A conditional gradient homotopy method with applications to Semidefinite Programming

Pavel Dvurechensky\textsuperscript{1}, Shimrit Shtern\textsuperscript{3}, and Mathias Staudigl\textsuperscript{4}

\textsuperscript{1}Weierstrass Institute for Applied Analysis and Stochastics, Mohrenstr. 39, 10117 Berlin, Germany, (Pavel.Dvurechensky@wias-berlin.de)
\textsuperscript{3}Faculty of Industrial Engineering and Management, Technion - Israel Institute of Technology, Haifa, Israel, (shimrits@technion.ac.il)
\textsuperscript{4}Department of Data Science and Knowledge Engineering, Maastricht University, P.O. Box 616, NL–6200 MD Maastricht, The Netherlands, (m.staudigl@maastrichtuniversity.nl)

July 8, 2022

Abstract

We propose a new homotopy-based conditional gradient method for solving convex optimization problems with a large number of simple conic constraints. Instances of this template naturally appear in semidefinite programming problems arising as convex relaxations of combinatorial optimization problems. Our method is a double-loop algorithm in which the conic constraint is treated via a self-concordant barrier, and the inner loop employs a conditional gradient algorithm to approximate the analytic central path, while the outer loop updates the accuracy imposed on the temporal solution and the homotopy parameter. Our theoretical iteration complexity is competitive when confronted to state-of-the-art SDP solvers, with the decisive advantage of cheap projection-free subroutines. Preliminary numerical experiments are provided for illustrating the practical performance of the method.

1 Introduction

In this paper we investigate a new algorithm for solving a large class of convex optimization problems with conic constraints of the following form:

$$\min_{x} g(x) \quad \text{s.t. } x \in X, \mathcal{P}(x) \in K \subseteq H,$$

where \( g : E \to (-\infty, \infty] \) is a closed convex and lower semi-continuous function, \( X \subseteq E \) is a bounded closed and convex set, \( \mathcal{P} : E \to H \) is an affine mapping between two finite-dimensional Euclidean vector spaces \( E \) and \( H \), and \( K \) is a closed convex pointed cone. Problem (P) is sufficiently generic to cover many optimization settings considered in the literature.

Example 1.1 (Packing SDP). The model template (P) covers the large class of packing semidefinite programs (SDPs) \([10, 15]\). In this problem class we have \( E = \mathbb{S}^n \) – the space of symmetric \( n \times n \) matrices, a collection of positive semidefinite input matrices \( A_1, \ldots, A_m \in \mathbb{S}^n_+ \), with constraints given by \( \mathcal{P}(x) = [\text{tr}(A_1x) - 1; \ldots; \text{tr}(A_mx) - 1]^{\top} \) and \( K = \mathbb{R}^m_+ \), coupled with the objective function \( g(x) = \langle c, x \rangle \) and the set \( X = \{x \in \mathbb{S}^n_+ | \text{tr}(x) \leq \rho \} \).

*Corresponding Author
**Example 1.2 (Sparse Recovery).** Recovering a signal from sparse measurements is a classical task in signal processing. A popular convex relaxation of this problem is the norm minimization problem

\[
\min_{x} \|x\| \quad \text{s.t.:} \quad x \in S^n_+, \|Ax - b\|^2 \leq \delta.
\]

Here \(\|\cdot\|\) is a suitable matrix norm (usually the nuclear norm to obtain a low-rank solution), \(Ax\) is the predicted output of the linear model \(b = Ax' + \xi\), where \(x'\) is the ground truth and \(\xi\) is the observation error. Let \(E = S^n = H\), and set \(P(x) = x, K = S^n_+,\) as well as \(X = \{x \in S^n | \frac{1}{2}\|Ax - b\|^2 - \delta \leq 0\}\), where \(\delta > 0\) is an a-priori bound on \(\frac{1}{2}\|\xi\|^2\). See [24] for a survey and [14] for an application of Conditional Gradient (CndG) methods to this norm minimization problem.

**Example 1.3 (Covering SDP).** A dual formulation of the packing SDP is the covering problem [10]. In its normalized version, the problem reads as

\[
\min \sum_{i=1}^{m} x_i \quad \text{s.t.:} \quad \sum_{i=1}^{m} x_i A_i - I \succeq 0, x \in \mathbb{R}^m_+,
\]

where \(A_1, \ldots, A_m \in S^n_+\) are given input matrices. This can be seen as a special case of problem (P) with \(g(x) = \sum_{i=1}^{m} x_i, P(x) = \sum_{i=1}^{m} x_i A_i - I, K = S^n_+, H = S^n, E = \mathbb{R}^m,\) and the constraint set \(X = \{x \in \mathbb{R}^m_+ | \sum_{i=1}^{m} x_i \leq \rho\}\) for some large \(\rho\) added to the problem for the compactification purposes.

The model template (P) can be solved with moderate accuracy using fast proximal gradient methods, like FISTA and related acceleration tricks [8]. In this formalism, the constraints imposed on the optimization problem have to be included in proximal operators that essentially means that one can calculate a projection onto the feasible set. Surprisingly, the computation of proximal operators can impose a significant burden and even lead to intractability of gradient-based methods. This is in particular the case in SDPs with a large number of constraints and decision variables, such as in convex relaxations of NP-hard combinatorial optimization problems [19]. Other prominent examples in machine learning are k-means clustering [23], k-nearest neighbor classification [27], sparse PCA [5], kernel learning [18], among many others. In many of these mentioned applications, it is not only the size of the problem that challenges the application of off-the-shelf solvers, but also the desire to obtain sparse solutions. As a result, the CndG method [11] has seen renewed interest in large-scale convex optimization, since it requires only a Linear minimization oracle (LMO) in each iteration. Compared to proximal methods, CndG features significantly reduced computational costs (e.g. when \(X\) is a spectrahedron), tractability and interpretability (e.g. they generate solutions as a combination of atoms of \(X\)). Modern references to this classical methodology are (among many others) [16, 17].

In this paper we propose a CndG framework for solving (P) with rigorous iteration complexity guarantees. Our approach retains the simplicity of CndG, but allows to disentangle the sources of complexity describing the problem. Specifically, we study the performance of a new homotopy/path-following method, which iteratively approaches a solution of problem (P). To develop the path-following argument we start by decomposing the feasible set into two components. We assume that \(X\) is a compact convex set which admits efficient applications of CndG. The challenging constraints are embodied by the membership condition \(P(x) \in K\). These are conic restrictions on the solution, and in typical applications of our method we have to manage a lot of them. We propose to do so via a logarithmically homogeneous barrier for the cone \(K\). [22] introduced this class of functions in connection with polynomial-time interior point methods. It is a well-known fact that any closed convex cone admits a logarithmically homogeneous barrier (the universal barrier) [22, Thm. 2.5.1]. A central assumption of our approach is that we have a practical representation of a logarithmically homogenous barrier for the cone \(K\). As many restrictions appearing in applications...
are linear or PSD inequalities, this assumption is rather mild. Section 5 presents a wealth of relevant SDPs which can be efficiently handled with our framework. Exploiting the announced decomposition, we approximate the target problem (P) by a family of parametric penalty-based composite convex optimization problems

\[
\min_{x \in \mathcal{X}} \left\{ V_t(x) := \frac{1}{t} F(x) + g(x) \right\},
\]

where \( t > 0 \) is a path-following/homotopy parameter, and \( F \) is a barrier function over the set \( \mathcal{P}^{-1}(K) \). By minimizing the function \( V_t \) over the set \( \mathcal{X} \) for a sequence of increasing values of \( t \), we can trace the analytic central path \( z^*(t) \) of (1.1) as it converges to a solution of (P). Importantly, our approach generates a sequence of feasible points for the original problem (P). Exact homotopy/path-following methods were developed in statistics and signal processing in the context of the \( \ell_1 \)-regularized least squares problem (LASSO) [26] to compute the entire regularization path. Relatedly, approximate homotopy methods have been studied in this context as well [13, 28], and superior empirical performance has been reported for seeking sparse solutions. [30] is the first reference which investigates the complexity of a proximal gradient homotopy method for the LASSO problem. Our paper provides the first complexity analysis of an approximate homotopy CndG algorithm to compute an approximate solution close to the analytic central path for the general problem class (P), building on and extending the recent work [33]. Specifically, [33] study minimization of a regularized self-concordant barrier by a CndG algorithm. Unlike [33], we analyze a more complicated dynamic problem (1.1), where the objective changes as we update the penalty parameter, and we propose also a line-search and inexact-LMO versions of the CndG algorithm for (1.1). Furthermore, we analyze the complexity of the whole path-following process in order to approximate a solution to the initial non-penalized problem (P). One of the challenging questions here is choosing a penalty parameter updates schedule, and we develop a practical answer to this question. To the best of our knowledge, this is the first path-following CndG method with explicit complexity analysis. Specifically, our contributions can be summarized as follows:

- We introduce a simple CndG framework for solving problem (P) and prove that it achieves a \( \tilde{O}(C_1 \varepsilon^{-2} + C_2 \varepsilon^{-1}) \) iteration complexity, where \( C_1, C_2 \) are constants depending on the problem and parameters of the algorithm, and \( \tilde{O}(\cdot) \) hides polylogarithmic factors.

