x-y) > \Phi/K$, or (2) **false**: $c(x-z) \leq \Phi$ for all $z \in P$.

1: $N \leftarrow \lceil \log(1-1/K)/\log(1-1/k) \rceil$
2: Choose $x_0 \in P$ vertex with $cx_0 \leq cx$.
3: **for** $i = 1$ **to** N **do**
4: **if** $c(x - x_{i-1}) \geq \Phi$ **then**
5: **return** x_{i-1}
6: **end if**
7: $x_i \leftarrow \text{AUG}_P(c + \frac{\Phi - c(x-x_{i-1})}{k}(\mathbf{1} - 2x_{i-1}), x_{i-1})$
8: **if** $x_i = x_{i-1}$ **then**
9: **return false**
10: **end if**
11: **end for**
12: **return** x_N

equivalent to $x_i \leftarrow \text{AUG}_P(c + \frac{\Phi - c(x-x_{i-1})}{k}\|\cdot - x_{i-1}\|_1, x_{i-1})$.

The algorithm obviously calls the oracle at most N times by design, and always returns a value, so we need to verify only the correctness of the returned value. We distinguish cases according to the output.

Clearly, Line 5 always returns an x_{i-1} with $c(x - x_{i-1}) \geq \Phi > [1 - (1-1/k)^N]\Phi$. When Line 9 is executed, the augmentation oracle just returned $x_i = x_{i-1}$, i.e., for all $y \in P$

$$cx_{i-1} \leq cy + \frac{\Phi - c(x - x_{i-1})}{k}\|y - x_{i-1}\|_1 \tag{4.20}$$

$$\leq cy + \frac{\Phi - c(x - x_{i-1})}{k}k \tag{4.21}$$

$$= c(y - x) + cx_{i-1} + \Phi, \tag{4.22}$$

so that $c(x-y) \leq \Phi$, as claimed.

Finally, when Line 12 is executed, the augmentation oracle has found an improving vertex x_i at every iteration, i.e.,

$$cx_{i-1} > cx_i + \frac{\Phi - c(x - x_{i-1})}{k}\|x_i - x_{i-1}\|_1 \geq cx_i + \frac{\Phi - c(x - x_{i-1})}{k}, \tag{4.23}$$

using $\|x_i - x_{i-1}\|_1 \geq 1$ by integrality. Rearranging provides the convenient form

$$\Phi - c(x - x_i) < \left(1 - \frac{1}{k}\right)[\Phi - c(x - x_{i-1})], \tag{4.24}$$

which by an easy induction provides

$$\Phi - c(x - x_N) < \left(1 - \frac{1}{k}\right)^N [\Phi - c(x - x_0)] \leq \left(1 - \frac{1}{K}\right)\Phi, \tag{4.25}$$

i.e., $c(x - x_N) \geq \frac{\Phi}{K}$, finishing the proof. □

4.5 Experiments

We implemented and compared the parameter-free variant of LCG (Algorithm 7) to the standard Frank-Wolfe algorithm (CG). Moreover, we implemented and compared Algorithm 3 (LPCG) to the Pairwise Conditional Gradient algorithm (PCG) variant of Garber and Meshi (2016) as well as implemented and compared Algorithm 6 (LOCG) to the Online Frank-Wolfe algorithm (OCG) of Hazan and Kale (2012). While we did implement the Local Linear Optimization Oracle based variant from Garber and Hazan (2013) as well, the very large constants in the original algorithms made it impractical to run.

We have used $K = 1.1$ and $K = 1$ as multiplicative factors for the weak separation oracle; for the impact of the choice of K see Section 4.5.4. For the baseline algorithms we use inexact variants, i.e., we solve linear optimization problems only approximately. This is a significant speedup in favor of non-lazy algorithms at the (potential) cost of accuracy, while neutral to lazy optimization as it solves an even more relaxed problem anyways. To put things in perspective, the non-lazy baselines could not complete even a single iteration for a significant fraction of the considered problems in the given time frame if we were to exactly solve the linear optimization problems.

The linear optimization oracle over $P \times P$ for LPCG was implemented by calling the

respective oracle over P twice: once for either component. Contrary to the non-lazy version, the lazy algorithms depend on the initial upper bound Φ_0. For the instances that need a very long time to solve the (approximate) linear optimization even once, we used for the lazy algorithms a binary search for Φ_0: starting from a conservative initial value, using the update rule $\Phi_0 \leftarrow \Phi_0/2$ until the separation oracle returns an improvement for the first time and then we start the algorithm with $2\Phi_0$, which is an upper bound on the Wolfe gap and hence also on the primal gap. This initial phase is also included in the reported wall-clock time. Alternatively, if the linear optimization is less time consuming we used a single (approximate) linear optimization at the start to obtain an initial bound on Φ_0 (see e.g., Section 4.3).

In some cases, especially when the underlying feasible region has a high dimension and the (approximate) linear optimization can be solved relatively fast compared to the cost of computing an inner product, we observed that the costs of maintaining the cache was very high. In these cases we reduce the cache size every 100 steps by keeping only the 100 points that were used the most so far. Both, the number of steps and the approximate size of the cache are chosen arbitrarily, however 100 for both worked very well for all our examples. Of course there are many different strategies for maintaining the cache, which could be used here and which could lead to further improvements in performance.

The stopping criteria for each of the experiments is a given wall clock time limit in seconds. The time limit was enforced separately for the main code, and the oracle code so in some cases the actual time used can be larger, when the last oracle call started before the time limit was reached and took longer than the time left.

We implemented all algorithms in `Python 2.7` with critical functions *cythonized* for performance employing `Numpy`. We used these packages from the `Anaconda 4.2.0` distribution as well as `Gurobi 7.0` Gurobi Optimization (2016) as a black box solver for the linear optimization oracle and the weak separation oracle. The latter was implemented via a callback function to stop the optimization as soon as a good enough feasible solution

has been found. The parameters for Gurobi were kept at their default settings except for enforcing the time limit of the tests and setting the acceptable duality gap to 10%, allowing Gurobi to terminate the linear optimization early avoiding the expensive *proof* of optimality. This is used to realize the inexact versions of the baseline algorithms. All experiments were performed on a 16-core machine with Intel Xeon E5-2630 v3 @ 2.40GHz CPUs and 128GB of main memory. While our code does not explicitly use multiple threads, both Gurobi and the numerical libraries use multiple threads internally.

4.5.1 Considered problems

We performed computational tests on a large variety of different problems that are instances of the three machine learning tasks video colocalization, matrix completion and structured regression.

Video colocalization. Video colocalization is the problem of identifying objects in a sequence of multiple frames in a video. In Joulin, Tang, and Fei-Fei (2014) it is shown that video colocalization can be reduced to optimizing a quadratic objective function over a flow or a path polytope , which is the problem we are going to solve. The quadratic functions are of the form $\|Ax - b\|^2$ where we choose the non-zero entries in A according to a density parameter at random and then each of these entries to be $[0, 1]$-uniformly distributed, while b is chosen as a linear combination of the columns of A with random multipliers from $[0, 1]$. For some of the instances we also use $\|x - b\|^2$ as the objective function with $b_i \in [0, 1]$ uniformly at random.

Matrix completion. The formulation of the matrix completion problem we are going to use is the following:

$$\min \sum_{(i,j)\in\Omega} \|X_{i,j} - a_{i,j}\|^2 \qquad \text{s.t.} \quad \|X\|_* \leq R, \tag{4.26}$$

where $\|\cdot\|_*$ denotes the nuclear norm, i.e., $\|A\|_* = tr(\sqrt{A^t A})$. The set Ω, the matrix A with entries $a_{i,j}$, and R are given parameters. Similarly to Lan and Zhou (2014) we generate the $m \times n$ matrix A as the product of A_L of size $m \times r$ and A_R of size $r \times n$. The entries in A_L and A_R are chosen from a standard Gaussian. The set of entries Ω is chosen uniformly of size $s = \min(5r(m+n-r), \lceil 0.99mn \rceil)$. The linear optimization oracle is implemented in this case by a singular value decomposition of the linear objective function.

Structured regression. The structured regression problem consists of solving a quadratic function of the form $\|Ax - b\|^2$ over some structured feasible set or a polytope. We construct the objective functions in the same way as for the video colocalization problem.

We will present in the following two sections the complete set of results for various problems grouped by the different versions of the considered algorithms. Every figure contains two columns, each containing one experiment. We use different measures to report performance: the first row reports loss or function value in wall-clock time (including time spent by the oracle), the second row contains loss or function value in the number of iterations. In some cases we include a row reporting the loss or function value over the number of linear optimization calls. In some other cases we report in another row the dual bound or Wolfe gap in wall-clock time. The last row always reports the cumulative number of calls to the linear optimization oracle for the lazy algorithm. The red line denotes the non-lazy algorithm and the greed line denotes the lazy variants. For each experiment we also report the cache hit rate, which is the number of oracle calls answered with a point from the cache over all oracle calls given in percent.

While we found convergence rates in the number of iterations quite similar (as expected!), we consistently observe a significant speedup in wall-clock time. In particular for many large-scale or hard combinatorial problems, lazy algorithms performed several thousand iterations whereas the non-lazy versions completed only a handful of iterations due to the large time spent approximately solving the linear optimization problem. The observed cache

hit rate was at least 90% in most cases, and often even above 99%.

4.5.2 Online Results

Additionally to the quadratic objective functions we tested the online version on random linear functions $cx + b$ with $c \in [-1, +1]^n$ and $b \in [0, 1]$. We used in each experiment a random sequence of 100 different random loss functions. For online conditional gradient algorithms, in every figure the left column uses linear loss functions, the right one uses quadratic loss functions of the form as described above over the same polytope.

As an instance of the structured regression problem we used the flow-based formulation for Hamiltonian cycles in graphs, i.e., the traveling salesperson problem (TSP) for graphs with 11 and 16 nodes (Figures 4.2 and 4.3). While relatively small, the oracle problem can be solved in reasonable time for these instances. Another instance of the structured regression problem uses the standard formulation of the cut polytope for graphs with 23 and 28 nodes as the feasible region (Figures 4.4 and 4.5). Another set of feasible regions corresponding to NP-hard problems for the structured regression problem we tested our algorithm on are the quadratic unconstrained boolean optimization (QUBO) instances defined on Chimera graphs Dash (2013), which are available at `http://researcher.watson.ibm.com/researcher/files/us-sanjeebd/chimera-data.zip`. The instances are relatively hard albeit their rather small size (Figure 4.6 and 4.7). One instance of the video colocalization problem uses a path polytope from `http://lime.cs.elte.hu/~kpeter/data/mcf/netgen/` that was generated with the netgen graph generator (Figure 4.8). Most of these instances are very large-scale minimum cost flow instances with several tens of thousands nodes in the underlying graphs, therefore solving still takes considerable time despite the problem being in P. We tested on the structured regression problems with the MIPLIB (Achterberg, Koch, and Martin 2006; Koch et al. 2011) instances `eil33-2` (Figure 4.9) and `air04` (Figure 4.10) as feasible regions. Finally for the spanning tree problem, we used the well-known extended formulation with $O(n^3)$ inequalities for an n-node graph.

We considered graphs with 10 and 25 nodes (Figures 4.11 and 4.12).

We observed that while OCG and LOCG converge comparably in the number of iterations, the lazy LOCG performed significantly more iterations; for hard problems, where linear optimization is costly and convergence requires a large number of iterations, this led LOCG converging much faster in wall-clock time. In extreme cases OCG could not complete even a single iteration. This is due to LOCG only requiring *some* good enough solution, whereas OCG requires a stronger guarantee. This is reflected in faster oracle calls for LOCG.

4.5.3 Offline Results

We describe the considered instances in the offline case separately for the vanilla Frank-Wolfe method and the Pairwise Conditional Gradients method.

Vanilla Frank-Wolfe Method We tested the vanilla Frank-Wolfe algorithm on the six video colocalization instances with underlying path polytopes from `http://lime.cs.elte.hu/~kpeter/data/mcf/netgen/` (Figures 4.13, 4.14 and 4.15). In these instances we additionally report the dual bound or Wolfe gap in wall clock time. We further tested the vanilla Frank-Wolfe algorithm on eight instances of the matrix completion problem generated as described above. For these examples we did not use line search. We give the used parameters for each example in the figures below (Figures 4.16, 4.17, 4.18 and 4.19). The last tests for this version were performed on three instances of the structured regression problem, two with the feasible region containing flow-based formulations of Hamiltonian cycles in graphs (Figures 4.20), two on two different cut polytope instances (Figure 4.21) and finally two on two spanning tree instances of different size (Figure 4.22).

Similarly to the online case, we observe a significant speedup of LCG compared to CG, due to the faster iteration of the lazy algorithm.

Pairwise Conditional Gradient Algorithm As we inherit structural restrictions of PCG on the feasible region, the problem repertoire is limited in this case. We tested the Pair-

wise Conditional Gradient algorithm on the structured regression problem with feasible regions from the MIPLIB instances `eil33-2`, `air04`, `eilB101`, `nw04`, `disctom`, `m100n500k4r1` (Figures 4.23, 4.24 and 4.25).

Again similarly to the online case and the vanilla Frank-Wolfe algorithm, we observe a significant improvement in wall-clock time of LPCG compared to CG, due to the faster iteration of the lazy algorithm.

4.5.4 Performance improvements, parameter sensitivity, and tuning

Effect of caching

As mentioned before, lazy algorithms have two improvements: caching and early termination. Here we depict the effect of caching in Figure 4.26, comparing OCG (no caching, no early termination), LOCG (caching and early termination) and LOCG (only early termination). We did not include a caching-only OCG variant, because caching without early termination does not make much sense: in each iteration a new linear optimization problem has to be solved; previous solutions can hardly be reused as they are unlikely to be optimal for the new linear optimization problem.

Effect of K

If the parameter K of the oracle can be chosen, which depends on the actual oracle implementation, then we can increase K to bias the algorithm towards performing more positive calls. At the same time the steps get shorter. As such there is a natural trade-off between the cost of many positive calls vs. a negative call. We depict the impact of the parameter choice for K in Figure 4.27.

Parameter-free vs. textbook variant

For illustrative purposes, we compare the textbook variant of the Lazy Conditional Gradient method (Algorithm 2) with its parameter-free counterpart (Algorithm 7) in Figure 4.28. The parameter-free variant outperforms the textbook variant due to the active management of Φ combined with line search.

Similar parameter-free variants can be derived for the other algorithms; see discussion in Section 4.3.