- We provide two extensions for the inner-loop CndG algorithm solving (1.1): a line-search and an inexact-LMO versions. For both, we show that the complexity of the inner and the outer loop are the same as in the basic variant up to constant factors.

- We present key instances of our framework, including instances of SDPs arising from convex relaxations of NP-hard combinatorial optimization problems, and provide promising results of the numerical experiments.

Specializing our setup to specific instances of SDPs, we remark that the theoretical complexity achieved by our method matches or even improves the state-of-the-art iteration complexity results reported in the literature. For packing and covering types of SDPs, the state-of-the-art upper complexity bound reported in [10] is worse than ours since their algorithm has complexity \( \tilde{O}(C_1 \varepsilon^{-2.5} + C_2 \varepsilon^{-2}) \), where \( C_1, C_2 \) are constants depending on the problem data. For SDPs with linear equality constraints and spectrahedron constraints, the primal-dual method CGAL of [32] provides an approximately optimal and feasible point in \( O(\varepsilon^{-2}) \) iterations. In contrast, our method is purely primal, generating feasible points anytime, at the same complexity.

This paper is organized as follows: Section 2 fixes the notation and terminology we use throughout
this paper. Section 3 presents the basic algorithm under consideration in this paper. In section 4 extensions and modifications of the basic scheme are derived and discussed. Section 5 shows how many relevant conic optimization problems, which are considered to be hard instances for perceived methods, can be formulated into our framework. Preliminary numerical results on the MAXCUT and the Markov Chain mixing problem are also reported there. Section 6 contains all the technical proofs of the main theorems presented in this paper.

2 Notation & Preliminaries

Let $E$ be a finite-dimensional Euclidean vector space, $E^*$ its dual space, which is formed by all linear functions on $E$. The value of function $s \in E^*$ at $x \in E$ is denoted by $\langle s, x \rangle$. Let $B : E \to E^*$ be a positive definite self-adjoint operator. Define the norms

$$
\|h\|_B = \langle Bh, h \rangle^{1/2}, \quad h \in E,
$$

$$
\|s\|_B^* = \langle s, B^{-1}s \rangle^{1/2}, \quad s \in E^*,
$$

$$
\|A\| = \max_{\|h\| \leq 1} \|Ah\|, \quad A : E \to E^*.
$$

For a differentiable function $f(x)$ with dom($f$) $\subseteq E$, we denote by $f'(x) \in E^*$ its gradient, by $f''(x) : E \to E^*$ its Hessian, and by $f'''(x)$ its third derivative. Accordingly, the directional derivative of $f$ in direction $h \in E$ is denoted as $f'(x)[h]$.

2.1 Self-Concordant Functions

Let $Q$ be an open and convex subset of $E$. A function $f : E \to (-\infty, \infty]$ is self-concordant (SC) on $Q$ if $f \in C^3(Q)$ and for all $x \in Q, h \in E$, we have $|f'''(x)[h,h,h]| \leq 2(f''(x)[h,h])^{3/2}$. In case where $\mathrm{cl}(Q)$ is a closed convex cone, more structure is available to us. We call $f$ a $\nu$-canonical barrier for $Q$, denoted by $f \in \mathcal{B}_\nu(Q)$, if it is SC and

$$
\forall (x, t) \in Q \times (0, \infty) : \quad f(tx) = f(x) - \nu \log(t).
$$

From [22, Prop. 2.3.4], the following properties are satisfied by a $\nu$-canonical barrier for each $x \in Q, h \in E, t > 0$:

$$
f'(tx) = t^{-1}f'(x), \quad (2.1)
$$

$$
f'(x)[h] = -f''(x)[x,h] \forall h \in E, \quad (2.2)
$$

$$
f'(x)[x] = -\nu \quad (2.3)
$$

$$
f''(x)[x,x] = \nu. \quad (2.4)
$$

We define the local norm $\|h\|_{f''(x)} := (f''(x)[h,h])^{1/2}$ for all $x \in \mathrm{dom}(f)$ and $h \in E$. SC functions are in general not Lipschitz smooth. Still, we have access to a version of a descent lemma of the following form [21, Thm. 5.1.9]:

$$
f(x + h) \leq f(x) + f'(x)[h] + \omega_\nu(\|h\|_{f''(x)}) \quad (2.5)
$$

for all $x \in \mathrm{dom}(f)$ and $h \in E$ such that $\|h\|_{f''(x)} < 1$, where $\omega_\nu(t) = -t - \log(1 - t)$. It also holds true that [21, Thm. 5.1.8]

$$
f(x + h) \geq f(x) + f'(x)[h] + \omega(\|h\|_{f''(x)}) \quad (2.6)
$$
for all \( x \in \operatorname{dom}(f) \) and \( h \in E \) such that \( x + h \in \operatorname{dom}(f) \), where \( \omega(t) = t - \log(1 + t) \) for \( t \geq 0 \).

A classical and useful bound is [21, Lemma 5.1.5]:

\[
\frac{t^2}{2 - t} \leq \omega(t) \leq \frac{t^2}{2(1 - t)} \quad \forall t \in [0, 1).
\]

(2.7)

### 2.2 The Optimization Problem

The following assumptions are made for the rest of this paper.

**Assumption 1.** \( K \subset H \) is a closed convex cone with \( \operatorname{int}(K) \neq \emptyset \), admitting a \( \nu \)-canonical barrier \( f \in \mathcal{B}_\nu(K) \).

The next assumption transports the barrier setup from the codomain \( K \) to the domain \( \mathcal{P}^{-1}(K) \). This is a common operation in the framework of the "barrier calculus" developed in [22, Section 5.1].

**Assumption 2.** The map \( \mathcal{P} : E \to H \) is linear, and \( F(x) := f(\mathcal{P}(x)) \) is a \( \nu \)-canonical barrier on the cone \( \mathcal{P}^{-1}(K) \), i.e. \( F \in \mathcal{B}_\nu(\mathcal{P}^{-1}(K)) \).

Note that \( \operatorname{dom}(F) = \operatorname{int}\mathcal{P}^{-1}(K) \). At this stage a simple example might be useful to illustrate the working of this transportation technique.

**Example 2.1.** Considering the normalized covering problem presented in Example 1.3. [10] solve this problem via a logarithmic potential function method [22], involving the logarithmically homogeneous barrier \( f(X) = \log \det(X) \) for \( X \in S^n_+ \). This is a typical choice in Newton-type methods to impose the semidefiniteness constraint. However, in our projection-free environment, we use the power of the linear minimization oracle to obtain search directions which leaves the cone of positive semidefinite matrices invariant. Instead, we employ barrier functions to incorporate the additional linear constraints in \( \mathcal{P} \). Hence, we set \( F(x) = f(\mathcal{P}(x)) = \log \det(\sum_{i=1}^m x_i A_i - I) \) to absorb the constraint \( \mathcal{P}(x) \in S^n_+ \). In particular, this frees us from matrix inversions, and related computationally intensive steps coming with Newton and interior-point methods.

**Assumption 3.** \( X \) is a nonempty compact convex set in \( E \), and \( C := X \cap \operatorname{dom}(f) \cap \operatorname{dom}(g) \neq \emptyset \).

Let \( \operatorname{Opt} := \min\{g(x) | x \in X, \mathcal{P}(x) \in K \} \). Thanks to assumptions 1 and 3, \( \operatorname{Opt} \) is attained. Our goal is to find an \( \epsilon \)-solution of problem \((P)\), defined as follows.

**Definition 2.1.** Given a tolerance \( \epsilon > 0 \), we say that \( z^*_\epsilon \) is an \( \epsilon \)-solution for \((P)\) if

\[
z^*_\epsilon \in \mathcal{P}^{-1}(K) \cap X \quad \text{and} \quad g(z^*_\epsilon) - \operatorname{Opt} \leq \epsilon.
\]

We underline that we seek for a feasible \( \epsilon \)-solution of problem \((P)\).