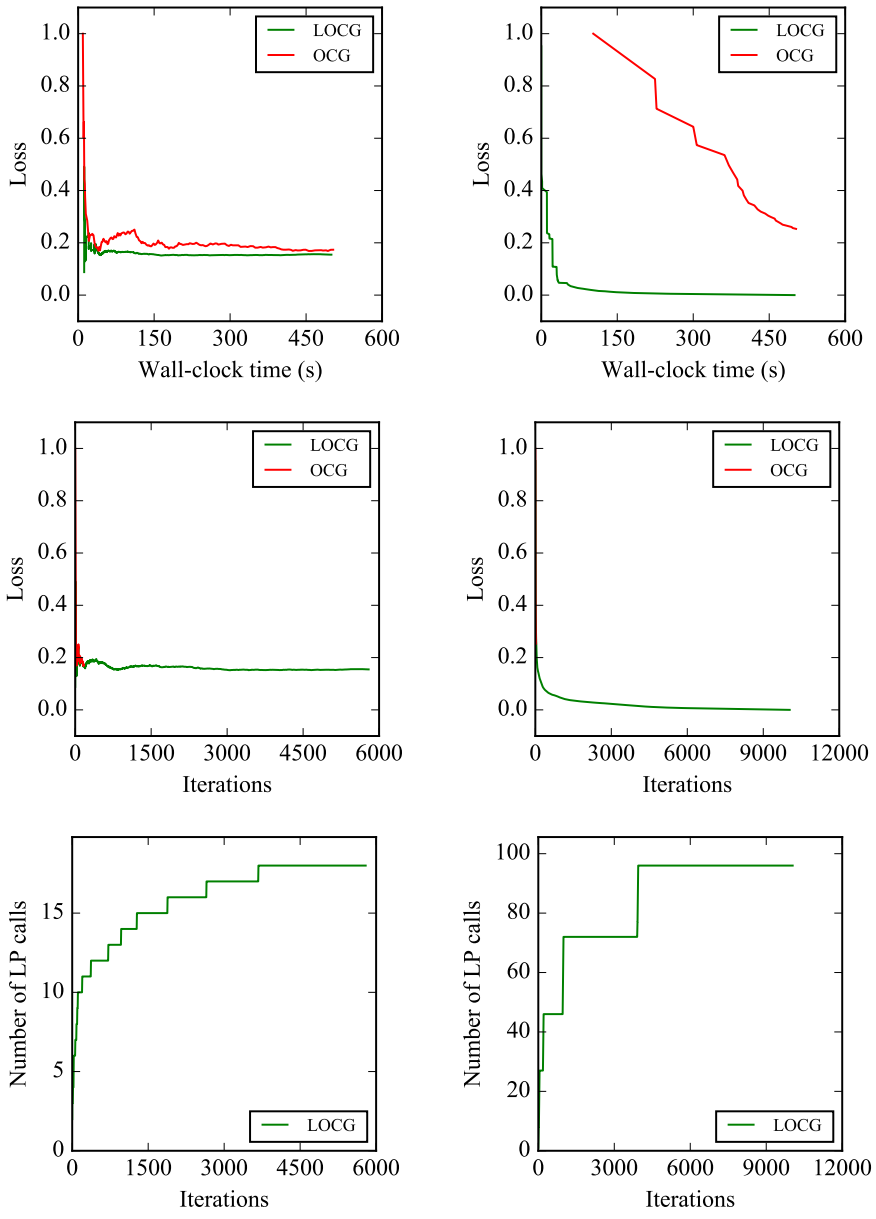

Figure 4.2: LOCG vs. OCG with the TSP polytope for a graph with 11 nodes as the feasible region and with a 500 seconds time limit. OCG completed only a few iterations, resulting in a several times larger final loss for quadratic loss functions. Notice that with time LOCG needed fewer and fewer LP calls.

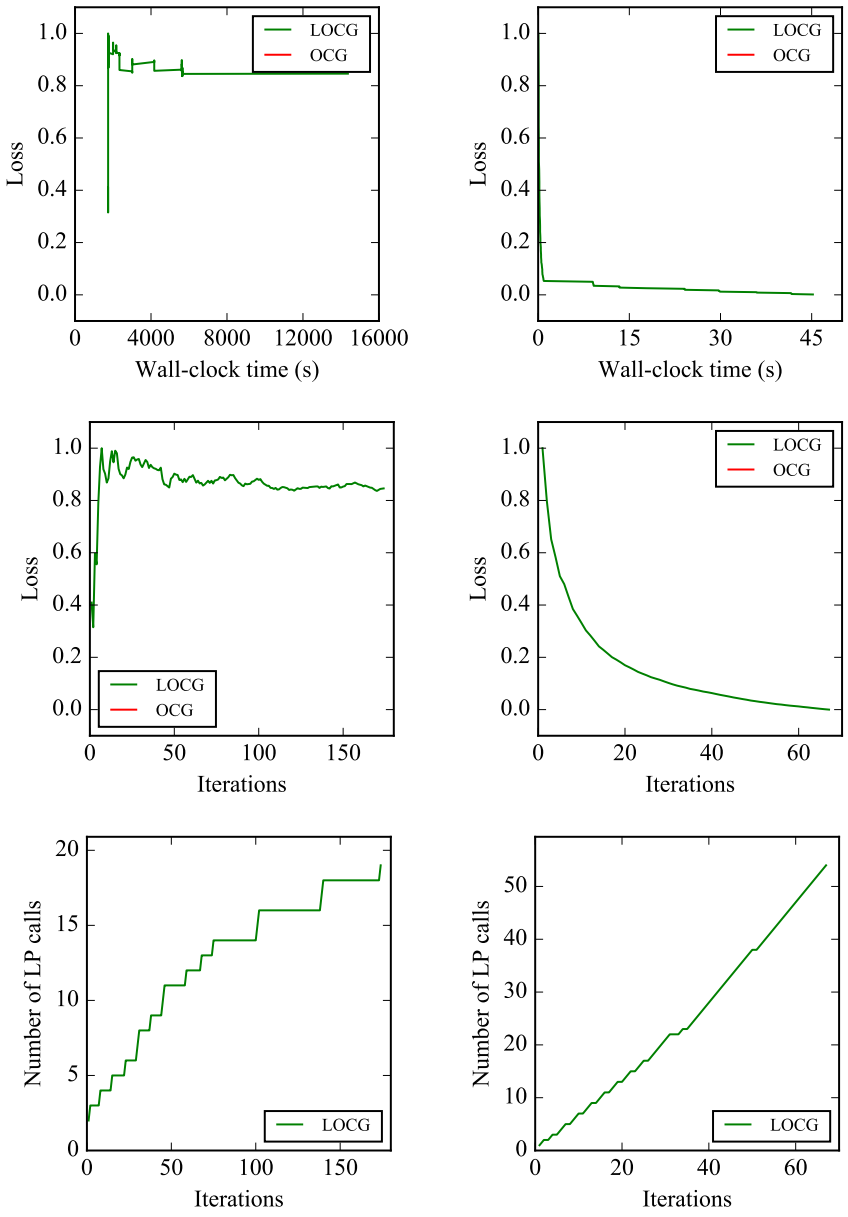

Figure 4.3: LOCG vs. OCG on the TSP polytope for a graph with 16 nodes with a time limit of 7200 seconds. OCG was not able to complete a single iteration and in the quadratic case even LOCG could not complete any more iteration after 50s. The quadratic losses on the right nicely demonstrate speed improvements (mostly) through early termination of the linear optimization as the cache rate is only 20.6%.

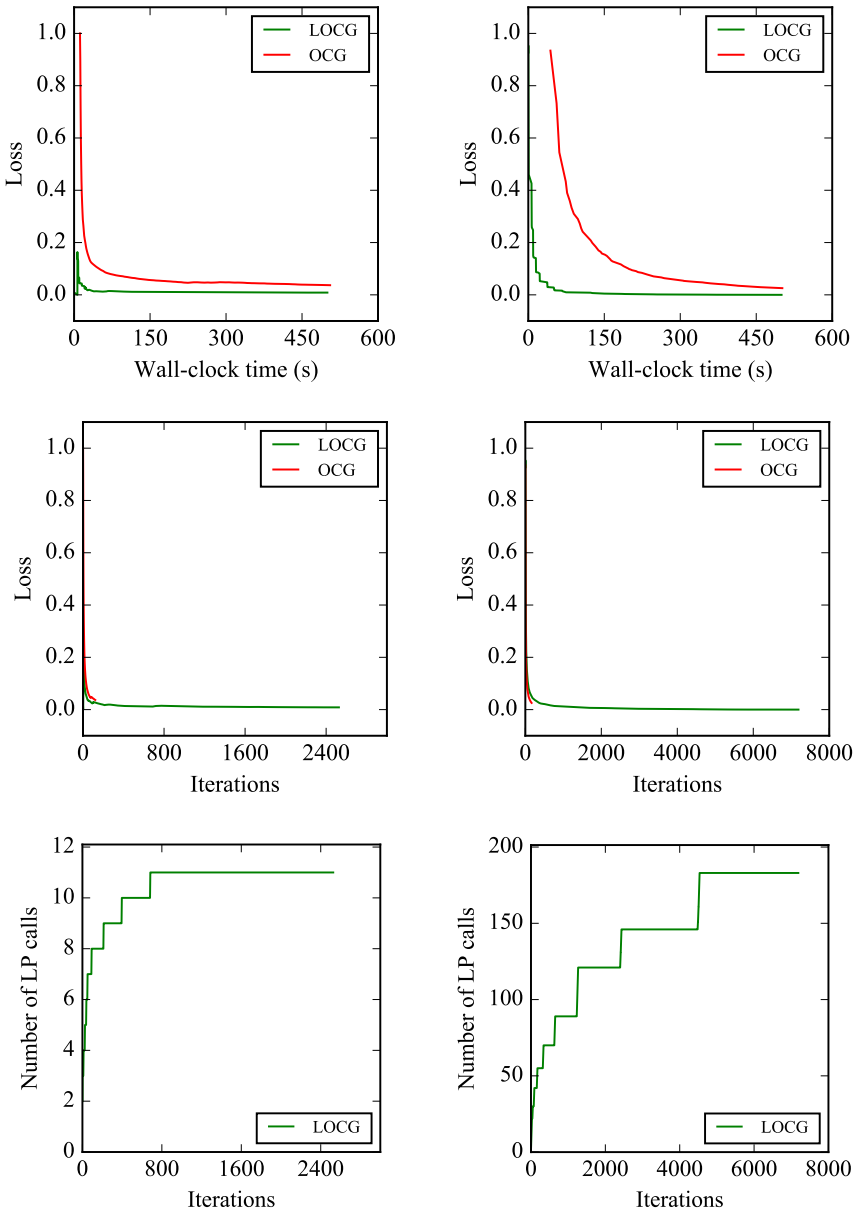

Figure 4.4: LOCG vs. OCG on the cut polytope for a graph with 23 nodes. Both LOCG and OCG converge to the optimum in a few iterations for linear losses, while LOCG is remarkably faster for quadratic losses. It demonstrates that the advantage of lazy algorithms strongly correlates with the difficulty of linear optimization. For linear losses, remarkably LOCG needed no oracle calls after one third of the time.

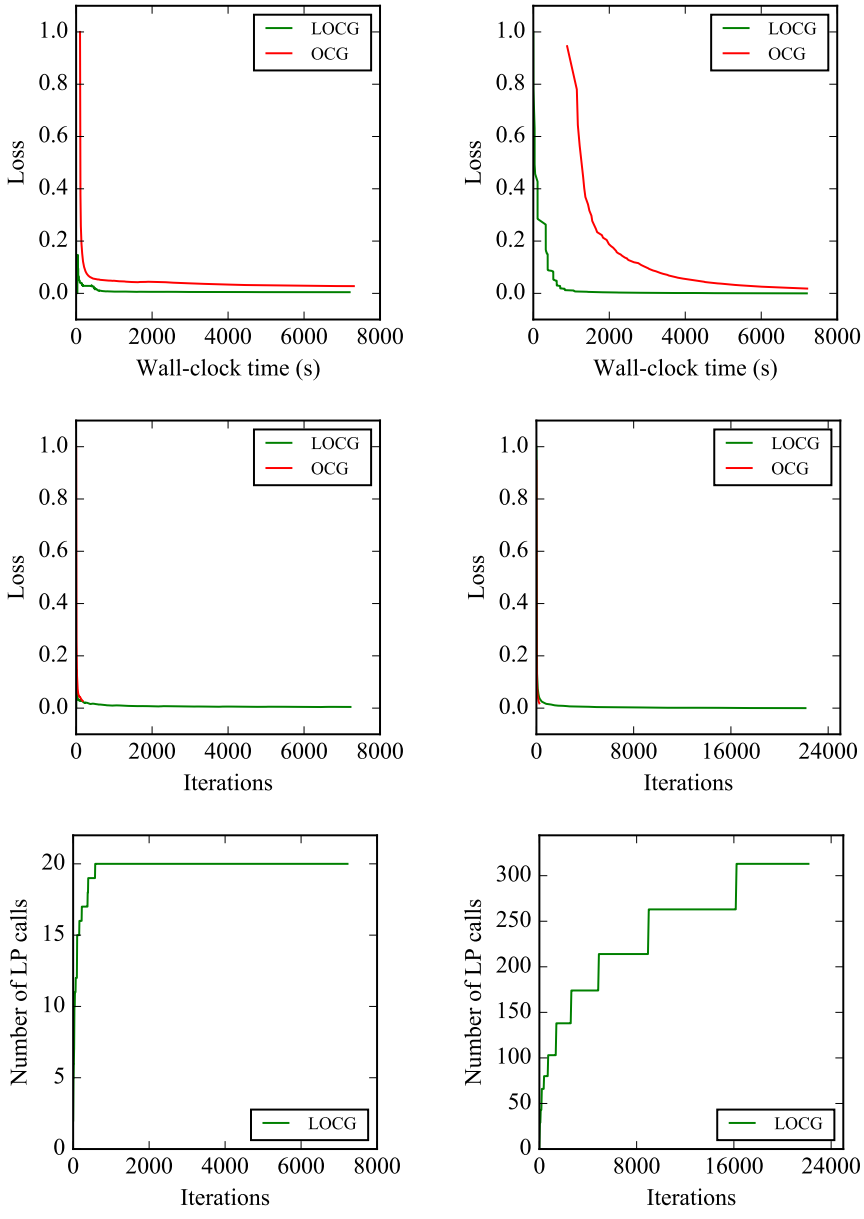

Figure 4.5: LOCG vs. OCG on the cut polytope for a 28-node graph. As for the smaller problem, this also illustrates the advantage of lazy algorithms when linear optimization is expensive. Again, LOCG needed no oracle calls after a small initial amount of time.

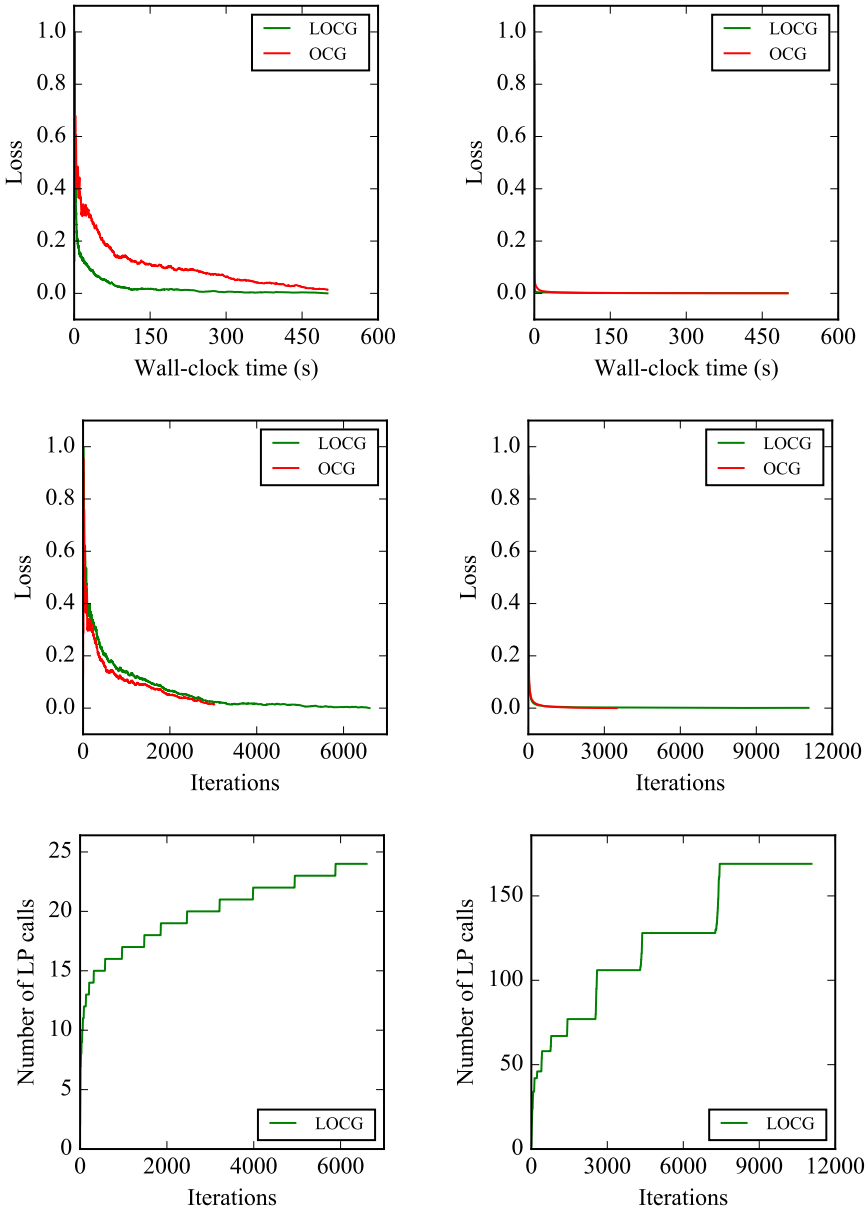

Figure 4.6: LOCG vs. OCG on a small QUBO instance. For quadratic losses, both algorithms converged very fast while LOCG still has a significant edge. For linear losses, LOCG is noticeably faster than OCG.

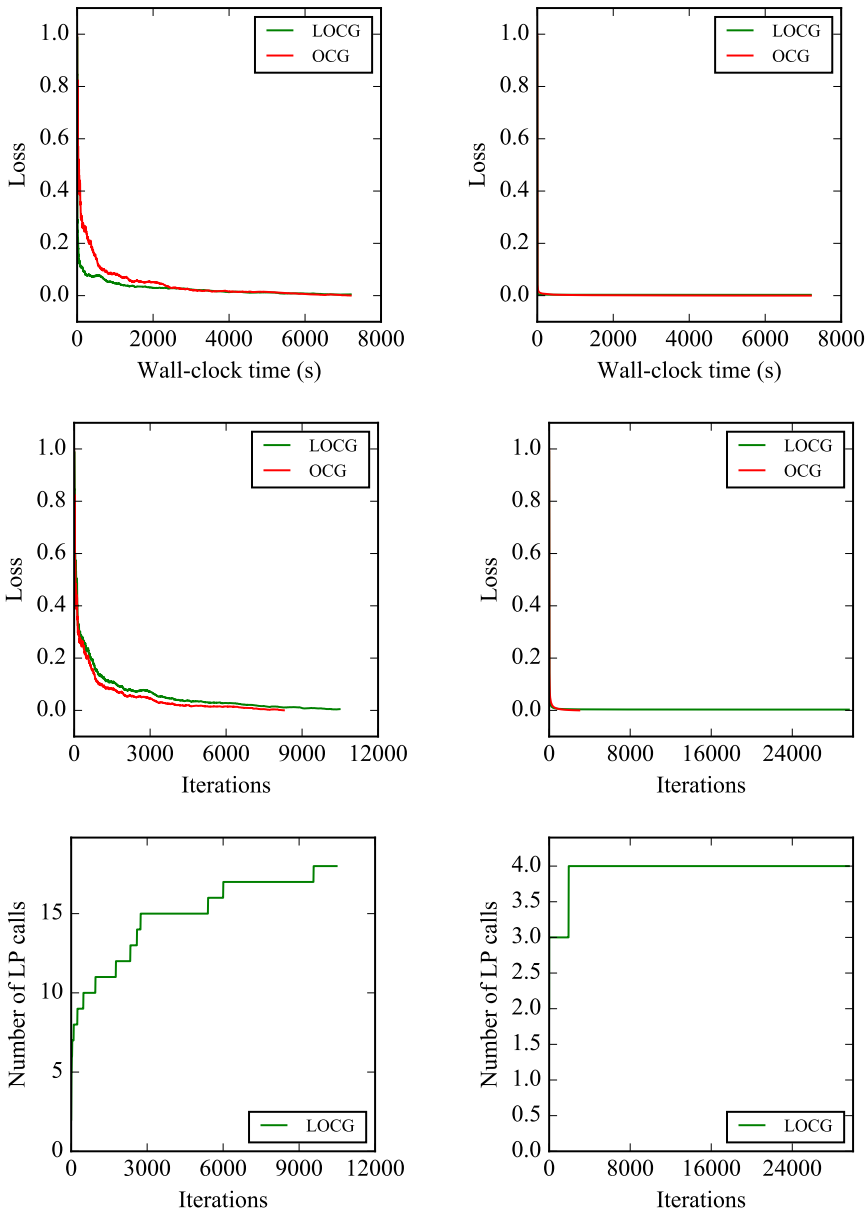

Figure 4.7: LOCG vs. OCG on a large QUBO instance. Both algorithms converge fast to the optimum. Interestingly, LOCG only performs 4 LP calls.

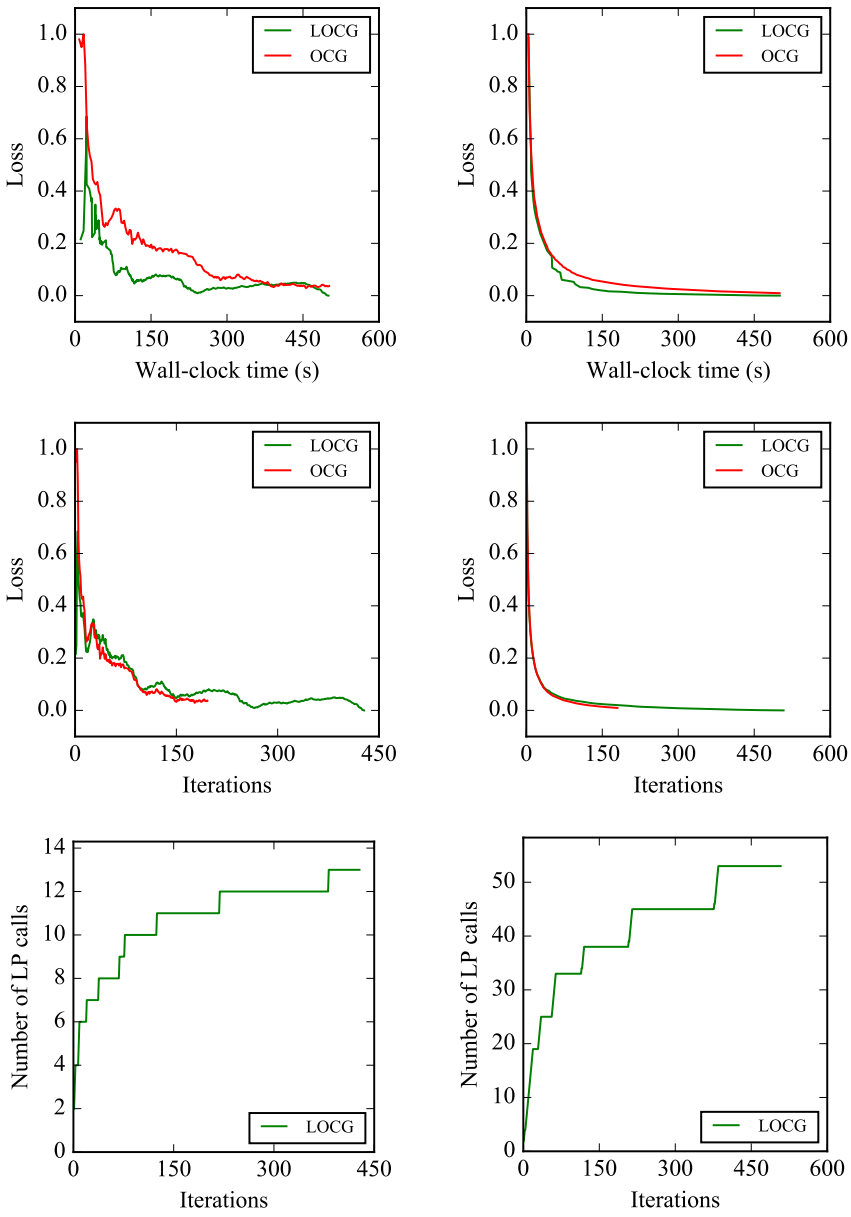

Figure 4.8: LOCG vs. OCG on a path polytope. Similar convergence rate in the number of iterations, but significant difference in terms of wall-clock time.

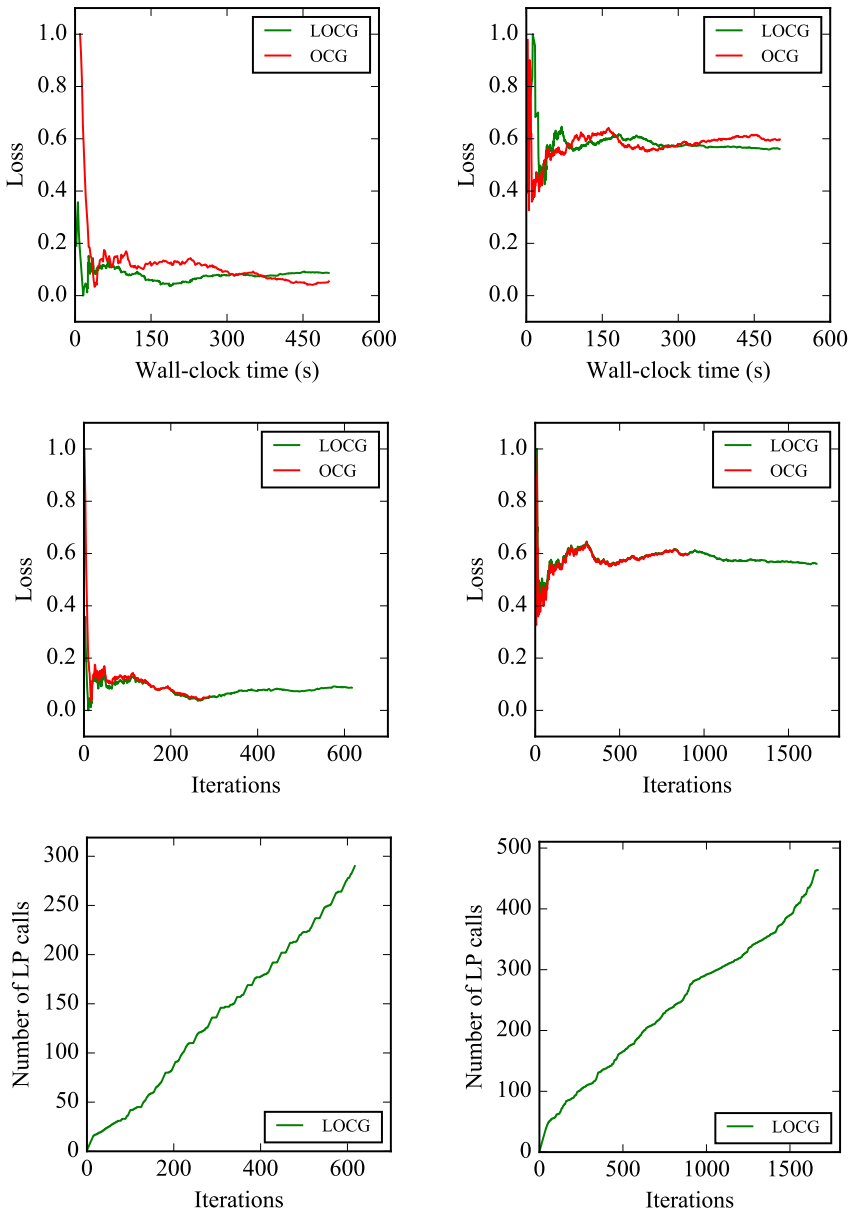

Figure 4.9: LOCG vs. OCG on the MIPLIB instance `eil33-2`. All algorithms performed comparably, due to fast convergence in this case.

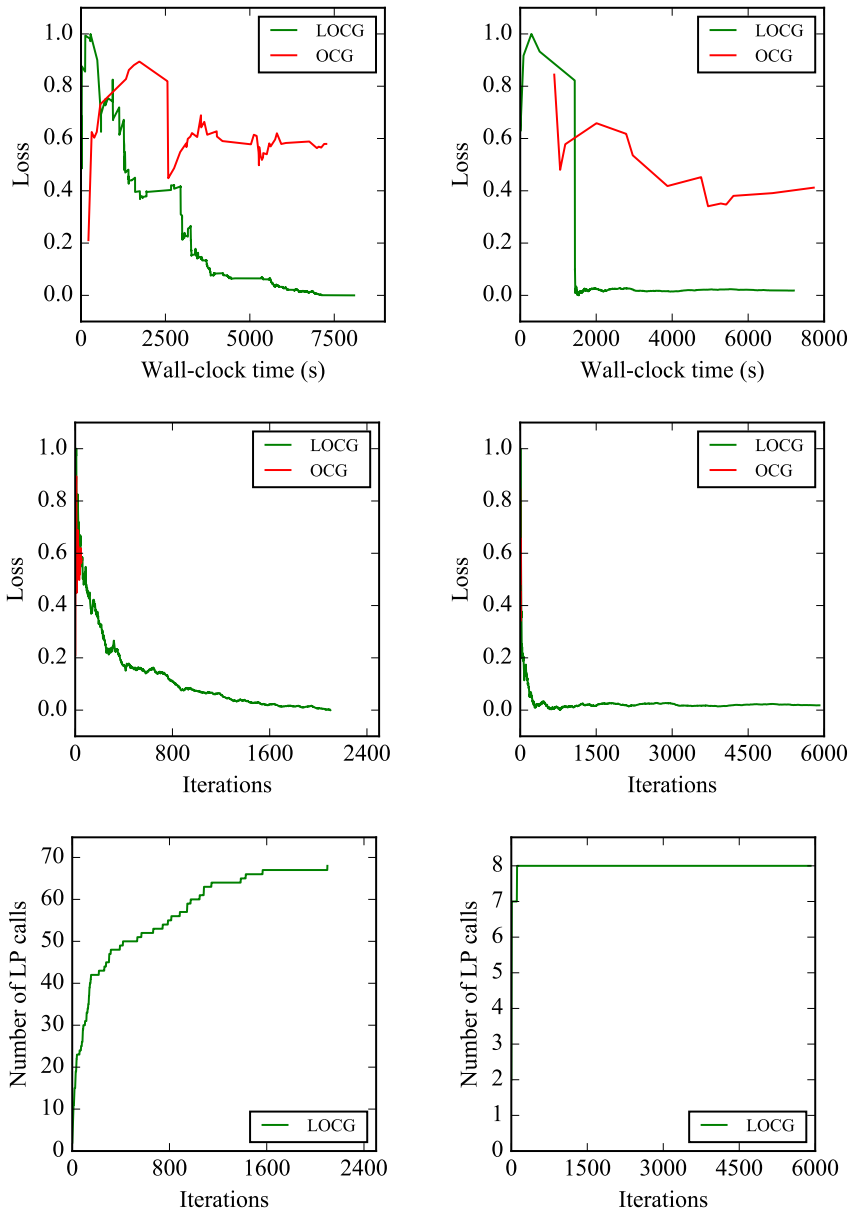

Figure 4.10: LOCG vs. OCG on the MIPLIB instance `air04`. LOCG clearly outperforms OCG as the provided time was not enough for OCG to complete the necessary number of iterations for entering reasonable convergence.