Given \( t > 0 \), define

\[
\operatorname{Opt}(t) := \min_{x \in X} \left\{ V_t(x) = \frac{1}{t} F(x) + g(x) \right\}, \quad \text{and} \quad z^*(t) \in \{ x \in X | V_t(x) = \operatorname{Opt}(t) \}.
\]

(2.8)

The following Lemma shows that the path \( t \mapsto z^*(t) \) traces a trajectory in the interior of the feasible set which can be used to approximate a solution of the original problem \((P)\), provided the penalty parameter \( t \) is chosen large enough.

**Lemma 2.2.** For all \( t > 0 \), it holds that \( z^*(t) \in C \). In particular,

\[
g(z^*(t)) - \operatorname{Opt} \leq \frac{\nu}{t}.
\]

(2.9)
Proof. Since $F \in B_{\nu}(\mathcal{P}^{-1}(K))$, it follows immediately that $z^*(t) \in \text{dom}(F) \cap X$. By Fermat’s principle we have

$$0 \in \frac{1}{t}F'(z^*(t)) + \partial g(z^*(t)) + NC_X(z^*(t)).$$

Convexity and [21, Thm. 5.3.7] implies that for all $z \in \text{dom}(F) \cap X$, we have

$$g(z) \geq g(z^*(t)) - \frac{1}{t}F'(z^*(t))[z - z^*(t)] \geq g(z^*(t)) - \frac{\nu}{t}.$$ 

Let $z^*$ be a feasible point satisfying $g(z^*) = \text{Opt}$. Choosing a sequence $\{z^j\}_{j \in \mathbb{N}} \subset \text{dom}(F) \cap X$ with $z^j \to z^*$, the continuity on $\text{dom}(g)$ gives

$$\text{Opt} \geq g(z^*(t)) - \frac{\nu}{t}.$$ 

Lemma 2.2 seems to indicate that our aim of finding an $\epsilon$-solution is already achieved: It seems to suffice to pick $t = \frac{\epsilon}{\nu}$, in order to meet our target. Unfortunately this naïve choice is usually not working well in practice. Choosing $t$ large (i.e. $\epsilon$ small) from the outset means that the subproblems involved in any iterative solver are nearly ill-conditioned. In practice this typically manifests in numerical stability issues and slow convergence. Instead, path-following ideas and continuation methods are usually designed in which the homotopy parameter $t$ is increasing.

3 Algorithm & Convergence

In this section we first describe a new CndG method for solving general conic constrained convex optimization problems of the form (2.8), for a fixed $t > 0$. This procedure serves as the inner loop of our homotopy method. The path-following strategy in the outer loop is then explained in Section 3.2.

3.1 The Proposed CndG Solver

To make our approach efficient, we rely on the following Assumption:

**Assumption 4.** For any $c \in E^*$, the auxiliary problem

$$\mathcal{L}_g(c) := \text{argmin}_{s \in X} \{\langle c, s \rangle + g(s)\}$$ (3.1)

is easily solvable.

We can compute the gradient and Hessian of $F$ as

$$F'(x) = f'(\mathcal{P}(x))\mathcal{P}'(x), \quad F''(x)[s, t] = f''(\mathcal{P}(x))[\mathcal{P}(s), \mathcal{P}(t)] \quad \forall x \in \text{int} \mathcal{P}^{-1}(K).$$

This means that we obtain a local norm on $H$ given by

$$\|w\|_H = F''(x)[w, w]^{1/2} \quad \forall (x, w) \in \text{int} \mathcal{P}^{-1}(K) \times H.$$ 

Note that in order to evaluate the local norm we do not need to compute the full Hessian $F''(x)$. It only requires a directional derivative, which is potentially easy to do numerically. Define the
vector field

\[ s_t(x) \in \mathcal{L}_g(t^{-1}F'(x)). \]  

(3.2)

Note that our analysis does not rely on a specific tie-breaking rule, so any proposal of the oracle (3.1) will be acceptable. To measure solution accuracy and overall algorithmic progress, we introduce two merit functions:

\[ \text{Gap}_t(x) := t^{-1}F'(x)[x - s_t(x)] + g(x) - g(s_t(x)), \]  
\[ \Delta_t(x) := V_t(x) - V_t(z^*(t)). \]  

(3.3)
\n(3.4)

Note that \( \text{Gap}_t(x) \geq 0 \) and \( \Delta_t(x) \geq 0 \) for all \( x \in \text{dom}(F) \). Convexity together with the definition of the point \( z^*(t) \), gives

\[
0 \leq \Delta_t(x) = t^{-1}[F(x) - F(z^*(t))] + g(x) - g(z^*(t)) \\
\leq t^{-1}F'(x)[x - z^*(t)] + g(x) - g(z^*(t)) \\
\leq t^{-1}F'(x)[x - s_t(x)] + g(x) - g(s_t(x)) = \text{Gap}_t(x).
\]

Define

\[ e_t(x) := \|s_t(x) - x\|_x. \]  

(3.5)

Then, for \( \alpha \in (0, \min\{1, 1/e_t(x)\}) \), we get from eq. (2.5)

\[ F(x + \alpha(s_t(x) - x)) \leq F(x) + \alpha F'(x)[s_t(x) - x] + \omega_{t}(\alpha e_t(x)). \]

Together with the convexity of \( g \), this implies

\[ V_t(x + \alpha(s_t(x) - x)) \leq V_t(x) - \alpha \text{Gap}_t(x) + t^{-1}\omega_{t}(\alpha e_t(x)). \]

We optimize the r.h.s. in \( \alpha \) to obtain the analytic step-size policy

\[
\alpha_t(x) := \min \left\{ 1, \frac{t \text{Gap}_t(x)}{e_t(x)(e_t(x) + t \text{Gap}_t(x))} \right\} \in [0, 1/e_t(x)].
\]  

(3.6)

Equipped with this step strategy, procedure \( \text{CG}^*(x^0, \epsilon, t) \), described in Algorithm 1, constructs a sequence \( \{x_t^k\}_{k \geq 0} \) which produces an approximately-optimal solution in terms of the merit function \( \text{Gap}_t(\cdot) \) and the potential function gap \( \Delta_t \). Specifically, the following iteration complexity results can be established.

---

**Algorithm 1: CG\((x^0, \epsilon, t)\)**

**Input:** \((x^0, t) \in \mathbb{C} \times (0, \infty)\) initial state; \(\epsilon > 0\) accuracy level

for \( k = 0, 1, \ldots \) do

if \( \text{Gap}_t(x^k) > \epsilon \) then

Obtain \( s^k = s_t(x^k) \) defined in (3.2).

\( \alpha^k = \alpha_t(x^k) \) defined in (3.6).

Set \( x^{k+1} = x^k + \alpha_t(s^k - x^k) \).

else

Return \( x^k \).

end if

end for
Algorithm 2: Method Homotopy$(x^0, \varepsilon, f)$

Input: $x^0 \in \mathbb{C}, f \in \mathcal{B}_v(K)$.
Parameters: $\varepsilon, t_0, \eta_0 > 0, \sigma \in (0, 1)$.
Initialize: $x^0_{t_0} = x^0$, $I = \lceil \log(2t_0/\varepsilon) \rceil$.
for $i = 0, 1, \ldots, I$ do
  Set $R_i = R(x^0_{t_i}, \eta_i, t_i)$.
  Set $\hat{x}_i = x^0_{t_i}$ the last iterate of CG$(x^0_{t_i}, \eta_i, t_i)$.
  Update $t_{i+1} = t_i/\sigma$, $\eta_{i+1} = \sigma \eta_i$, $x^0_{t_{i+1}} = \hat{x}_i$.
end for

$z = \hat{x}_i$ an $\varepsilon$-solution of $(P)$.

Proposition 3.1. Given $\eta, t > 0$. Let $R(x^0_{t_i}, \eta, t)$ be the first iterate $k$ of Algorithm CG$(x^0_{t_i}, \eta, t)$ satisfying $\text{Gap}_i(x^0_{t_i}) \leq \eta$. Then

$$R(x^0_{t_i}, \eta, t) \leq [5.3(v + t\Delta_i(x^0_{t_i}) + t\Omega_g) \log(10.6t\Delta_i(x^0_{t_i}))] + \frac{24}{t\eta}(v + t\Omega_g)^2,$$

where $\Omega_g := \max_{x, y \in \text{dom}(g)} |x| g(x) - g(y)$.