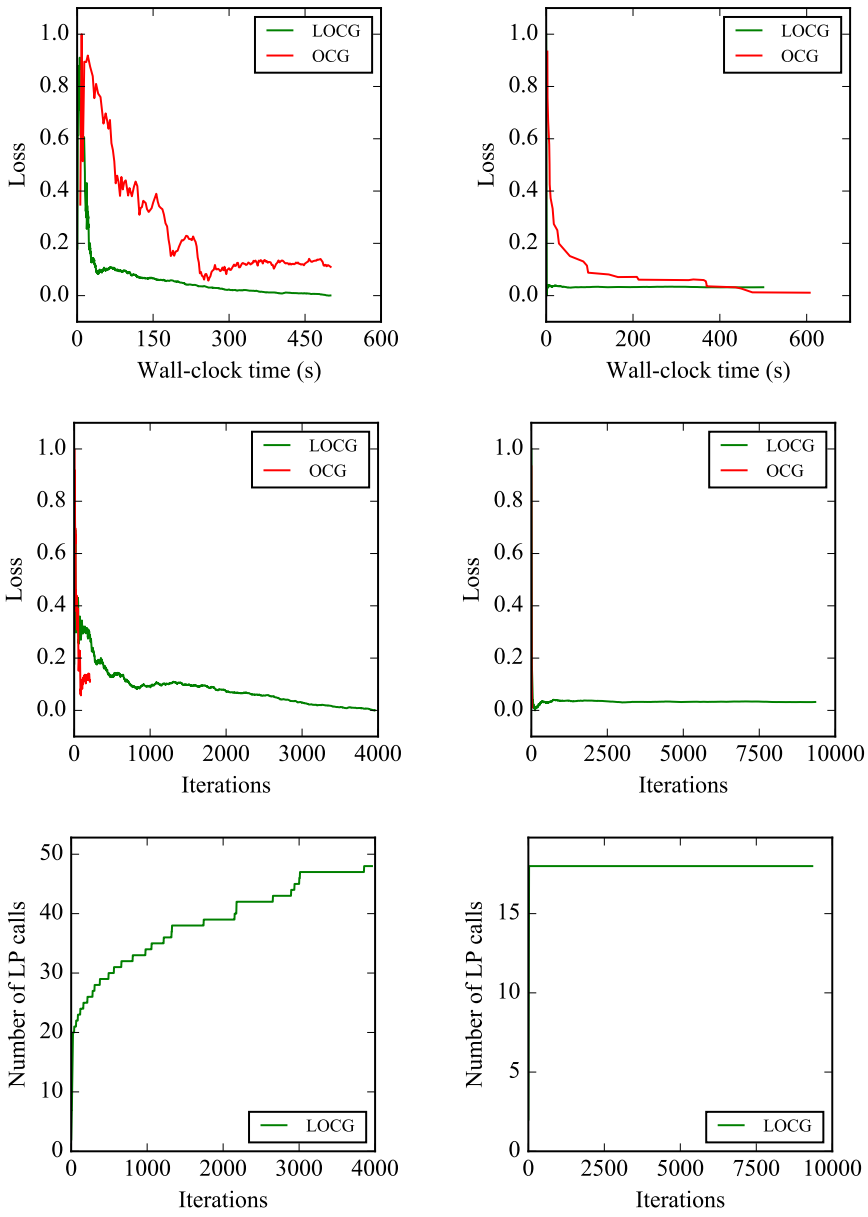

Figure 4.11: LOCG vs. OCG on a spanning tree instance for a 10-node graph. LOCG makes significantly more iterations, few oracle calls, and converges faster in wall-clock time.

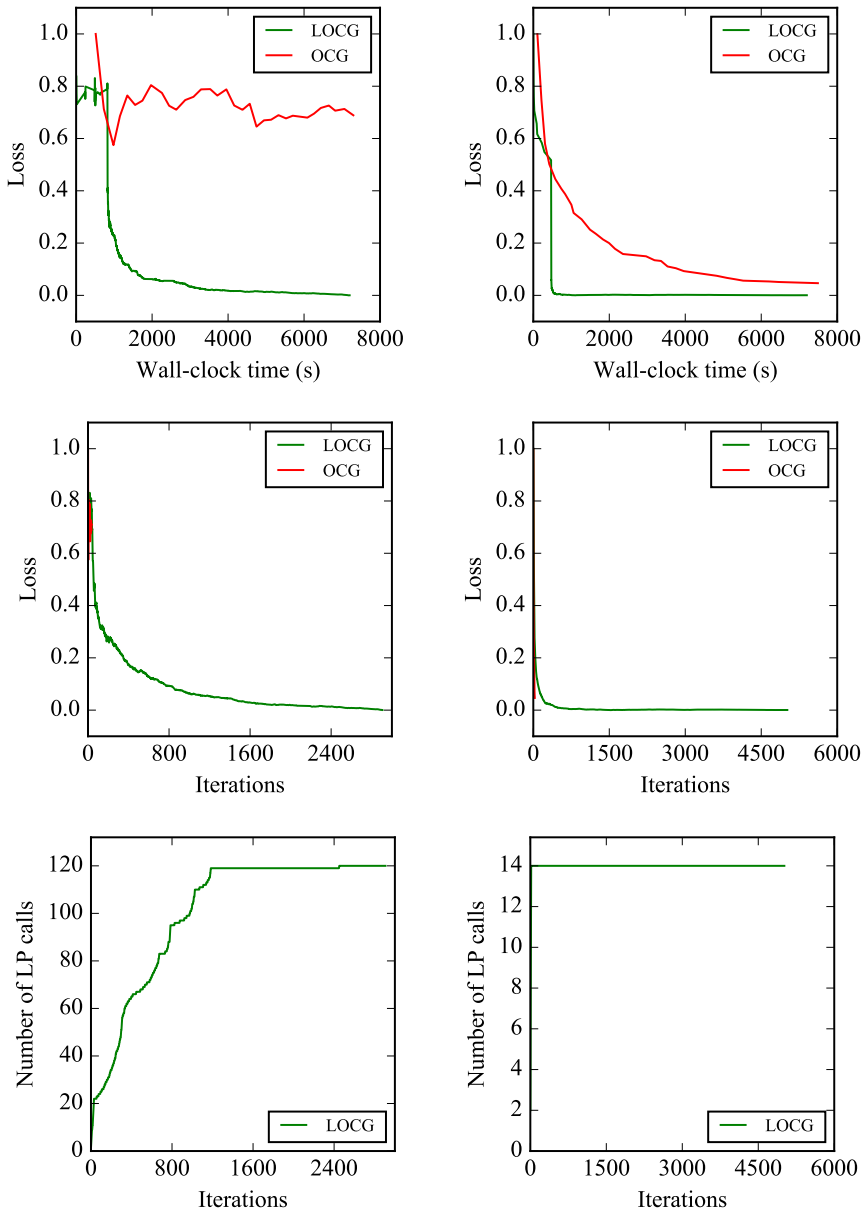

Figure 4.12: LOCG vs. OCG on a spanning tree instance for a 25-node graph. On the left, early fluctuation can be observed, bearing no consequence for later convergence rate. OCG did not get past this early stage. In both cases LOCG converges significantly faster.

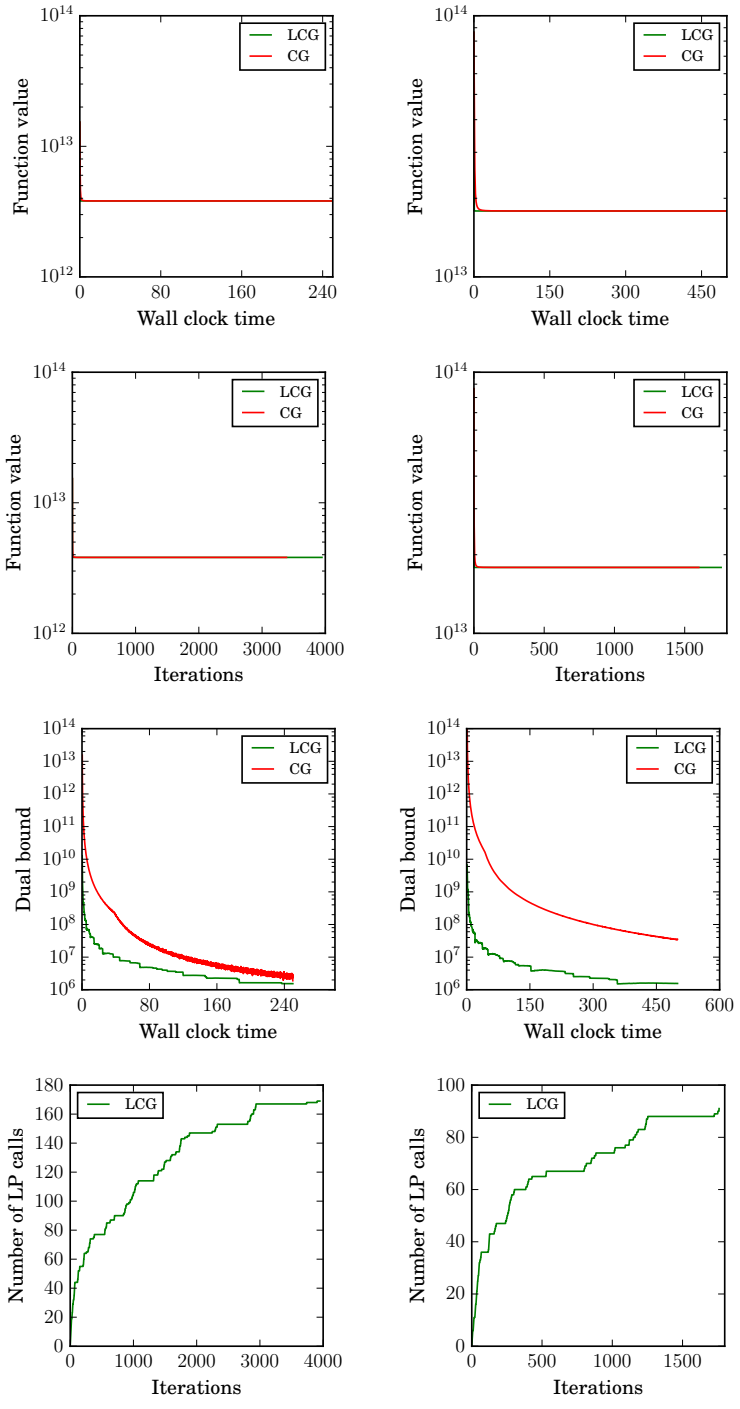

Figure 4.13: LCG vs. CG on small netgen instances `netgen 08a` (left) and `netgen 10a` (right) with quadratic objective functions. In both cases both algorithms are able to reduce the function value very fast, however the dual bound or Wolfe gap is reduced much faster by LCG. Observe that the vertical axis is given with a logscale.

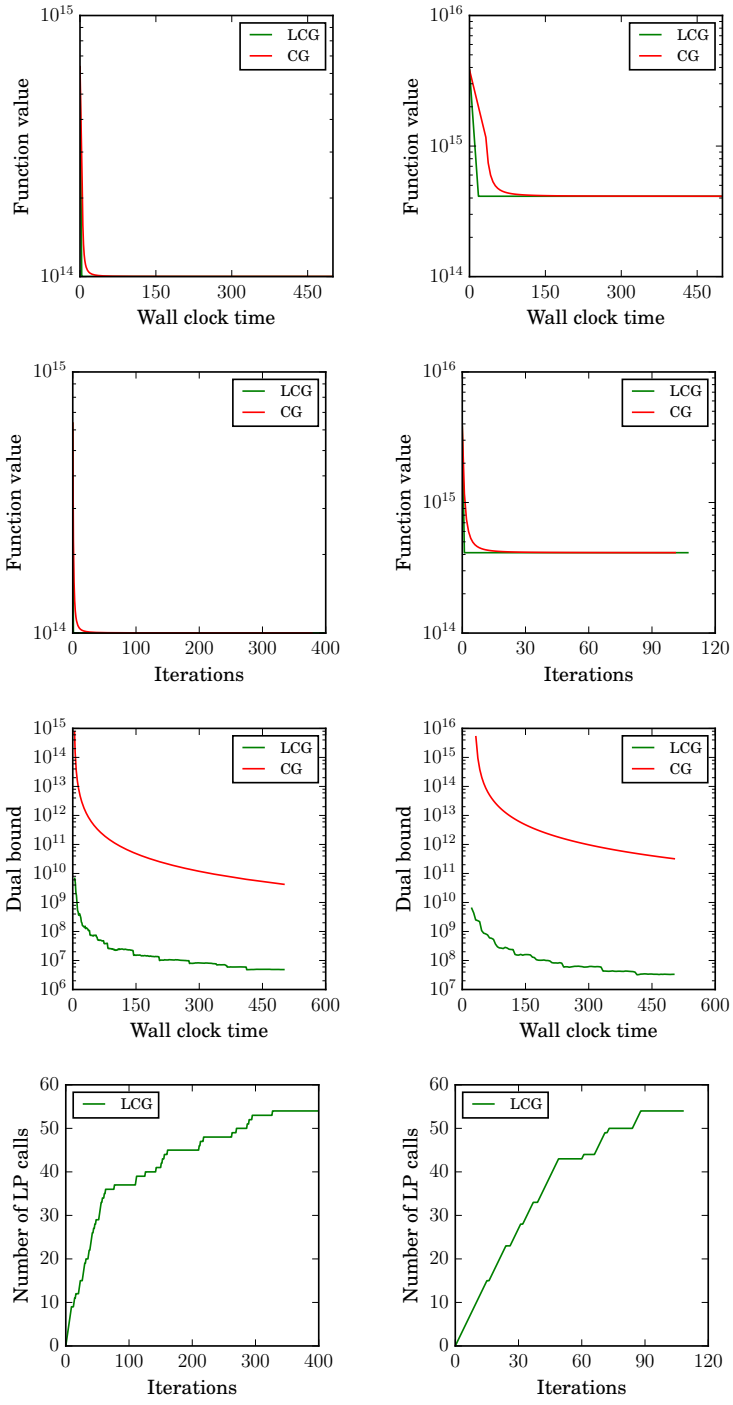

Figure 4.14: LCG vs. CG on medium sized netgen instances `netgen 12b` (left) and `netgen 14a` (right) with quadratic objective functions. The behavior of both versions on these instances is very similar to the small netgen instances (Figure 4.13), however both in the function value and the dual bound the difference between the lazy and the non-lazy version is more prominent. Again, we used a logscale for the vertical axis.

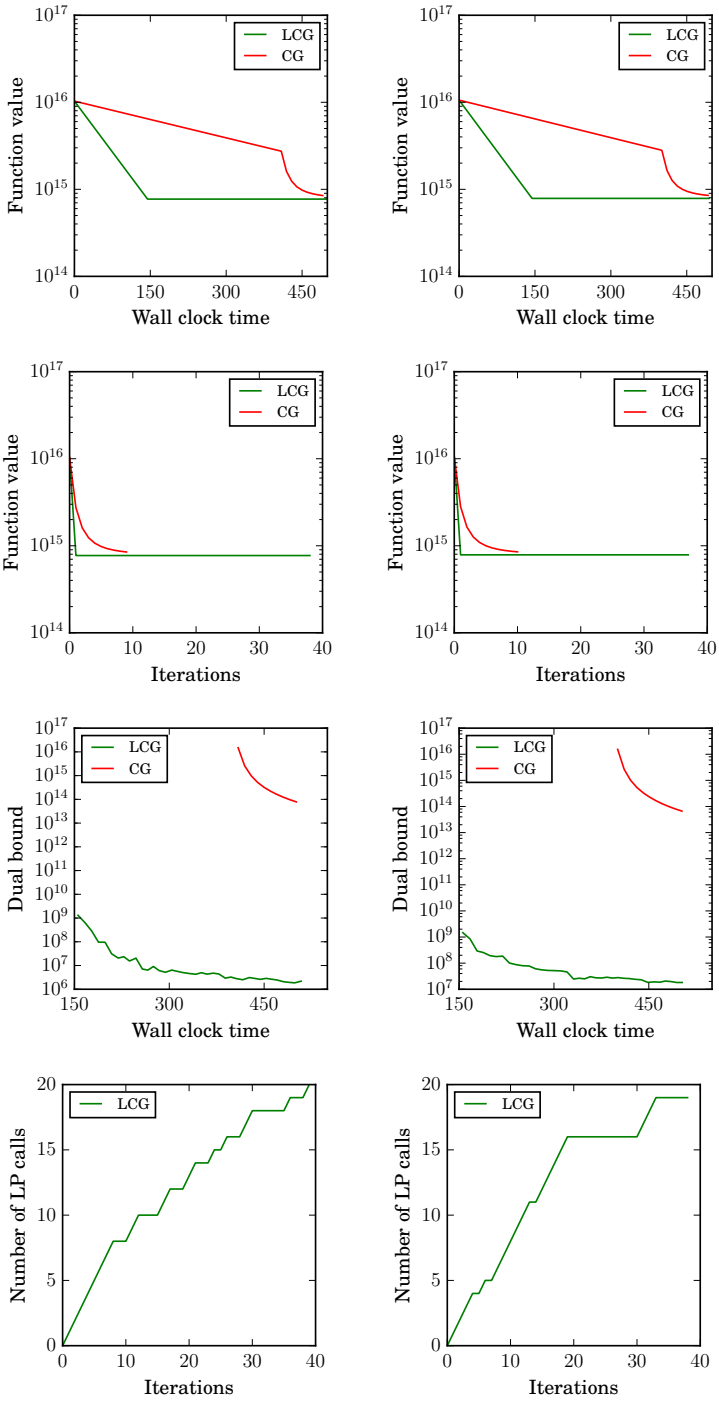

Figure 4.15: LCG vs. CG on large netgen instances `netgen 16a` (left) and `netgen 16b` (right) with quadratic objective functions. In both cases the difference in function value between the two versions of the algorithm is large. In the dual bound the performance of the lazy version is multiple orders of magnitude better than the performance of the non-lazy counterpart. The cache hit rates for these two instances are lower due to the high dimension of the polytope.