Proposition 3.2. Algorithm CG$(x^0_{t_i}, \eta, t)$ requires at most

$$N(x^0_{t_i}, \eta, t) := [5.3(v + t\Delta_i(x^0_{t_i}) + t\Omega_g) \log(10.6t\Delta_i(x^0_{t_i}))] + [12(v + t\Omega_g)^2(\frac{1}{t\eta} - \frac{1}{t\Delta_i(x^0_{t_i})})^+]$$

iterations, in order to reach a point satisfying $\Delta_i(x^0_{t_i}) \leq \eta$.

We prove these results in Section 6.1.

### 3.2 Updating the Homotopy Parameter

Our analysis so far focuses on minimization of the potential function $V_t(x)$ for a fixed $t > 0$. However, in order to solve the initial problem $(P)$, one must trace the sequence of approximate solutions as $t_i \uparrow \infty$. The construction of such an increasing sequence of homotopy parameters is the purpose of this section.

Our aim is to reach an $\varepsilon$-solution (Definition 2.1). Let $\{\eta_i\}_{i \geq 0}, \{t_i\}_{i \geq 0}$ be a sequence of approximation errors and homotopy parameters. For each run $i$, we activate procedure CG$(x^0_{t_i}, \eta_i, t_i)$ with the given configuration $(\eta_i, t_i)$. For $i = 0$, we assume to have an admissible initial point $x^0_{t_0} = x^0$ available. For $i \geq 1$, we restart Algorithm 1 using a kind of warm-start strategy by choosing $x^0_{t_i} = x^0_{t_i-1}$, where $R_{i-1} \equiv R(x^0_{t_i-1}, \eta_{i-1}, t_{i-1})$ is the first iterate $k$ of Algorithm CG$(x^0_{t_i-1}, \eta_{i-1}, t_{i-1})$ satisfying $\text{Gap}_{t_{i-1}}(x^0_{t_{i-1}}) \leq \eta_{i-1}$. Note that $R_{i-1}$ is upper bounded in Proposition 3.1. After the $i$-th restart, let us call the obtained iterate $\hat{x}_i := x^0_{t_i}$. In this way we obtain a sequence $\{\hat{x}_i\}_{i \geq 0}$, consisting of candidates for approximate solutions following the central path, as they are $\eta_i$-close in terms of the gap $\text{Gap}_{t_i}$ (and, hence, in terms of the function gap) to the stage $t_i$’s optimal point $z^*(t_i)$. We update the parameters $(t_i, \eta_i)$ as follows:

- The sequence of homotopy parameters is determined as $t_i = t_0 \sigma^{-i}$ for $\sigma \in (0, 1)$ until the last round of Homotopy$(x^0, \varepsilon, f)$ is reached.
The sequence of accuracies requested in the algorithm $\text{CG}(x_i^0, \eta_i, t_i)$ is updated by $\eta_i = \eta_0 \sigma^i$.

The Algorithm stops after $I = I(3, \eta_0, \epsilon) = \lceil \log(2 \eta_0 / \epsilon) \rceil$ updates of the accuracy and homotopy parameters and yields an $\epsilon$-approximate solution of problem (P).

Our updating strategy for the parameters ensures that $\eta_i t_i = t_0 \eta_0$. This equilibrating choice between the increasing homotopy parameter and the decreasing accuracy parameter is sensible, because the iteration complexity of the CndG solver is inversely proportional to $t_i \eta_i$ (cf. Proposition 3.1). Making the judicious choice $\eta_0 t_0 = 2 \nu$, yields a compact assessment of the total complexity of method $\text{Homotopy}(x^0, \epsilon, f)$.

**Theorem 3.3.** Choose $t_0 = \frac{\nu}{\Omega_{\epsilon}}$ and $\eta_0 = 2 \Omega_{\epsilon}$. The total iteration complexity of method $\text{Homotopy}(x^0, \epsilon, f)$ to find an $\epsilon$-solution of (P) is $\text{Comp1}(x^0, \epsilon, f) = \tilde{O}\left(\frac{384 \Omega_{\epsilon}^2 \nu}{\epsilon^2 (1 - \sigma)} + \Omega_{\epsilon} \log(21.2 \nu (1 + (2 / \epsilon) \Omega_{\epsilon}))\left(\frac{21.2 \nu}{2 - \sigma \eta} \frac{2 - \sigma}{1 - \sigma}\right)\right)$, where $\tilde{O}$ hides polylogarithmic factors in $\epsilon^{-1}$.

The theorem leaves open the choice of the parameter $\sigma$. This is a tunable hyperparameter that can be chosen in a tailor-made fashion. The proof of Theorem 3.3 can be found in Section 6.2.

### 4 Extensions

#### 4.1 Line Search

The step size policy employed in Algorithm 1 is inversely proportional to the local norm $e_t(x)$. In particular, its derivation is based on the a-priori restriction that we force our iterates to stay within a trust-region defined by the Dikin ellipsoid. This restriction may force the method to take very small steps, and as a result display bad performance in practice. A simple remedy to this is to use the line search. Given $(x, t) \in \text{dom}(F) \times (0, +\infty)$, let

$$
\gamma_t(x) = \arg\min_{\gamma \in [0, 1]} \text{ s.t. } x + \gamma(s_t(x) - x) \in \text{dom}(F) \quad (4.1)
$$

Thanks to the barrier structure of the potential function $V_t$, some useful consequences can be drawn from the definition of $\gamma_t(x)$. First, $\gamma_t(x) \in \{\gamma \geq 0 | x + \gamma(s_t(x) - x) \in \text{dom}(F)\}$. This implies $x + \gamma_t(x)(s_t(x) - x) \in X \cap \text{dom}(F) \cap \text{dom}(g)$. Second, since $\alpha_t(x)$ is also contained in the latter set, we have

$$
V_t(x + \gamma_t(x)(s_t(x) - x)) \leq V_t(x + \alpha_t(x)(s_t(x) - x)) \quad \forall (x, t) \in \text{dom}(F) \times (0, +\infty).
$$
Via a comparison principle, this allows us to deduce the analysis of the sequence produced by LCG($x^0$, $\epsilon$, $t$) from the analysis of the sequence induced by $\text{CG}(x^0, \epsilon, t)$. Indeed, if $\{\xi^k_t\}_{k \geq 0}$ is the sequence constructed by the line search procedure $\text{LCG}(x^0, \eta, t)$, then we have $V_t(t^{k+1}) \leq V_t(t^k + \alpha_t(\xi^k_t - x^k))$ for all $k$. Hence, we can perform the complexity analysis on the majorizing function as in the analysis of procedure $\text{CG}(x^0, \epsilon, t)$. Consequently, all the complexity-related estimates for the method $\text{CG}(x^0, \eta, t)$ apply verbatim to the sequence $\{\xi^k_t\}_{k \geq 0}$ induced by the method $\text{LCG}(x^0, \eta, t)$.

4.2 Inexact Implementation

Algorithm 1 assumes that the LMO performs exact computation of the search direction $s_t(x)$. This may not be directly available in practice. Just like in [16], we show that our analysis carries essentially through even in the relaxed LMO model with inexact implementation. Let $\gamma \in [0, 1)$ be a given parameter, measuring the approximation quality of the oracle (3.1) relative to the target accuracy $\eta$. Instead of the search direction $s_t(x)$, suppose the LMO returns us a point $\tilde{s}_t = \tilde{s}_t(x)$ satisfying

$$t^{-1}F(x)[\tilde{s}_t(x)] + g(\tilde{s}_t(x)) \leq \min_{s \in X} \left\{ t^{-1}F(x)[s] + g(s) \right\} + \gamma \eta.$$  \hfill (4.2)

Following the derivations performed in the exact computational model, we are defining the inexact gap function

$$\widehat{\text{Gap}}_t(x) := t^{-1}F(x)[x - \tilde{s}_t(x)] + g(x) - g(\tilde{s}_t(x)),$$  \hfill (4.3)

and the associated analytic step size policy, using $\tilde{e}_t(x) := \|\tilde{s}_t(x) - x\|_x$,

$$\tilde{\alpha}_t(x) = \min \left\{ 1, \frac{t\widehat{\text{Gap}}_t(x)}{\tilde{e}_t(x)(\tilde{e}_t(x) + t\widehat{\text{Gap}}_t(x))} \right\}. \hfill (4.4)$$

Remark 4.1. Note that as $\eta \to 0$, the LMO’s accuracy as defined in (4.2) improves. [16] achieves such an improving oracle by coupling the accuracy imposed on the subproblem solver with the $1/k$-step size policy employed in the vanilla CndG.

The next propositions are the resulting iteration complexity statements of Algorithm $\text{ICG}(x^0, \eta, t)$, mimicking the statements in Proposition 3.2 and 3.1. Since the analysis of the inexact implementation regime is analogous to the exact LMO model, we provide sketches of proofs of these statements in Section 6.3.
Proposition 4.1. Algorithm \( ICG(x^0, \eta, t) \) requires at most \( \tilde{N}_g(x^0, \eta, t) \) iterations, in order to reach a point satisfying \( \Delta_t(x^0) \leq \eta \), where

\[
\tilde{N}_g(x^0, \eta, t) := K_{t, g}(x^0) + \lceil 12(v + t\Omega_s)^2 \left( \frac{1}{t\eta} - \frac{1}{t(\Delta_t(x^0) - \gamma\eta)} \right) \rceil,
\]

and

\[
K_{t, g}(x^0) := \begin{cases} 
0 & \text{if } \gamma\eta \geq \frac{1}{10.6t} \\
\lceil 5.3(v + t(\Omega_s - \gamma\eta) + t\Delta_t(x^0)) \rceil \log \left( \frac{10.6t\Delta_t(x^0)}{10.6t\gamma\eta} \right) & \text{if } \gamma\eta < \frac{1}{10.6t}.
\end{cases}
\]

Proposition 4.2. Given \( \eta, t > 0 \) and \( \gamma \in (0, 1/3) \). Let \( \tilde{R}(x^0, \eta, t) \) be the first iterate \( k \) satisfying \( \text{Gap}_g(x^0) \leq \eta \) for Algorithm \( ICG(x^0, \eta, t) \). Then

\[
\tilde{R}(x^0, \eta, t) \leq K_{t, g}(x^0) + \lceil \frac{24}{t(1 - \gamma)\eta} (v + t\Omega_s)^2 \rceil,
\]

where \( K_{t, g}(x^0) \) is defined in (4.6).

Remark 4.2. Observe that all the complexity estimates reduce to the ones proved for the exact oracle model by setting \( \gamma = 0 \).

Based on the above estimates, it becomes clear that the worst-case bounds on the overall iteration complexity of \( \text{Homotopy}(x^0, \varepsilon, f) \) with \( ICG(x^0, \eta, t) \) as the subroutine is the same as in Theorem 3.3, subject to the adjustment of constant factors.

5 Examples and Numerical Experiments

In this section we give several examples covered by the problem template (P) and report some preliminary numerical experiments illustrating the practical performance of our method.

5.1 A Generic Model Problem

The following class of SDPs is studied in [2]. This SDP arises in many algorithms such as approximating MAXCUT, approximating the CUTNORM of a matrix, and approximating solutions to the little Grothendieck problem [3, 4]. The program is given by

\[
\begin{align*}
\text{max} & \quad \langle C, X \rangle \\
\text{s.t.} & \quad X_{ii} \leq 1 \quad i = 1, \ldots, n \\
& \quad X \geq 0
\end{align*}
\]

\[(\text{MAXQP})\]

Let \( Q := \{ y \in \mathbb{R}^n | y_i \leq 1, \ i = 1, \ldots, n \} \), and consider the conic hull of \( Q \), defined as \( K := \{(y, t) \in \mathbb{R}^n \times \mathbb{R}^+ | y \in Q, t > 0 \} \subset \mathbb{R}^{n+1} \). This set admits the \( v = n \) logarithmically homogenous barrier

\[
f(y, t) = - \sum_{i=1}^n \log(t - y_i) = - \sum_{i=1}^n \log(1 - \frac{1}{t} y_i) - n \log(t).
\]

We can then reformulate (MAXQP) as

\[
\begin{align*}
\text{max}_{X, t} & \quad \langle g(X, t) := \langle C, X \rangle \rangle \\
\text{s.t.} & \quad (X, t) \in S^n \times \mathbb{R} | X \geq 0, \ \text{tr}(X) \leq n, t = 1, \| P(X, t) \| \in K
\end{align*}
\]

\[(5.1)\]
where \( P(X, t) := [X_{11}; \ldots; X_{nn}; t]^\top \) is a linear homogenous mapping from \( S^n \times \mathbb{R} \rightarrow \mathbb{R}^{n+1} \). Set \( F(X, t) = f(P(X, t)) \) for \((X, t) \in S^n \times (0, \infty)\).

| Data set | # Nodes | # Vertices | Value | \( \lambda_{\text{min}} \) | Feas. value |
|----------|---------|------------|-------|----------------|-------------|
| G1       | 800     | 19176      | 48333 | 0              | -           |
| G2       | 800     | 19176      | 48357 | 0              | -           |
| G3       | 800     | 19176      | 48337 | 0              | -           |
| G4       | 800     | 19176      | 48446 | 0              | -           |
| G5       | 800     | 19176      | 48400 | 0              | -           |
| G6       | 800     | 19176      | 10668 | 0              | -           |
| G7       | 800     | 19176      | 9977  | 0              | -           |
| G8       | 800     | 19176      | 10141 | 0              | -           |
| G9       | 800     | 19176      | 48446 | 0              | -           |
| G10      | 800     | 19176      | 48446 | 0              | -           |
| G11      | 800     | 19176      | 48446 | 0              | -           |
| G12      | 800     | 19176      | 48446 | 0              | -           |
| G13      | 800     | 19176      | 48446 | 0              | -           |
| G14      | 800     | 19176      | 48446 | 0              | -           |
| G15      | 800     | 19176      | 48446 | 0              | -           |
| G16      | 800     | 4672       | 12700 | 0              | -           |
| G17      | 800     | 4667       | 12685 | 0              | -           |
| G18      | 800     | 4694       | 4695  | 0              | -           |
| G19      | 800     | 4661       | 4364  | 0              | -           |
| G20      | 800     | 4672       | 4476  | 0              | -           |
| G21      | 800     | 4667       | 4448  | 0              | -           |
| G22      | 2000    | 19990      | 73108 | -1             | 48262       |
| G23      | 2000    | 19990      | 73157 | -1             | 39951       |
| G24      | 2000    | 19990      | 73147 | -1             | 39991       |
| G25      | 2000    | 19990      | 73174 | -1             | 39984       |
| G26      | 2000    | 19990      | 73083 | -1             | 39982       |
| G27      | 2000    | 19990      | 33218 | -1             | 16567       |
| G28      | 2000    | 19990      | 33014 | -1             | 16403       |
| G29      | 2000    | 19990      | 33511 | -1             | 16836       |
| G30      | 2000    | 19990      | 33524 | -1             | 16862       |

| Data set | # Nodes | # Vertices | Value | \( \lambda_{\text{min}} \) | Feas. value |
|----------|---------|------------|-------|----------------|-------------|
| G16      | 800     | 4672       | 12700 | 0              | -           |
| G17      | 800     | 4667       | 12685 | 0              | -           |
| G18      | 800     | 4694       | 4695  | 0              | -           |
| G19      | 800     | 4661       | 4364  | 0              | -           |
| G20      | 800     | 4672       | 4476  | 0              | -           |
| G21      | 800     | 4667       | 4448  | 0              | -           |
| G22      | 2000    | 19990      | 73108 | -1             | 48262       |
| G23      | 2000    | 19990      | 73157 | -1             | 39951       |
| G24      | 2000    | 19990      | 73147 | -1             | 39991       |
| G25      | 2000    | 19990      | 73174 | -1             | 39984       |
| G26      | 2000    | 19990      | 73083 | -1             | 39982       |
| G27      | 2000    | 19990      | 33218 | -1             | 16567       |
| G28      | 2000    | 19990      | 33014 | -1             | 16403       |
| G29      | 2000    | 19990      | 33511 | -1             | 16836       |
| G30      | 2000    | 19990      | 33524 | -1             | 16862       |