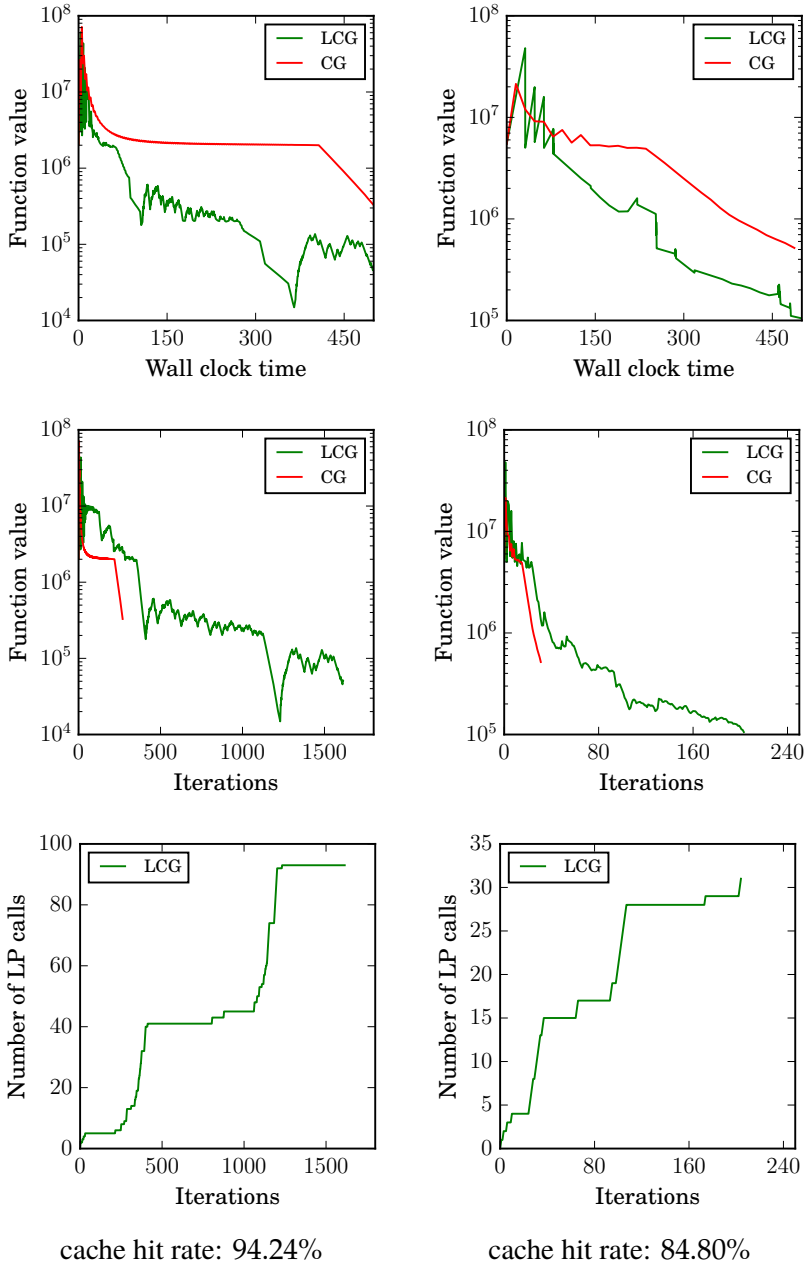

Figure 4.16: LCG vs. CG on two matrix completion instances. We solve the problem as given in Equation (4.26) with the parameters $n = 3000$, $m = 1000$, $r = 10$ and $R = 30000$ for the left instance and $n = 10000$, $m = 100$, $r = 10$ and $R = 10000$ for the right instance. In both cases the lazy version is slower in iterations, however significantly faster in wall clock time.

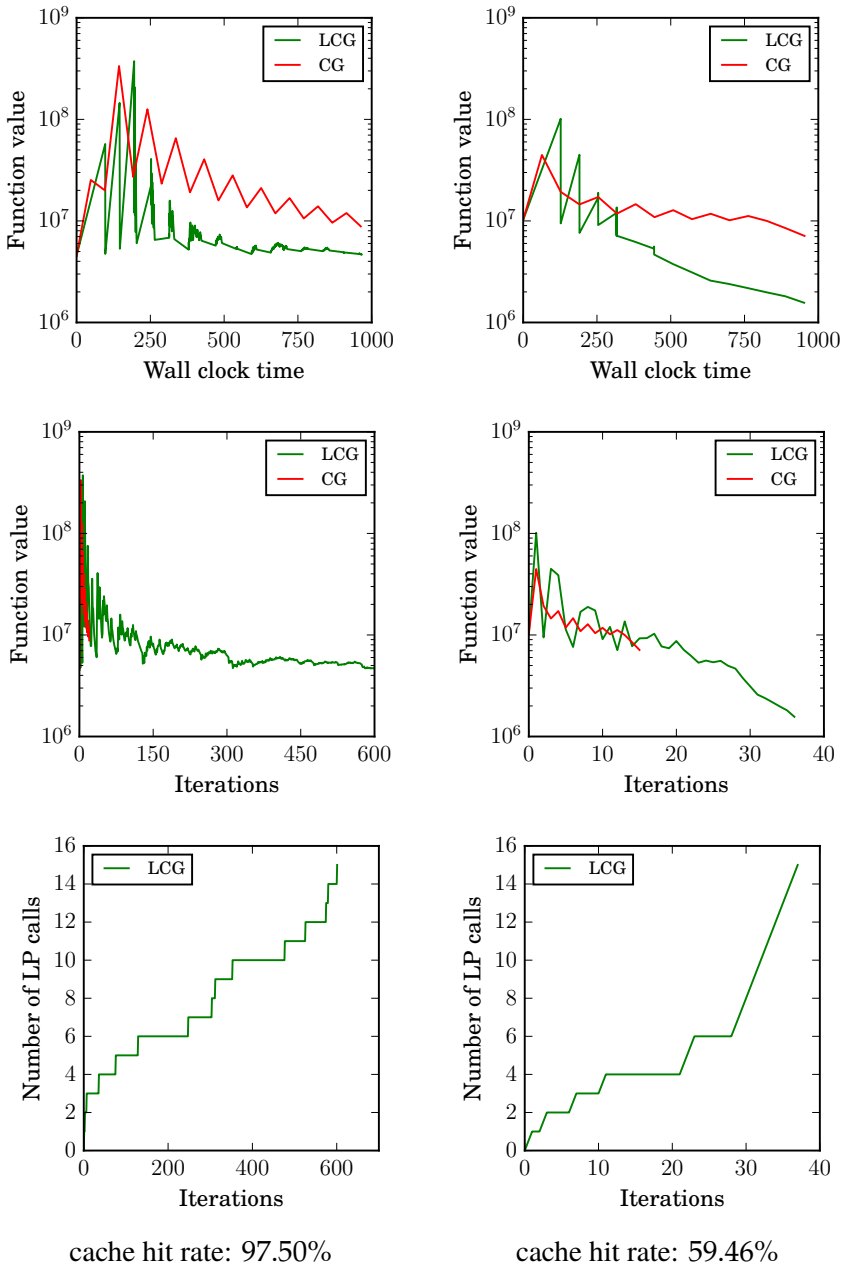

Figure 4.17: LCG vs. CG on two more matrix completion instances. The parameters for Equation (4.26) are given by $n = 5000$, $m = 4000$, $r = 10$ and $R = 50000$ for the left instance and $n = 100$, $m = 20000$, $r = 10$ and $R = 15000$ for the right instance. In both of these cases the performance of the lazy and the non-lazy version are comparable in iterations, however in wall clock time the lazy version reaches lower function values faster.

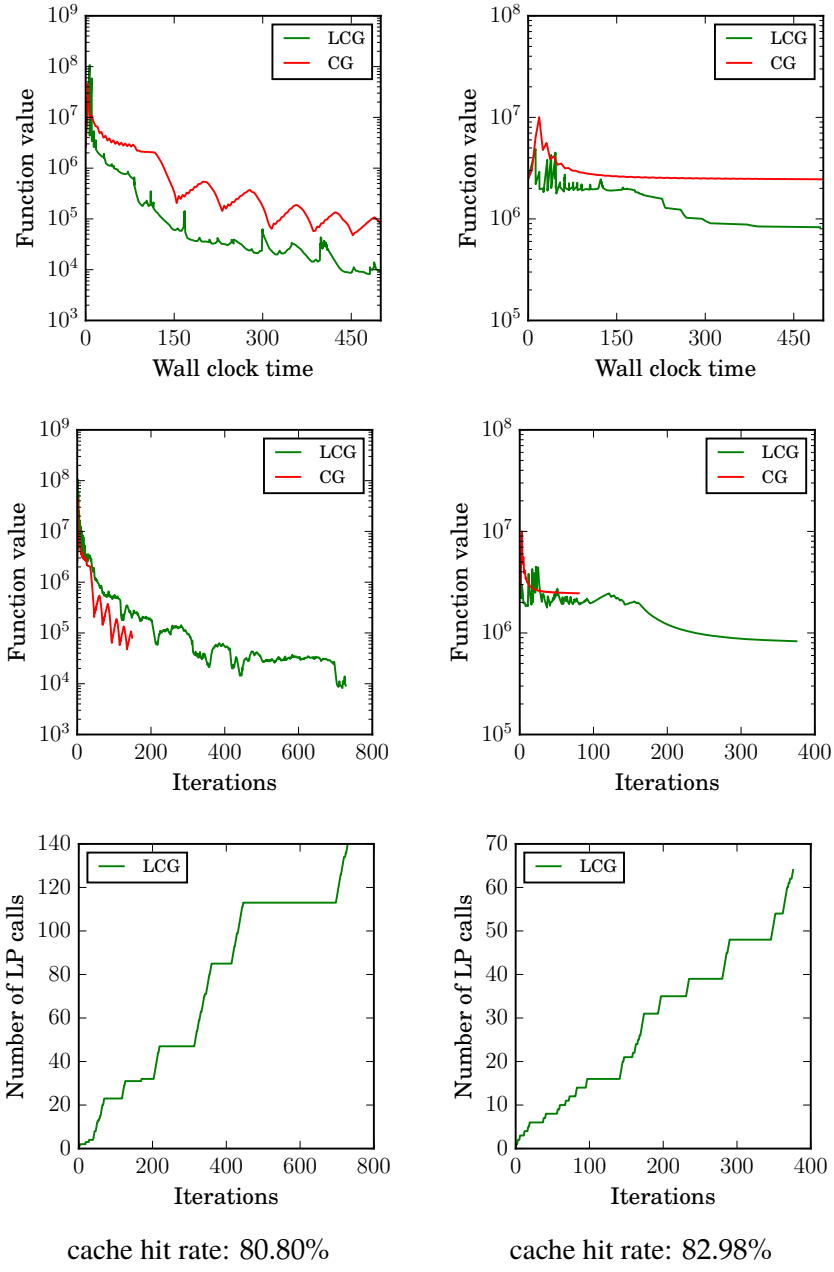

Figure 4.18: LCG vs. CG on our fifth and sixth instances of the matrix completion problem. The parameters are $n = 5000$, $m = 100$, $r = 10$ and $R = 15000$ for the left instance and $n = 3000$, $m = 2000$, $r = 10$ and $R = 10000$ for the right instance. The behavior is very similar to Figure 4.17. similar performance over iterations however advantages for the lazy version over wall clock time.

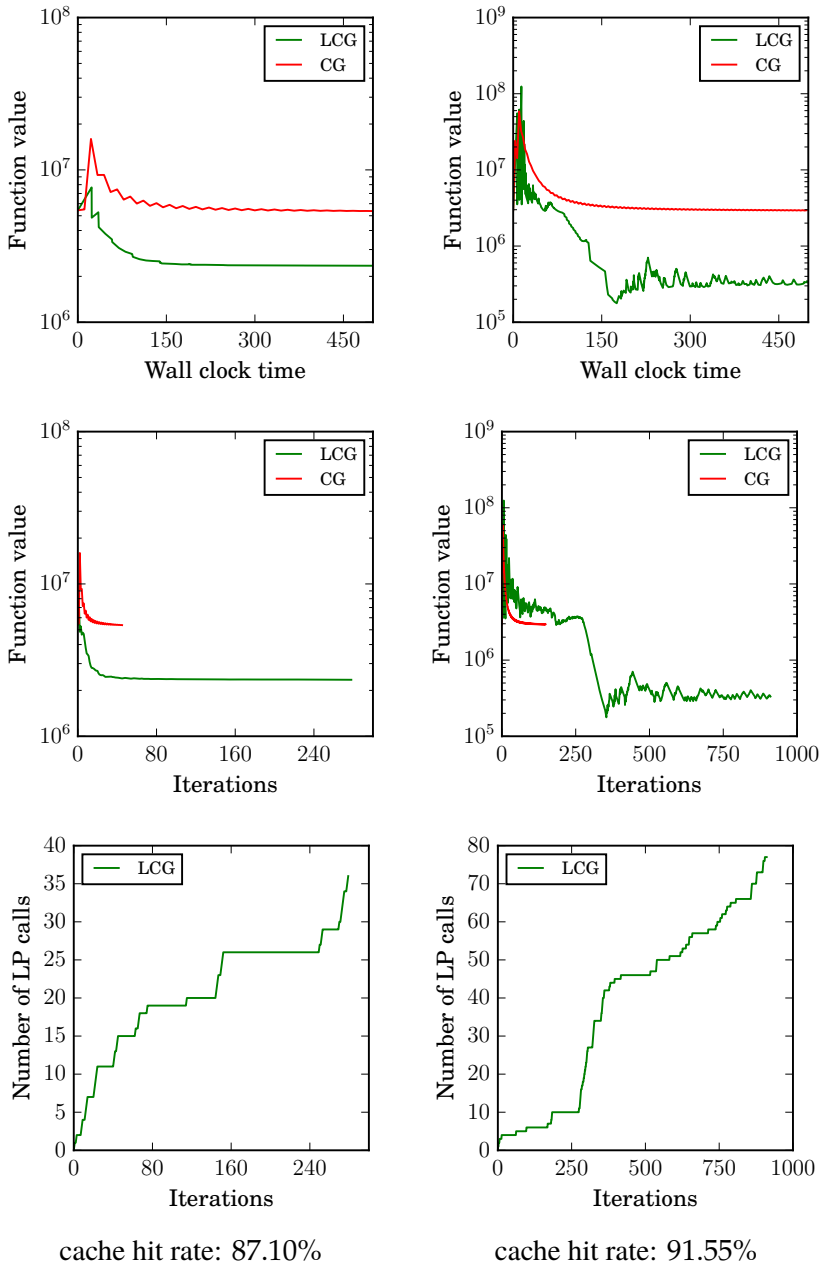

Figure 4.19: LCG vs. CG on the final two matrix completion instances. The parameters are $n = 10000$, $m = 1000$, $r = 10$ and $R = 1000$ for the left instance and $n = 5000$, $m = 1000$, $r = 10$ and $R = 30000$ for the right instance. On the left in both measures, instances and wall clock time, the lazy version performs better than the non-lazy counterpart, due to a suboptimal direction at the beginning with a fairly large step size in the non-lazy version.