Table 1: Max-Cut datasets

We apply this formulation to the classical MAXCUT problem in which \( C = L \), i.e. the combinatorial Laplace matrix of an undirected graph \( G = (V, E) \) with vertex set \( V = \{1, \ldots, n\} \) and edge set \( E \). To evaluate the performance, we consider the random graphs G1-G30, published online in [1]. We implement Algorithm 2 using procedure CG and LCG as solvers for the inner loops. The Max-Cut problem was run using Matlab R2019b on an Intel(R) Xeon(R) Gold 6254 CPU @ 3.10GHz server limited to 4 threads per run and 512G total RAM. We used CVX version 2.2 with SDPT3 version 4.0 to obtain the interior point solution which we take as a reference value in order to benchmark our results. The benchmark solutions are displayed in Table 1. The relative gap we compute to assess the relative performance of our method is then computed as \( \frac{g(x) - g(x^{opt})}{|g(x^{opt})|} \). Table 1 also displays the
Figure 1: Max-Cut datasets G1-G21 relative gap from optimal solution vs. iteration.
Figure 2: Max-Cut datasets G1-G21 relative gap from optimal solution vs. time.
size of each dataset, the value obtained by solving the Max-Cut SDP relaxation using CVX with the SDPT3 solver and the minimum eigenvalue of the solution $\lambda_{\text{min}}(X)$. We observe that for larger graphs SDPT3 returns infeasible solutions, featuring negative eigenvalues. If this occurs, the value obtained by a corrected solution is used as a reference value instead. The corrected solution $\tilde{X}$ is obtained as $\tilde{X} = (X - \lambda_{\text{min}}(X)I)/\alpha$ where $I$ is the identity matrix, and $\alpha$ is the minimal number for which $\tilde{X}_{ii} \leq 1$ for all $i = 1, \ldots, n$. Figures 1-6 illustrate the performance of our methods for various parameter values, and Table 2 collects the numerical values obtained for the schemes CG and LCG, in dependence of the scaling factor $\sigma$. 

Figure 3: Max-Cut datasets G1-G21 objective function value vs. iteration.
Figure 4: Max-Cut datasets G1-G21 objective function value vs. time.
Figure 5: Max-Cut datasets G22-G30 objective function value vs. iteration.

Table 2: Results for datasets G1-G30 after a number of iterations, including objective function value, relative gap from SDPT3 corrected solution, and time (in seconds).