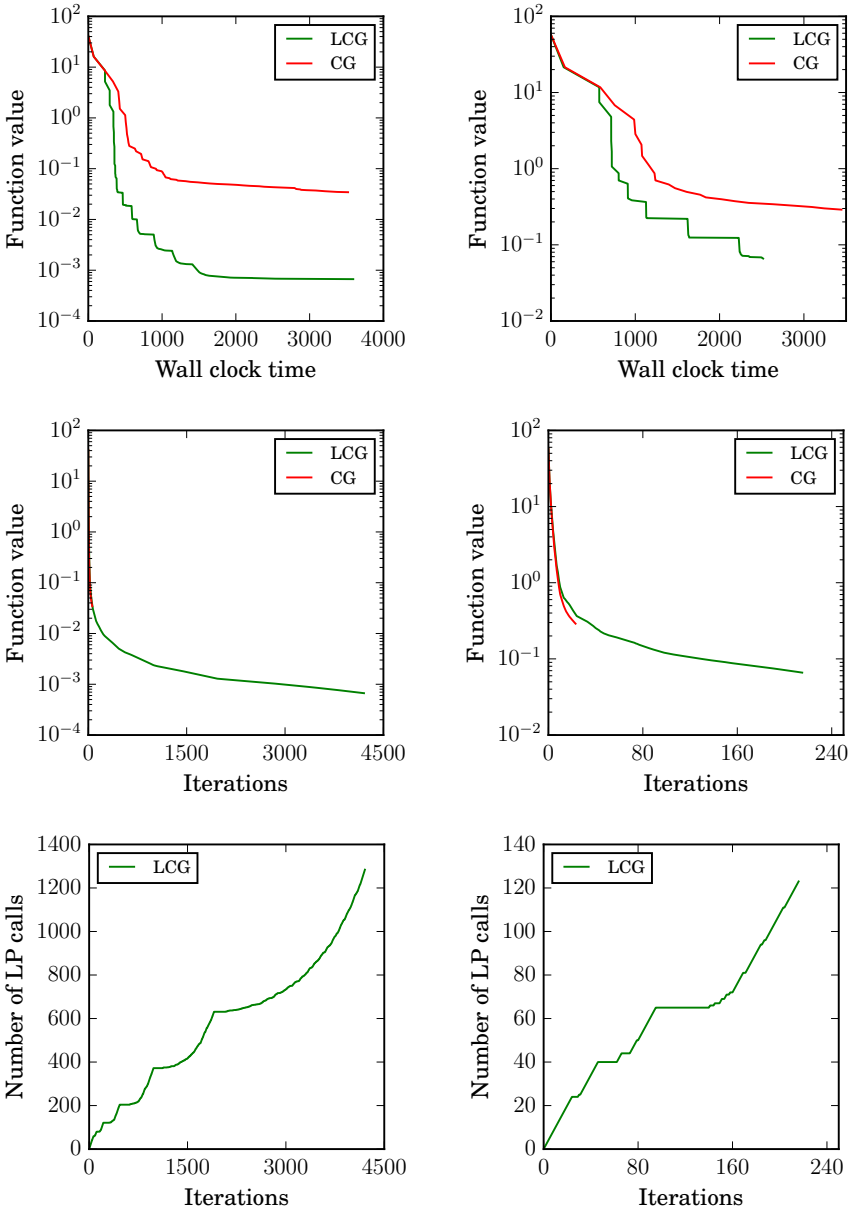

Figure 4.20: LCG vs. CG on structured regression problems with feasible regions being a TSP polytope over 11 nodes (left) and 12 nodes (right). In both cases LCG is significantly faster in wall-clock time.

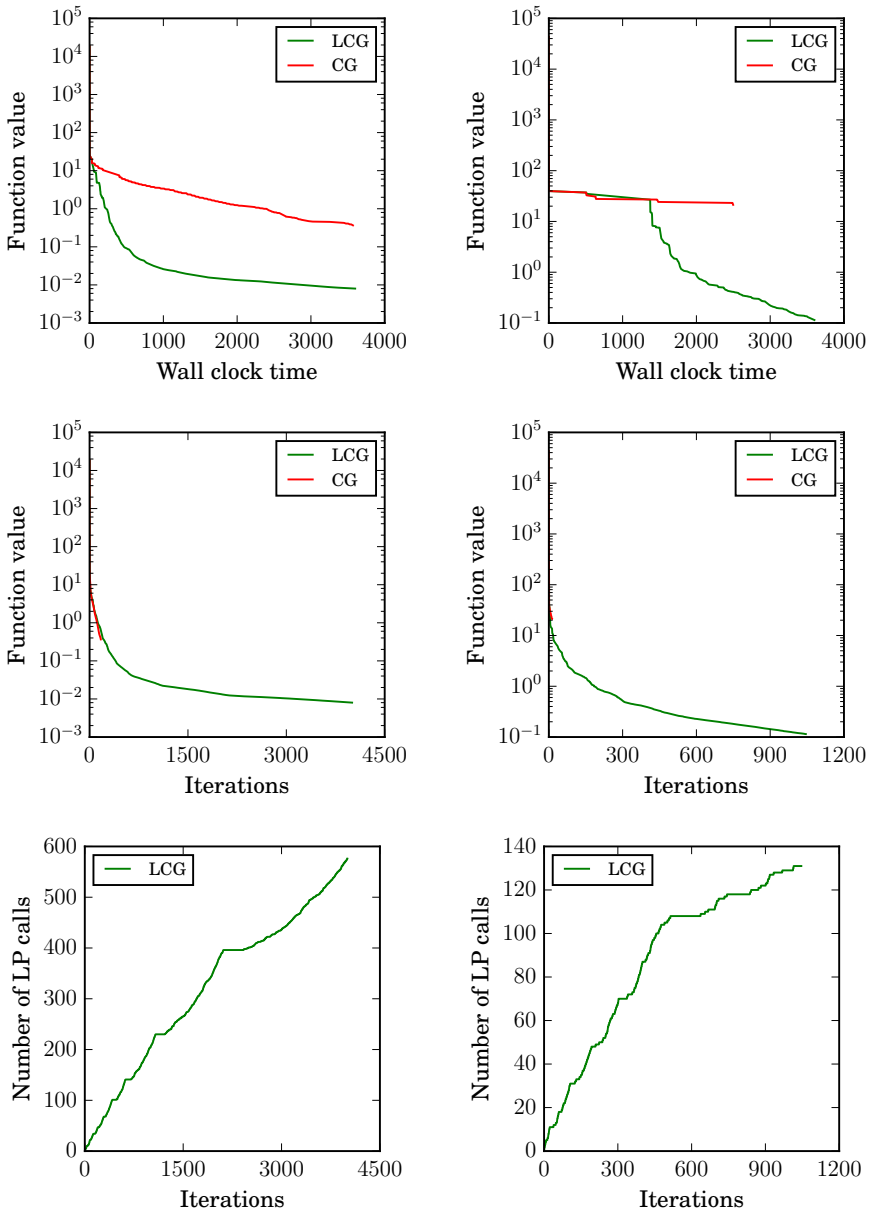

Figure 4.21: LCG vs. CG on structured regression instances using cut polytopes over a graph on 23 nodes (left) and over 28 nodes (right) as feasible region. In both instances LCG performs significantly better than CG.

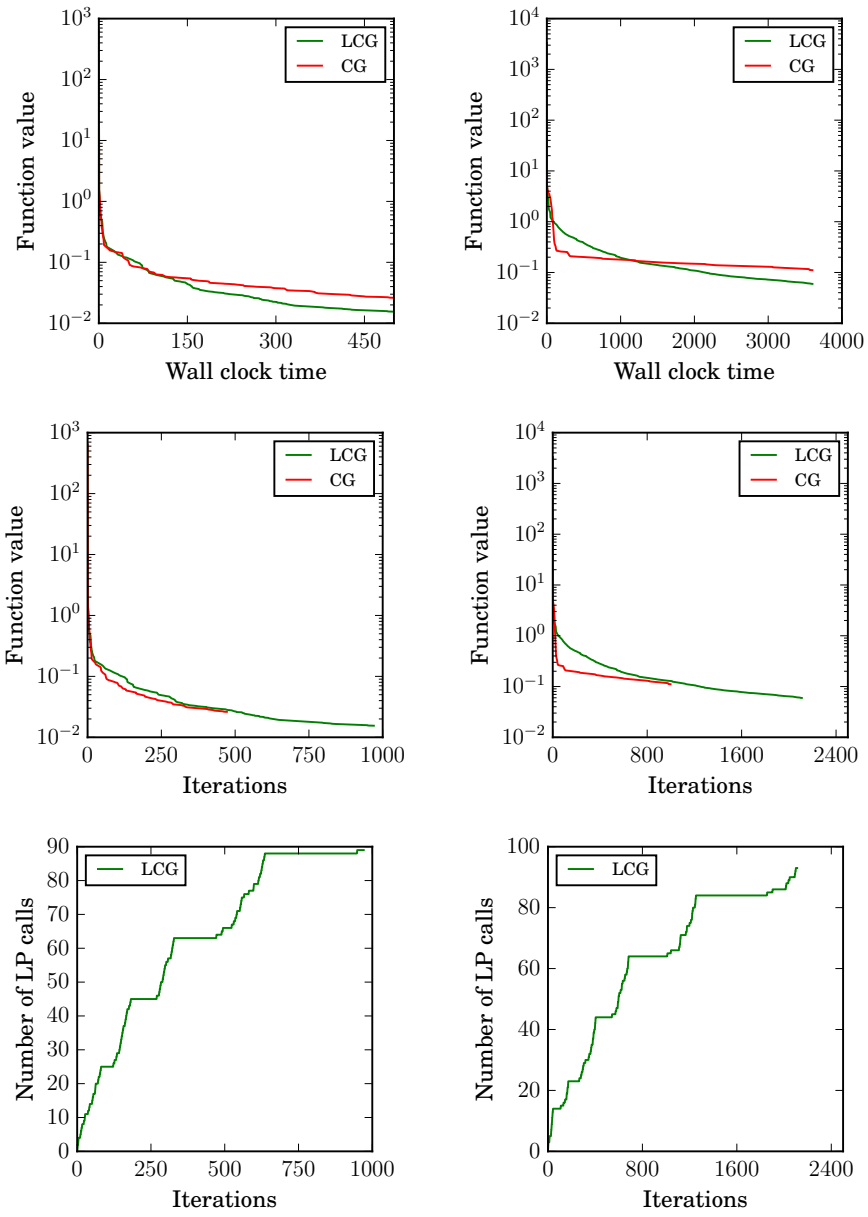

Figure 4.22: LCG vs. CG on structured regression instances with extended formulation of the spanning tree problem on a 10 node graph on the left and a 15 node graph on the right.

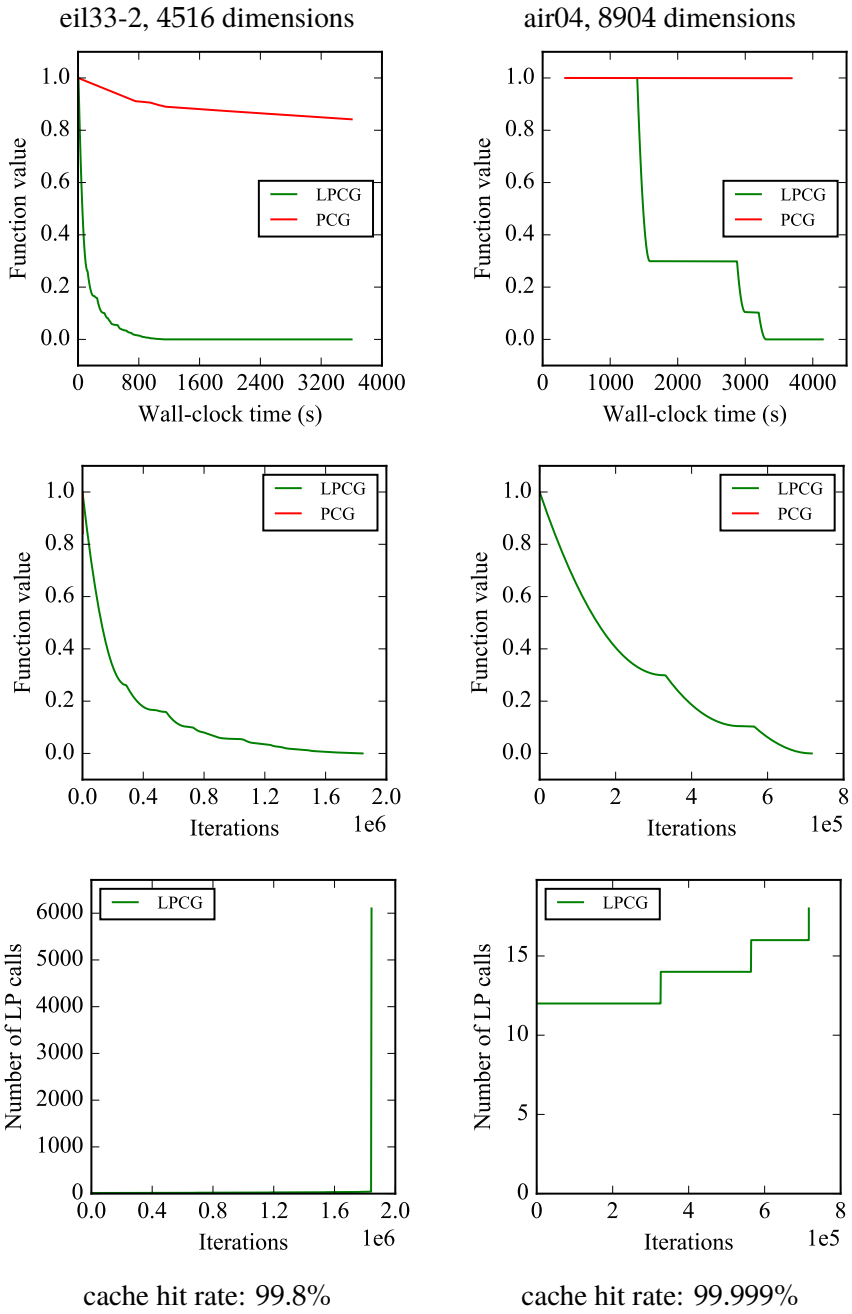

Figure 4.23: LPCG vs. PCG on two MIPLIB instances `eil33-2` and `air04`. LPCG converges very fast, making millions of iterations with a relatively few oracle calls, while PCG completed only comparably few iterations due to the time-consuming oracle calls. This clearly illustrates the advantage of lazy methods when the cost of linear optimization is non-negligible. On the left, when reaching ε-optimality, LPCG performs many (negative) oracle calls to (re-)prove optimality; at that point one might opt for stopping the algorithm. On the right LPCG needed a rather long time for the initial bound tightening of Φ_0, before converging significantly faster than PCG.