| Dataset | Iter. | Alg. | CG | LCG | \(\sigma\) | \(\sigma\) | \(\sigma\) |
|---------|-------|-----|----|-----|-----|-----|-----|
|         |       |     | \(\sigma\) | 0.99 | 0.9 | 0.5 | 0.99 | 0.9 | 0.5 |
| Value   | \(g(z)\) | Gap | Time |     |     |     |     |     |     |
| G1      | 100   | 29141 | 29500 | 28611 | 33151 | 35604 | 35605 |
|         |       | 3.97e-01 | 3.90e-01 | 4.08e-01 | 3.14e-01 | 2.63e-01 | 2.63e-01 |
|         |       | 14 | 15 | 16 | 34 | 50 | 45 |
|         | 1000  | 45292 | 45290 | 45175 | 46034 | 46139 | 46204 |
|         |       | 6.29e-02 | 6.29e-02 | 6.53e-02 | 4.76e-02 | 4.54e-02 | 4.40e-02 |
|         |       | 171 | 169 | 188 | 278 | 305 | 298 |
|         | 10000 | 47785 | 47788 | 47776 | 47813 | 47828 | 47861 |
|         |       | 1.13e-02 | 1.13e-02 | 1.15e-02 | 1.08e-02 | 1.04e-02 | 9.77e-03 |
|         |       | 2013 | 2033 | 2126 | 2619 | 2549 | 2490 |
| G2      | 100   | 29127 | 29371 | 28544 | 33314 | 35423 | 35324 |
|         |       | 3.98e-01 | 3.93e-01 | 4.10e-01 | 3.11e-01 | 2.67e-01 | 2.70e-01 |
|     | 15  | 16  | 16  | 34  | 54  | 54  |
|-----|-----|-----|-----|-----|-----|-----|
| 1000| 45268| 45268| 45134| 46048| 46148| 46203|
|     | 6.39e-02| 6.39e-02| 6.67e-02| 4.78e-02| 4.57e-02| 4.46e-02|
|     | 165| 163| 181| 270| 318| 343|
| 10000| 47809| 47812| 47797| 47835| 47850| 47884|
|     | 1.13e-02| 1.13e-02| 1.16e-02| 1.08e-02| 1.05e-02| 9.79e-03|
|     | 1947| 1957| 1990| 2689| 2753| 2742|
| G3 | 100 | 28622| 28917| 28167| 33150| 35287|
|     | 4.08e-01| 4.02e-01| 4.17e-01| 3.14e-01| 2.70e-01| 2.85e-01|
|     | 16| 16| 16| 33| 50| 48|
| 1000 | 45258| 45241| 45082| 46015| 46111| 46141|
|     | 6.37e-02| 6.41e-02| 6.73e-02| 4.80e-02| 4.61e-02| 4.54e-02|
|     | 175| 172| 193| 286| 310| 316|
| 10000| 47796| 47800| 47784| 47821| 47833| 47874|
|     | 1.12e-02| 1.11e-02| 1.14e-02| 1.07e-02| 1.04e-02| 9.59e-03|
|     | 2066| 2090| 2172| 2664| 2574| 2542|
| G4 | 100 | 28967| 29132| 28376| 33013| 35264|
|     | 4.02e-01| 3.99e-01| 4.14e-01| 3.19e-01| 2.72e-01| 2.68e-01|
|     | 15| 15| 14| 29| 51| 52|
| 1000 | 45402| 45403| 45235| 46183| 46283| 46355|
|     | 6.28e-02| 6.28e-02| 6.63e-02| 4.67e-02| 4.46e-02| 4.32e-02|
|     | 162| 159| 173| 260| 311| 323|
| 10000| 47911| 47914| 47922| 47939| 47952| 47995|
|     | 1.10e-02| 1.10e-02| 1.08e-02| 1.05e-02| 1.02e-02| 9.30e-03|
|     | 1837| 1860| 1911| 2624| 2712| 2661|
| G5 | 100 | 28920| 29414| 28296| 32808| 35454|
|     | 4.02e-01| 3.92e-01| 4.15e-01| 3.22e-01| 2.67e-01| 2.74e-01|
|     | 17| 15| 17| 37| 54| 48|
| 1000 | 45355| 45365| 45184| 46148| 46234| 46247|
|     | 6.29e-02| 6.27e-02| 6.64e-02| 4.65e-02| 4.48e-02| 4.45e-02|
|     | 181| 178| 195| 306| 319| 313|
| 10000| 47866| 47870| 47868| 47891| 47905| 47945|
|     | 1.10e-02| 1.09e-02| 1.10e-02| 1.05e-02| 1.02e-02| 9.40e-03|
|     | 2125| 2113| 2106| 2822| 2593| 2581|
| G6 | 100 | 3119| 6048| 5803| 3280| 7529|
|     | 7.08e-01| 4.33e-01| 4.56e-01| 6.93e-01| 2.94e-01| 2.91e-01|
|     | 15| 17| 21| 26| 16| 36|
| 1000 | 9819| 9788| 9652| 10088| 10062| 10024|
|     | 7.96e-02| 8.25e-02| 9.52e-02| 5.43e-02| 5.68e-02| 6.03e-02|
|     | 202| 208| 246| 280| 134| 281|
| 10000| 10497| 10486| 10450| 10528| 10505| 10471|
|     | 1.60e-02| 1.71e-02| 2.04e-02| 1.31e-02| 1.53e-02| 1.85e-02|
|     | 3188| 3029| 3086| 3941| 1649| 2828|
| G7 | 100 | 3569| 5641| 5359| 3771| 7098|
|     | 6.42e-01| 4.35e-01| 4.63e-01| 6.22e-01| 2.89e-01| 3.07e-01|
|     | 18| 19| 18| 28| 35| 39|
| 1000 | 9171| 9137| 9020| 9438| 9415| 9335|
|     | 8.08e-02| 8.42e-02| 9.59e-02| 5.40e-02| 5.63e-02| 6.43e-02|
|     | G8    |     | G9    |     | G10  |     | G11  |     | G12  |     |
|-----|-------|-----|-------|-----|------|-----|------|-----|------|-----|
|     | 1000  |     | 1000  |     | 1000 |     | 1000 |     | 1000 |     |
|     |       |     |       |     |      |     |      |     |      |     |
|     | 1600-02 | 1740-02 | 2030-02 | 1380-02 | 1600-02 | 1850-02 | 3315 | 3224 | 2878 | 3509 | 3352 | 3095 |
|     | 2788  | 5812 | 5614  | 2931 | 7333 | 7095 | 1000 | 9282 | 9263 | 9141 | 9544 | 9520 | 9486 |
|     | 7230-01 | 4230-01 | 4420-01 | 7090-01 | 2720-01 | 2950-01 | 14 | 17 | 18 | 20 | 24 | 24 |
|     | 9282  | 9263 | 9141  | 9544 | 9520 | 9486 | 1000 | 9910 | 9896 | 9859 | 9931 | 9909 | 9876 |
|     | 1560-02 | 1690-02 | 2050-02 | 1340-02 | 1570-02 | 1890-02 | 3272 | 3186 | 2561 | 2810 | 2359 | 2045 |
|     | 2900  | 5539 | 5214  | 3073 | 6966 | 6708 | 1000 | 9322 | 9300 | 9162 | 9588 | 9570 | 9498 |
|     | 7140-01 | 4540-01 | 4860-01 | 6970-01 | 3130-01 | 3390-01 | 18 | 17 | 18 | 26 | 35 | 36 |
|     | 9322  | 9300 | 9162  | 9588 | 9570 | 9498 | 1000 | 9980 | 9968 | 9934 | 9985 | 9953 | 10000 |
|     | 8070-02 | 8290-02 | 9660-02 | 5460-02 | 5630-02 | 6350-02 | 247 | 236 | 244 | 274 | 289 | 308 |
|     | 9980  | 9968 | 9934  | 9985 | 9953 | 10000 | 9880 | 9804 | 9775 | 9844 | 9825 | 9796 |
|     | 1580-02 | 1710-02 | 2040-02 | 1540-02 | 1860-02 | 1860-02 | 3565 | 3381 | 3101 | 3520 | 3245 | 1000 |
|     | 3378  | 5716 | 5486  | 3557 | 7146 | 6910 | 1000 | 9156 | 9126 | 9007 | 9424 | 9405 | 9325 |
|     | 6610-01 | 4270-01 | 4500-01 | 6430-01 | 2840-01 | 3070-01 | 17 | 17 | 16 | 24 | 28 | 29 |
|     | 9156  | 9126 | 9007  | 9424 | 9405 | 9325 | 1000 | 9816 | 9804 | 9775 | 9844 | 9825 | 9796 |
|     | 8230-02 | 8540-02 | 9730-02 | 5540-02 | 5740-02 | 6540-02 | 224 | 224 | 203 | 235 | 222 | 229 |
|     | 9816  | 9804 | 9775  | 9844 | 9825 | 9796 | 10000 | 9244 | 3087 | 2545 | 3019 | 2664 | 2414 |
|     | 1620-02 | 1730-02 | 2020-02 | 1340-02 | 1520-02 | 1810-02 | 3244 | 3087 | 2545 | 3019 | 2664 | 2414 |
|     | 350   | 635 | 665   | 419  | 853  | 864  | 1000 | 1922 | 1932 | 1911 | 2027 | 2035 | 2021 |
|     | 8620-01 | 7500-01 | 7380-01 | 8350-01 | 6640-01 | 6600-01 | 14 | 15 | 16 | 23 | 22 | 27 |
|     | 1922  | 1932 | 1911  | 2027 | 2035 | 2021 | 1000 | 2396 | 2392 | 2379 | 2409 | 2407 | 2402 |
|     | 2430-01 | 2390-01 | 2470-01 | 2020-01 | 1980-01 | 2040-01 | 211 | 212 | 221 | 264 | 270 | 292 |
|     | 2396  | 2392 | 2379  | 2409 | 2407 | 2402 | 10000 | 5650-02 | 5790-02 | 6310-02 | 5110-02 | 5220-02 | 5420-02 |
|     | 2792  | 2826 | 2650  | 3060 | 3053 | 2891 | 10000 | 5650-02 | 5790-02 | 6310-02 | 5110-02 | 5220-02 | 5420-02 |
|     | 2393  | 2390 | 2380  | 2402 | 2400 | 2389 | 10000 | 369   | 655  | 652  | 422  | 811  | 869  |
|     | 369   | 655 | 652   | 422  | 811  | 869  | 1000 | 1961 | 1969 | 1938 | 2047 | 2051 | 2024 |
|     | 8530-01 | 7390-01 | 7410-01 | 8320-01 | 6770-01 | 6540-01 | 15 | 16 | 17 | 18 | 19 | 21 |
|     | 1961  | 1969 | 1938  | 2047 | 2051 | 2024 | 1000 | 2200-01 | 2170-01 | 2290-01 | 1860-01 | 1840-01 | 1950-01 |
|     | 2200  | 2170 | 2290  | 1860 | 1840 | 1950 | 10000 | 2393 | 2390 | 2380 | 2402 | 2400 | 2389 |
|     | 4800-02 | 4900-02 | 5320-02 | 4430-02 | 4540-02 | 4950-02 | 198 | 200 | 232 | 217 | 229 | 233 |
|     | 4800  | 4900 | 5320  | 4430 | 4540 | 4950 | 10000 | 2393 | 2390 | 2380 | 2402 | 2400 | 2389 |
|   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|
| G13 | 100 | 351 | 665 | 662 | 418 | 829 | 859 |