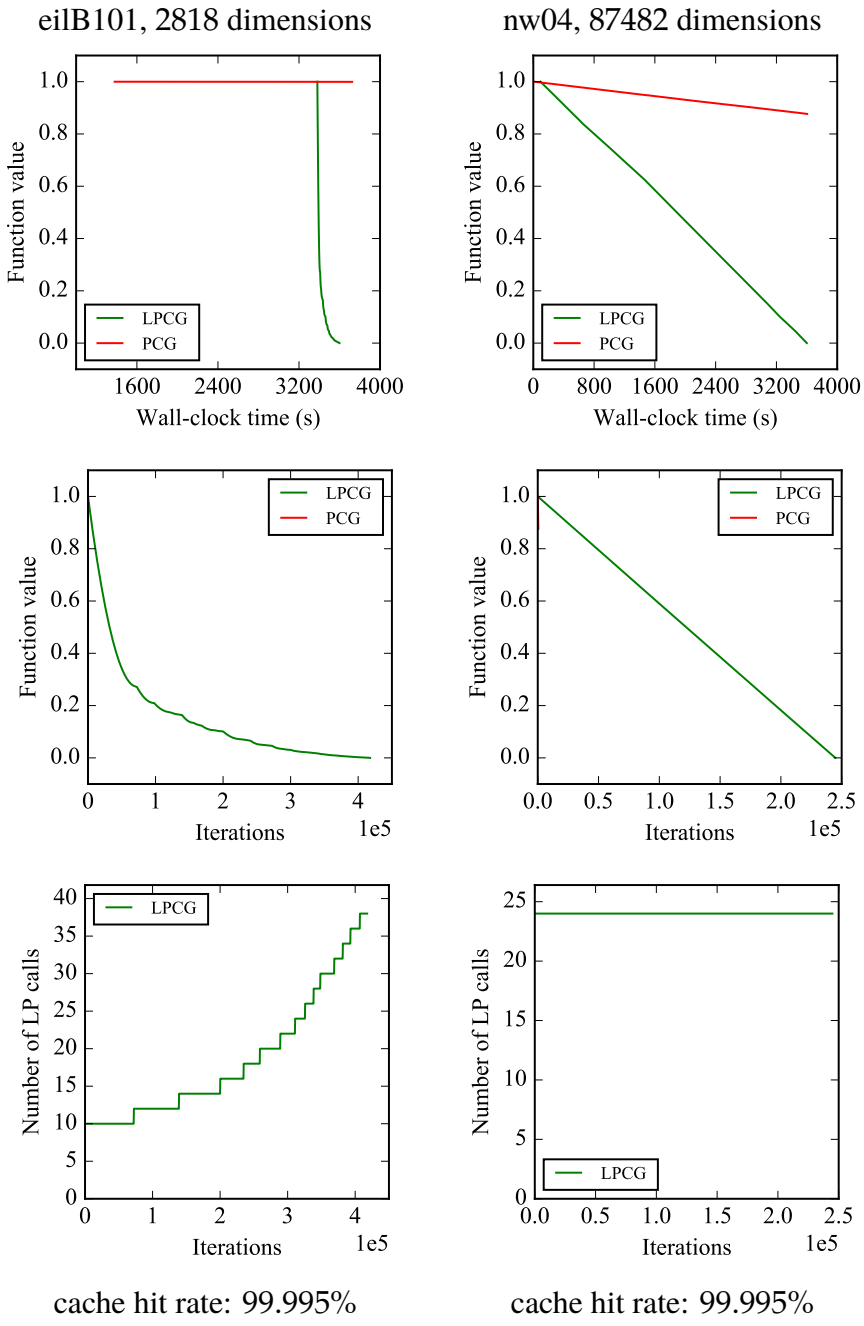

Figure 4.24: LPCG vs. PCG on MIPLIB instances `eilB101` and `nw04` with quadratic loss functions. For the `eilB101` instance, LPCG spent most of the time tightening Φ_0, after which it converged very fast, while PCG was unable to complete a single iteration even solving the problem only approximately. For the `nw04` instance LPCG needed no more oracle calls after an initial phase, while significantly outperforming PCG.

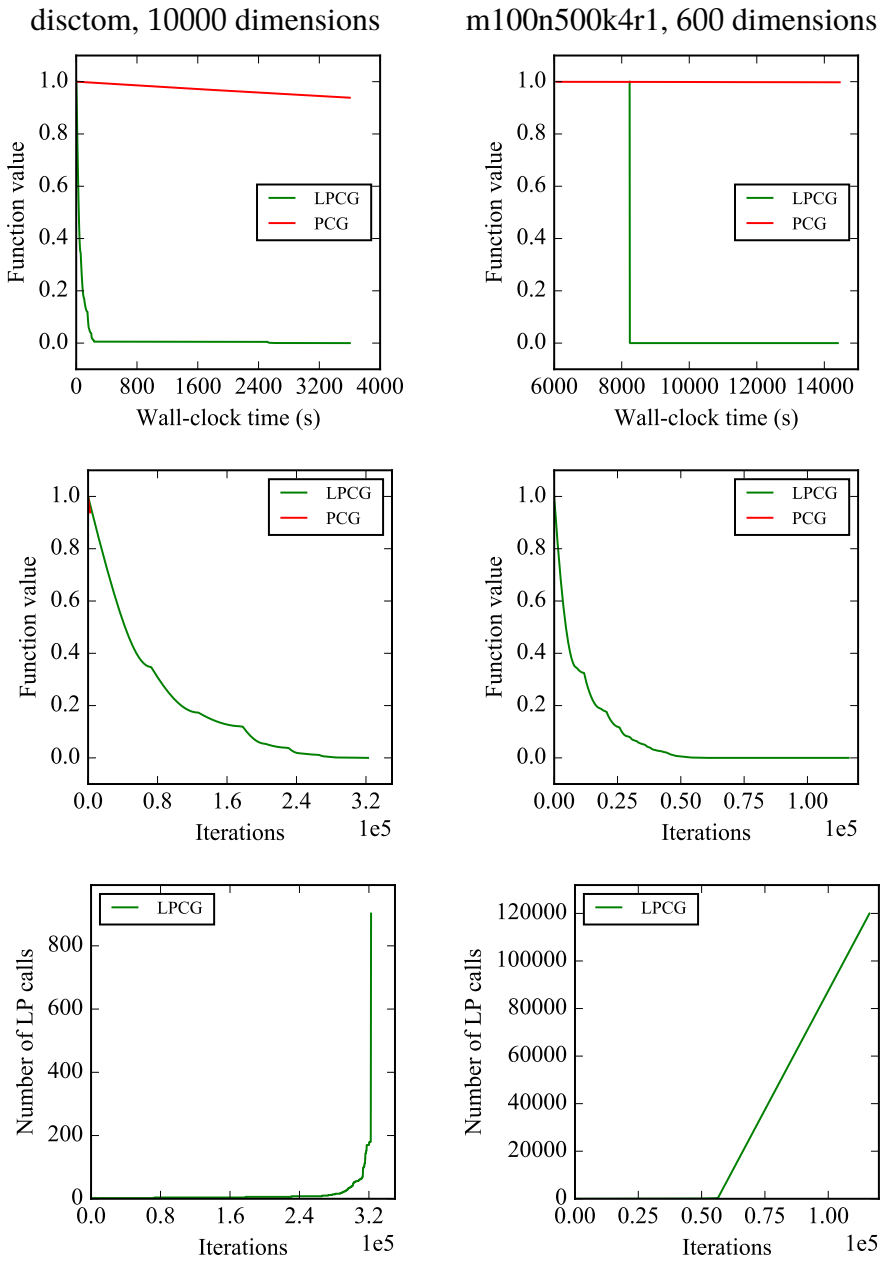

Figure 4.25: LPCG vs. PCG on MIPLIB instances `disctom` and `m100n500k4r1`. After very fast convergence, there is a huge increase in the number of oracle calls for the lazy algorithm LPCG due to reaching ε-optimality as explained before. On the right the initial bound tightening for Φ_0 took a considerable amount of time but then convergence is almost instantaneous.

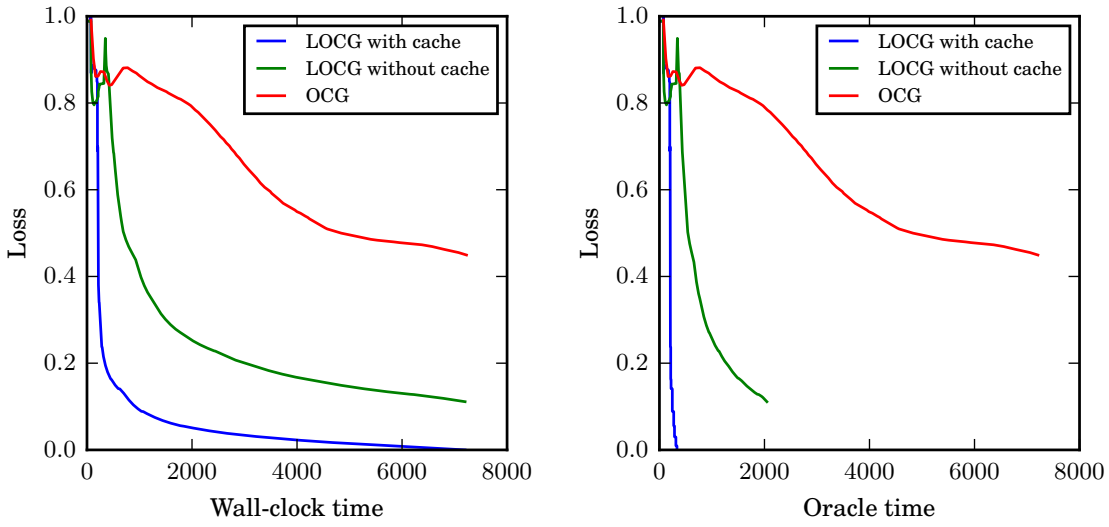

Figure 4.26: Performance gain due to caching and early termination for stochastic optimization over a maximum cut problem with linear losses. The red line is the OCG baseline, the green one is the lazy variant using only early termination, and the blue one uses caching and early termination. Left: loss vs. wall-clock time. Right: loss vs. total time spent in oracle calls. Time limit was 7200 seconds. Caching allows for a significant improvement in loss reduction in wall-clock time. The effect is even more obvious in oracle time as caching cuts out a large number of oracle calls.

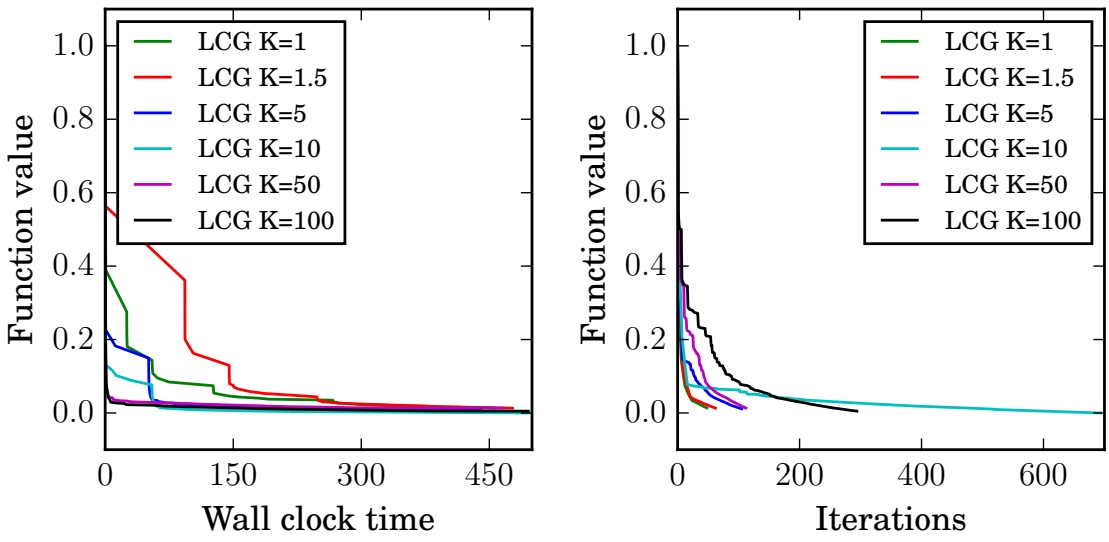

Figure 4.27: Impact of the oracle approximation parameter K depicted for the Lazy CG algorithm. We can see that increasing K leads to a deterioration of progress in iterations but improves performance in wall-clock time. The behavior is similar for other algorithms.

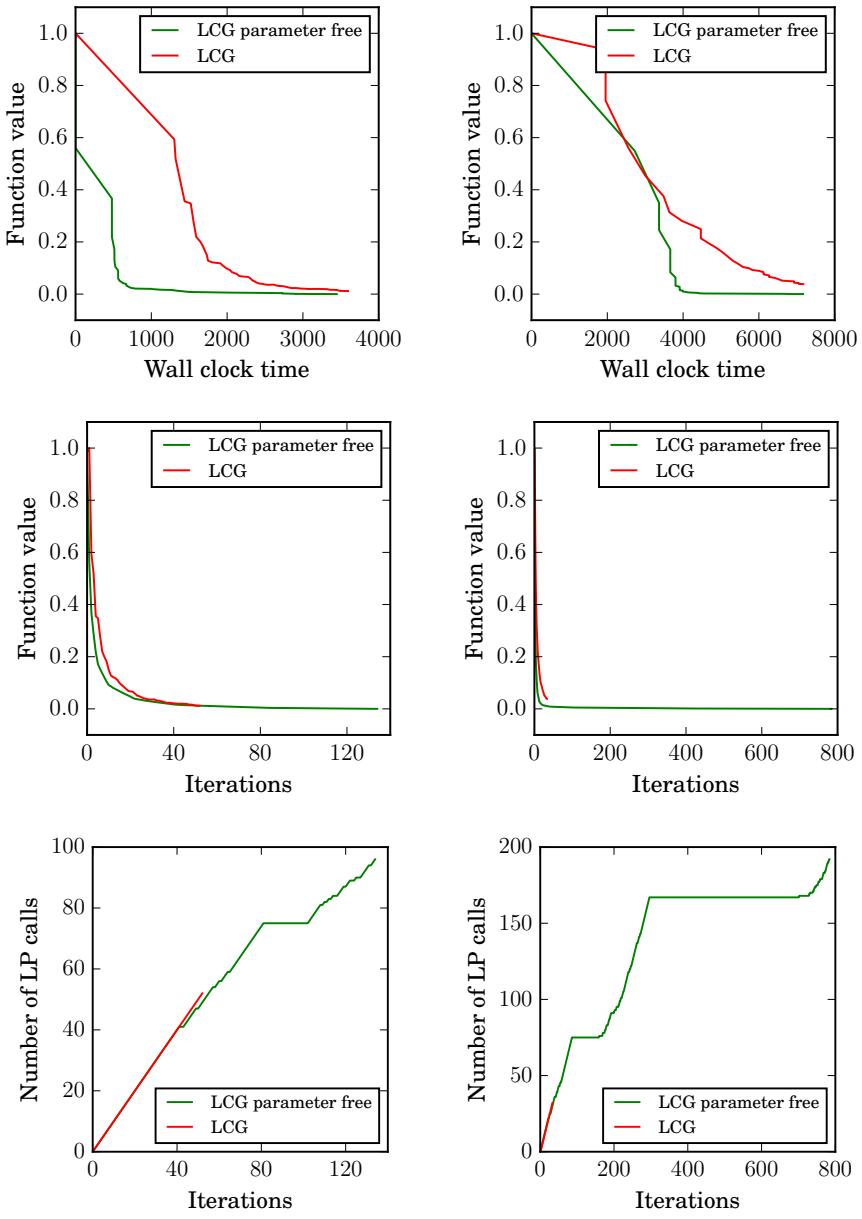

Figure 4.28: Comparison of the 'textbook' variant of the Lazy CG algorithm (Algorithm 2) vs. the Parameter-free Lazy CG (Algorithm 7) depicted for two sample instances to demonstrate behavior. The parameter-free variant usually has a slightly improved behavior in terms of iterations and a significantly improved behavior in terms of wall-clock performance. In particular, the parameter-free variant can execute significantly more oracle calls, due to the Φ-halving strategy and the associated bounded number of negative calls (see Theorem 4.3.3).

REFERENCES

Achterberg, Tobias, Thorsten Koch, and Alexander Martin (2006). "MIPLIB 2003". In: *Operations Research Letters* 34.4, pp. 361–372. DOI: 10.1016/j.orl.2005.07.009. URL: http://www.zib.de/Publications/abstracts/ZR-05-28/.

Agarwal, Amit et al. (2005). "$O(\sqrt{\log n})$ approximation algorithms for Min UnCut, Min 2CNF Deletion, and directed cut problems". In: *Proceedings of STOC*. ACM, pp. 573–581.

Ahuja, Ravindra K and James B Orlin (2001). "Inverse optimization". In: *Operations Research* 49.5, pp. 771–783.

Arvind, Vikraman et al. (2012). "Approximate graph isomorphism". In: *Mathematical Foundations of Computer Science 2012*. Springer, pp. 100–111.

Audibert, Jean-Yves, Sébastien Bubeck, and Gábor Lugosi (2013). "Regret in online combinatorial optimization". In: *Mathematics of Operations Research* 39.1, pp. 31–45.

Austrin, Per, Subhash Khot, and Muli Safra (2009). "Inapproximability of vertex cover and independent set in bounded degree graphs". In: *Computational Complexity, CCC*. IEEE, pp. 74–80.

Avis, David and Hans Raj Tiwary (2015). "On the extension complexity of combinatorial polytopes". In: *Mathematical Programming* 153.1, pp. 95–115.

Babai, László (2015). "Graph Isomorphism in Quasipolynomial Time". In: *arXiv:1512.03547*. arXiv: 1512.03547 [cs.CC]. URL: http://arxiv.org/abs/1512.03547.

Bazzi, Abbas et al. (2015). "No Small Linear Program Approximates Vertex Cover within a Factor 2–e". In: *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*. IEEE, pp. 1123–1142.

Bodic, Pierre Le et al. (2015). "Solving MIPs via Scaling-based Augmentation". In: *arXiv preprint arXiv:1509.03206*.

Braun, G., S. Fiorini, and S. Pokutta (2014). "Average case polyhedral complexity of the maximum stable set problem". In: *Proceedings of RANDOM / arXiv:1311.4001*.

Braun, G. and S. Pokutta (2011). “An algebraic take on symmetric extended formulations”. Manuscript.

Braun, G. et al. (2012). “Approximation Limits of Linear Programs (Beyond Hierarchies)”. In: *53rd IEEE Symp. on Foundations of Computer Science (FOCS 2012)*, pp. 480–489. ISBN: 978-1-4673-4383-1. DOI: 10.1109/FOCS.2012.10. arXiv: 1204.0957 [cs.CC].

— (2014a). “Approximation Limits of Linear Programs (Beyond Hierarchies)”. In: *Mathematics of Operations Research*. DOI: 10.1287/moor.2014.0694. DOI: 10.1287/moor.2014.0694. arXiv: 1204.0957 [cs.CC].

Braun, Gábor, Samuel Fiorini, and Sebastian Pokutta (2016). “Average case polyhedral complexity of the maximum stable set problem”. In: *Mathematical Programming* 160.1-2, pp. 407–431.

Braun, Gábor and Sebastian Pokutta (2015a). “The matching polytope does not admit fully-polynomial size relaxation schemes”. In: *Proc. SODA*, pp. 837–846. DOI: 10.1137/1.9781611973730.57.

Braun, Gábor and Sebastian Pokutta (2015b). “The Matching Problem Has No Fully Polynomial Size Linear Programming Relaxation Schemes”. In: *IEEE Transactions on Information Theory* 61.10, pp. 5754–5764. DOI: 10.1109/TIT.2015.2465864.

Braun, Gábor and Sebastian Pokutta (2016). “Common information and unique disjointness”. In: *Algorithmica* 76.3, pp. 597–629.

Braun, Gábor, Sebastian Pokutta, and Aurko Roy (2016). “Strong reductions for extended formulations”. In: *International Conference on Integer Programming and Combinatorial Optimization*. Springer, pp. 350–361.

Braun, Gábor, Sebastian Pokutta, and Daniel Zink (2015). “Inapproximability of combinatorial problems via small LPs and SDPs”. In: *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*. ACM, pp. 107–116.

Braun, Gábor et al. (2014b). “Information-theoretic approximations of the nonnegative rank”. In: *computational complexity*, pp. 1–51.

Braverman, M. and A. Moitra (2013). “An information complexity approach to extended formulations”. In: *Proceedings of STOC*, pp. 161–170.

Briët, Jop, Daniel Dadush, and Sebastian Pokutta (2015). “On the existence of 0/1 polytopes with high semidefinite extension complexity”. In: *Mathematical Programming* 153.1, pp. 179–199.

Buss, Sam et al. (1999). “Linear gaps between degrees for the polynomial calculus modulo distinct primes”. In: *Proc. STOC*, pp. 547–556.

Chan, Siu On (2013). “Approximation Resistance from Pairwise Independent Subgroups”. In: *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*. STOC. ACM, pp. 447–456. ISBN: 978-1-4503-2029-0. DOI: 10.1145/2488608.2488665.

Chan, S.O. et al. (2013). “Approximate constraint satisfaction requires large LP relaxations”. In: *Proceedings of FOCS*, pp. 350–359. DOI: 10.1109/FOCS.2013.45.

Charikar, M., K. Makarychev, and Y. Makarychev (2009). “Integrality gaps for Sherali-Adams relaxations”. In: *Proceedings of STOC*, pp. 283–292.

Chawla, Shuchi et al. (2006). “On the hardness of approximating multicut and sparsest-cut”. In: *computational complexity* 15.2, pp. 94–114.

Chlebík, Miroslav and Janka Chlebíková (2004). “On approximation hardness of the minimum 2SAT-DELETION problem”. In: *Mathematical Foundations of Computer Science 2004*. Springer, pp. 263–273.

Cohen, Alon and Tamir Hazan (2015). “Following the Perturbed Leader for Online Structured Learning”. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1034–1042.

Dash, Sanjeeb (2013). “A note on QUBO instances defined on Chimera graphs”. In: *preprint arXiv:1306.1202*.

Dixon, J.D. and B. Mortimer (1996). *Permutation groups*. Springer Verlag.

Edmonds, J. (1965). “Maximum matching and a polyhedron with 0, 1 vertices”. In: *Journal of Research National Bureau of Standards* 69B, pp. 125–130.

Feige, Uriel and Michel Goemans (1995). “Approximating the value of two power proof systems, with applications to max 2sat and max dicut”. In: *Theory of Computing and Systems, 1995. Proceedings., Third Israel Symposium on the*. IEEE, pp. 182–189.

Feige, Uriel, Marek Karpinski, and Michael Langberg (2002). “Improved approximation of Max-Cut on graphs of bounded degree”. In: *Journal of Algorithms* 43.2, pp. 201–219. ISSN: 0196-6774. DOI: 10.1016/S0196-6774(02)00005-6. URL: http://www.paradise.caltech.edu/~mikel/papers/deg_cut.ps.

Feige, Uriel et al. (1991). “Approximating clique is almost NP-complete”. In: *Proceedings of FOCS*. IEEE Comput. Soc. Press., pp. 2–12. ISBN: 0-8186-2445-0. DOI: 10.1109/SFCS.1991.185341.

Fiorini, S. et al. (2012). “Linear vs. Semidefinite Extended Formulations: Exponential Separation and Strong Lower Bounds”. In: *Proceedings of STOC 2012*.

Fiorini, Samuel et al. (2015). “Exponential Lower Bounds for Polytopes in Combinatorial Optimization”. In: *J. Assoc. Comput. Mach.* 62.2, p. 17. DOI: 10.1145/2716307.

Frank, András and Éva Tardos (1987). “An application of simultaneous Diophantine approximation in combinatorial optimization”. In: *Combinatorica* 7.1, pp. 49–65.

Frank, Marguerite and Philip Wolfe (1956). “An algorithm for quadratic programming”. In: *Naval research logistics quarterly* 3.1-2, pp. 95–110.

Garber, Dan and Elad Hazan (2013). “A linearly convergent conditional gradient algorithm with applications to online and stochastic optimization”. In: *arXiv preprint arXiv:1301.4666*.

Garber, Dan and Ofer Meshi (2016). “Linear-Memory and Decomposition-Invariant Linearly Convergent Conditional Gradient Algorithm for Structured Polytopes”. In: *Advances in Neural Information Processing Systems*, pp. 1001–1009.

Garey, Michael R. and David S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. A Series of Books in the Mathematical Sciences. New York, NY, USA: W. H. Freeman and Company, pp. x+338. ISBN: 0-7167-1045-5.

Goemans, Michel X. (2015). “Smallest compact formulation for the permutahedron”. In: *Math. Program.* 153.1, pp. 5–11. DOI: 10.1007/s10107-014-0757-1.

Goemans, Michel X. and David P. Williamson (1994). “New 3/4-approximation algorithms for the maximum satisfiability problem”. In: *SIAM J. Disc. Math.* 7.4, pp. 656–666.

Gouveia, Joao, Pablo A Parrilo, and Rekha R Thomas (2013). “Lifts of convex sets and cone factorizations”. In: *Mathematics of Operations Research* 38.2, pp. 248–264.

Grigoriev, Dima (2001). “Linear lower bound on degrees of Positivstellensatz calculus proofs for the parity”. In: *Theoretical Computer Science* 259.1, pp. 613–622.

Grötschel, Martin and László Lovász (1993). *Combinatorial optimization: A survey*.

Gupta, Swati, Michel Goemans, and Patrick Jaillet (2016). “Solving Combinatorial Games using Products, Projections and Lexicographically Optimal Bases”. In: *arXiv preprint arXiv:1603.00522*.

Gurobi Optimization (2016). *Gurobi optimizer reference manual version 6.5*. URL: https://www.gurobi.com/documentation/6.5/refman/.

Håstad, Johan (2001). "Some optimal inapproximability results". In: *Journal of the ACM (JACM)* 48.4, pp. 798–859.

Hazan, Elad (2016). "Introduction to online convex optimization". In: *Foundations and Trends in Optimization* 2.3–4, pp. 157–325. DOI: 10.1561/2400000013. URL: http://ocobook.cs.princeton.edu/.

Hazan, Elad and Satyen Kale (2012). "Projection-free Online Learning". In: *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pp. 521–528.

Jaggi, Martin (2013). "Revisiting Frank–Wolfe: Projection-free sparse convex optimization". In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 427–435.

Joulin, Armand, Kevin Tang, and Li Fei-Fei (2014). "Efficient image and video co-localization with frank-wolfe algorithm". In: *European Conference on Computer Vision*. Springer, pp. 253–268.

Kaibel, V., K. Pashkovich, and D.O. Theis (2010). "Symmetry Matters for the Sizes of Extended Formulations". In: *Proc. IPCO 2010*, pp. 135–148.

Kaibel, Volker and Stefan Weltge (2015). "A short proof that the extension complexity of the correlation polytope grows exponentially". In: *Discrete & Computational Geometry* 53.2, pp. 397–401.

Kalai, Adam and Santosh Vempala (2005). "Efficient algorithms for online decision problems". In: *Journal of Computer and System Sciences* 71.3, pp. 291–307.

Kann, Viggo et al. (1997). "On the Hardness of Approximating Max-k-CUT and Its Dual". In: *Chicago Journal of Theoretical Computer Science* 1997.2, pp. 1–18.

Karloff, Howard (1999). "How Good is the Goemans–Williamson MAX CUT Algorithm?" In: *SIAM Journal on Computing* 29.1, pp. 336–350. DOI: 10.1137/S0097539797321481.

Khanna, Sanjeev et al. (2001). "The approximability of constraint satisfaction problems". In: *SIAM Journal on Computing* 30.6, pp. 1863–1920.

Khot, S. et al. (2007). "Optimal Inapproximability Results for MAX-CUT and Other 2-Variable CSPs?" In: *SIAM J. Comput.* 37.1, pp. 319–357.

Koch, Thorsten et al. (2011). "MIPLIB 2010". In: *Mathematical Programming Computation* 3.2, pp. 103–163. DOI: 10.1007/s12532-011-0025-9. URL: http://mpc.zib.de/index.php/MPC/article/view/56/28.

Kothari, Pravesh, Raghu Meka, and Prasad Raghavendra (2016). "Approximating Rectangles by Juntas and Weakly-Exponential Lower Bounds for LP Relaxations of CSPs". In: *arXiv preprint arXiv:1610.02704*.

Lacoste-Julien, Simon and Martin Jaggi (2015). "On the Global Linear Convergence of Frank–Wolfe Optimization Variants". In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc., pp. 496–504. URL: http://papers.nips.cc/paper/5925-on-the-global-linear-convergence-of-frank-wolfe-optimization-variants.pdf.

Lan, Guanghui and Yi Zhou (2014). "Conditional gradient sliding for convex optimization". In: *Optimization-Online preprint (4605)*.

Lee, James R et al. (2014). "On the Power of Symmetric LP and SDP Relaxations". In: *Proceedings of the 2014 IEEE 29th Conference on Computational Complexity*. IEEE Computer Society, pp. 13–21.

Lee, J.R., P. Raghavendra, and D. Steurer (2014). "Lower bounds on the size of semidefinite programming relaxations". In: *STOC 2015, arXiv:1411.6317*. cs.CC: 1411.6317v1.

Levitin, Evgeny S and Boris T Polyak (1966). "Constrained minimization methods". In: *USSR Computational mathematics and mathematical physics* 6.5, pp. 1–50.

Neu, Gergely and Gábor Bartók (2013). "An efficient algorithm for learning with semi-bandit feedback". In: *Algorithmic Learning Theory*. Springer, pp. 234–248.

Oertel, Timm, Christian Wagner, and Robert Weismantel (2014). "Integer convex minimization by mixed integer linear optimization". In: *Operations Research Letters* 42.6, pp. 424–428.

Papadimitriou, Christos H and Mihalis Yannakakis (1991). "Optimization, approximation, and complexity classes". In: *Journal of computer and system sciences* 43.3, pp. 425–440.

Pashkovich, K. (2012). "Extended Formulations for Combinatorial Polytopes". PhD thesis. Magdeburg Universität.

Pashkovich, Kanstantsin (2014). "Tight Lower Bounds on the Sizes of Symmetric Extensions of Permutahedra and Similar Results". In: *Math. Oper. Res.* 39.4, pp. 1330–1339. DOI: 10.1287/moor.2014.0659.

Pokutta, Sebastian and Mathieu Van Vyve (2013). "A note on the extension complexity of the knapsack polytope". In: *Operations Research Letters* 41.4, pp. 347–350.

Rothvoß, Thomas (2013). "Some 0/1 polytopes need exponential size extended formulations". In: *Mathematical Programming* 142.1-2, pp. 255–268.

Rothvoß, Thomas (2014). "The matching polytope has exponential extension complexity". In: *Proceedings of STOC*. arXiv: 1311.2369.

Schoenebeck, G. (2008). "Linear Level Lasserre Lower Bounds for Certain k-CSPs". In: *Proceedings of FOCS*, pp. 593–602. DOI: 10.1109/FOCS.2008.74.

Schöning, Uwe (1988). "Graph isomorphism is in the low hierarchy". In: *Journal of Computer and System Sciences* 37.3, pp. 312–323.

Schrijver, A. (1986). *Theory of Linear Programming*. Wiley-Interscience.

Schulz, Andreas S and Robert Weismantel (2002). "The complexity of generic primal algorithms for solving general integer programs". In: *Mathematics of Operations Research* 27.4, pp. 681–692.

Schulz, Andreas S., Robert Weismantel, and Günter M. Ziegler (1995). "0/1-Integer Programming: Optimization and Augmentation are Equivalent". In: *Algorithms – ESA '95, Proceedings*, pp. 473–483.

Sherali, H. D. and W. P. Adams (1990). "A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems". In: *SIAM J. Discrete Math.* 3, pp. 411–430. DOI: 10.1137/0403036.

Singh, Mohit (2010). "Bellairs Workshop on Approximation Algorithms". Open problem session #1.

Trevisan, Luca (1998). "Parallel Approximation Algorithms by Positive Linear Programming". In: *Algorithmica* 21.1, pp. 72–88. ISSN: 0178-4617 (print) and 1432-0541 (online). DOI: 10.1007/PL00009209.

Yannakakis, M. (1988). "Expressing Combinatorial Optimization Problems by Linear Programs (Extended Abstract)". In: *Proc. STOC 1988*, pp. 223–228.

— (1991). "Expressing combinatorial optimization problems by linear programs". In: *J. Comput. System Sci.* 43.3, pp. 441–466. DOI: 10.1016/0022-0000(91)90024-Y.