|   | 8.65e-01 | 7.45e-01 | 7.46e-01 | 8.40e-01 | 6.82e-01 | 6.70e-01 |
|   | 8 | 20 | 17 | 13 | 25 | 25 |
| 1000 | 2034 | 2050 | 2003 | 2124 | 2135 | 2112 |
|   | 2.20e-01 | 2.14e-01 | 2.31e-01 | 1.85e-01 | 1.81e-01 | 1.90e-01 |
|   | 111 | 251 | 231 | 160 | 299 | 283 |
| 10000 | 2490 | 2487 | 2476 | 2499 | 2496 | 2486 |
|   | 4.45e-02 | 4.57e-02 | 4.98e-02 | 4.13e-02 | 4.21e-02 | 4.60e-02 |
|   | 3046 | 3126 | 2741 | 3158 | 3249 | 2849 |
| G14 | 100 | 1313 | 2909 | 2794 | 1335 | 2978 | 3092 |
|   | 8.97e-01 | 7.72e-01 | 7.81e-01 | 8.95e-01 | 7.67e-01 | 7.58e-01 |
|   | 4 | 9 | 7 | 8 | 10 | 8 |
| 1000 | 9101 | 9127 | 8753 | 9381 | 9379 | 9060 |
|   | 2.87e-01 | 2.85e-01 | 3.14e-01 | 2.65e-01 | 2.65e-01 | 2.90e-01 |
|   | 302 | 416 | 350 | 256 | 260 | 244 |
| 10000 | 12071 | 12056 | 11929 | 12141 | 12129 | 12010 |
|   | 5.44e-02 | 5.56e-02 | 6.56e-02 | 4.90e-02 | 4.99e-02 | 5.93e-02 |
|   | 4359 | 4129 | 3151 | 2756 | 2776 | 2488 |
| G15 | 100 | 1092 | 2786 | 2750 | 1103 | 3182 | 3163 |
|   | 9.14e-01 | 7.80e-01 | 7.83e-01 | 9.13e-01 | 7.49e-01 | 7.51e-01 |
|   | 3 | 8 | 6 | 9 | 20 | 16 |
| 1000 | 8824 | 8913 | 8456 | 9133 | 9187 | 8766 |
|   | 3.04e-01 | 2.97e-01 | 3.33e-01 | 2.80e-01 | 2.76e-01 | 3.09e-01 |
|   | 370 | 388 | 325 | 196 | 432 | 365 |
| 10000 | 11961 | 11947 | 11787 | 12022 | 12014 | 11865 |
|   | 5.72e-02 | 5.83e-02 | 7.09e-02 | 5.24e-02 | 5.30e-02 | 6.47e-02 |
|   | 4233 | 3815 | 3188 | 4010 | 4101 | 3329 |
| G16 | 100 | 1474 | 2979 | 2825 | 1451 | 3070 | 2970 |
|   | 8.84e-01 | 7.65e-01 | 7.78e-01 | 8.86e-01 | 7.58e-01 | 7.66e-01 |
|   | 8 | 8 | 4 | 9 | 19 | 16 |
| 1000 | 9014 | 9014 | 8653 | 9314 | 9270 | 8952 |
|   | 2.90e-01 | 2.90e-01 | 3.19e-01 | 2.67e-01 | 2.70e-01 | 2.95e-01 |
|   | 474 | 420 | 273 | 213 | 457 | 423 |
| 10000 | 11981 | 11965 | 11851 | 12052 | 12038 | 11931 |
|   | 5.66e-02 | 5.79e-02 | 6.69e-02 | 5.10e-02 | 5.21e-02 | 6.06e-02 |
|   | 5148 | 4298 | 2691 | 4131 | 4316 | 3339 |
| G17 | 100 | 1175 | 2937 | 2948 | 1179 | 3129 | 3313 |
|   | 9.07e-01 | 7.68e-01 | 7.68e-01 | 9.07e-01 | 7.53e-01 | 7.39e-01 |
|   | 3 | 9 | 6 | 13 | 15 | 16 |
| 1000 | 8869 | 8919 | 8494 | 9110 | 9170 | 8800 |
|   | 3.01e-01 | 2.97e-01 | 3.30e-01 | 2.82e-01 | 2.77e-01 | 3.06e-01 |
|   | 168 | 391 | 330 | 331 | 351 | 336 |
| 10000 | 11942 | 11924 | 11790 | 12006 | 11999 | 11870 |
|   | 5.86e-02 | 6.00e-02 | 7.06e-02 | 5.35e-02 | 5.41e-02 | 6.43e-02 |
|   | 3842 | 3932 | 3807 | 3511 | 3477 | 3448 |
| G18 | 100 | 435 | 2134 | 2038 | 451 | 2643 | 2548 |
|   | 9.07e-01 | 5.45e-01 | 5.66e-01 | 9.04e-01 | 4.37e-01 | 4.57e-01 |
|    |   4   |   26  |   22  |   16  |   30  |   31  |
|----|-------|-------|-------|-------|-------|-------|
| 100| 4075  | 4062  | 3952  | 4230  | 4222  | 4157  |
|    | 1.32e-01 | 1.35e-01 | 1.58e-01 | 9.91e-02 | 1.01e-01 | 1.15e-01 |
|    | 186   | 430   | 384   | 341   | 358   | 373   |
| G19| 100   | 210   | 2062  | 1921  | 209   | 2460  |
|    | 9.52e-01 | 5.27e-01 | 5.60e-01 | 9.52e-01 | 4.36e-01 | 4.56e-01 |
|    | 3     | 25    | 24    | 12    | 28    | 28    |
| 1000| 3816  | 3809  | 3731  | 3944  | 3943  | 3881  |
|    | 1.26e-01 | 1.27e-01 | 1.45e-01 | 9.62e-02 | 9.67e-02 | 1.11e-01 |
|    | 254   | 447   | 451   | 336   | 344   | 337   |
| G20| 100   | 691   | 2141  | 2028  | 702   | 2534  |
|    | 8.46e-01 | 5.22e-01 | 5.47e-01 | 8.43e-01 | 4.34e-01 | 4.31e-01 |
|    | 6     | 26    | 25    | 18    | 32    | 33    |
| 1000| 3931  | 3908  | 3809  | 4058  | 4037  | 3961  |
|    | 1.22e-01 | 1.27e-01 | 1.49e-01 | 9.32e-02 | 9.79e-02 | 1.15e-01 |
|    | 340   | 428   | 444   | 350   | 361   | 376   |
| G21| 100   | 396   | 2161  | 2065  | 403   | 2638  |
|    | 9.11e-01 | 5.14e-01 | 5.36e-01 | 9.09e-01 | 4.07e-01 | 4.41e-01 |
|    | 9     | 27    | 32    | 14    | 28    | 26    |
| 1000| 3890  | 3879  | 3775  | 4025  | 4019  | 3965  |
|    | 1.25e-01 | 1.28e-01 | 1.51e-01 | 9.51e-02 | 9.65e-02 | 1.09e-01 |
|    | 491   | 535   | 557   | 334   | 353   | 337   |
| G22| 100   | 25464 | 24946 | 23293 | 34937 | 35727 |
|    | 4.72e-01 | 4.83e-01 | 5.17e-01 | 2.76e-01 | 2.60e-01 | 3.04e-01 |
|    | 23    | 20    | 21    | 162   | 167   | 76    |
| 1000| 50906 | 50819 | 50279 | 53128 | 53106 | 52908 |
|    | -5.48e-02 | -5.30e-02 | -4.18e-02 | -1.01e-01 | -1.00e-01 | -9.63e-02 |
|    | 257   | 257   | 259   | 1392  | 1447  | 692   |
| 10000| 55618 | 55605 | 55516 | 55827 | 55828 | 55790 |
|    | -1.52e-01 | -1.52e-01 | -1.50e-01 | -1.57e-01 | -1.57e-01 | -1.56e-01 |
|    | 3021  | 3026  | 2870  | 10531 | 10514 | 8783  |
| G23| 100   | 20667 | 25646 | 24530 | 25985 | 36225 |
|    | 4.83e-01 | 3.58e-01 | 3.86e-01 | 3.50e-01 | 9.33e-02 | 1.31e-01 |
|    | 31    | 34    | 32    | 99    | 140   | 126   |
| 1000| 51116 | 51217 | 50980 | 52939 | 53089 | 53198 |
|    | -2.79e-01 | -2.82e-01 | -2.76e-01 | -3.25e-01 | -3.29e-01 | -3.32e-01 |
|    | 394   | 409   | 394   | 1052  | 1088  | 1081  |
| 10000| 55639 | 55641 | 55670 | 55764 | 55769 | 55843 |
|    | -3.93e-01 | -3.93e-01 | -3.93e-01 | -3.96e-01 | -3.96e-01 | -3.98e-01 |
|    | 4605  | 4604  | 4074  | 10485 | 9659  | 9384  |
| G24| 100   | 19364 | 25066 | 23552 | 24082 | 34833 |
|    | 5.16e-01 | 3.73e-01 | 4.11e-01 | 3.98e-01 | 1.29e-01 | 1.57e-01 |
|    | 28    | 33    | 32    | 99    | 123   | 131   |
| 1000| 51035 | 51119 | 50828 | 52847 | 52957 | 53104 |
|    | -2.76e-01 | -2.78e-01 | -2.71e-01 | -3.21e-01 | -3.24e-01 | -3.28e-01 |
|    | 362   | 390   | 389   | 1065  | 1057  | 1123  |
| 10000| 55615 | 55623 | 55646 | 55750 | 55762 | 55826 |
|    | -3.91e-01 | -3.91e-01 | -3.91e-01 | -3.94e-01 | -3.94e-01 | -3.96e-01 |
|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
|   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |
| G25 | 100 | 4371 | 4381 | 4323 | 9844 | 9736 | 9322 |
|     | 18750 | 25136 | 23712 | 23942 | 34972 | 34383 |   |
|     | 5.31e-01 | 3.71e-01 | 4.07e-01 | 4.01e-01 | 1.25e-01 | 1.40e-01 |   |
|     | 28 | 33 | 32 | 102 | 129 | 132 |   |
| 1000 | 51044 | 51159 | 50885 | 52832 | 52984 | 53120 |   |
|     | -2.77e-01 | -2.79e-01 | -2.73e-01 | -3.21e-01 | -3.25e-01 | -3.29e-01 |   |
|     | 380 | 384 | 392 | 1076 | 1066 | 1140 |   |
| 10000 | 55632 | 55640 | 55661 | 55768 | 55778 | 55842 |   |
|     | -3.91e-01 | -3.92e-01 | -3.92e-01 | -3.95e-01 | -3.95e-01 | -3.97e-01 |   |
|     | 4386 | 4384 | 4311 | 9864 | 9732 | 9311 |   |
| G26 | 100 | 18204 | 24203 | 22165 | 22893 | 32934 | 34850 |
|     | 5.45e-01 | 3.95e-01 | 4.46e-01 | 4.27e-01 | 1.76e-01 | 1.28e-01 |   |
|     | 29 | 34 | 33 | 94 | 142 | 126 |   |
| 1000 | 50949 | 51053 | 50717 | 52774 | 52909 | 53029 |   |
|     | -2.74e-01 | -2.77e-01 | -2.69e-01 | -3.20e-01 | -3.23e-01 | -3.26e-01 |   |
|     | 399 | 419 | 410 | 1040 | 1130 | 1083 |   |
| 10000 | 55579 | 55585 | 55608 | 55719 | 55725 | 55789 |   |
|     | -3.90e-01 | -3.90e-01 | -3.91e-01 | -3.94e-01 | -3.94e-01 | -3.95e-01 |   |
|     | 4634 | 4644 | 4127 | 10429 | 9631 | 9361 |   |
| G27 | 100 | 61 | 6809 | 6827 | 61 | 10089 | 7904 |
|     | 9.96e-01 | 5.89e-01 | 5.88e-01 | 7.32e-01 | 3.93e-01 | 4.14e-01 |   |
|     | 19 | 34 | 28 | 97 | 66 | 112 |   |
| 1000 | 14788 | 14900 | 14719 | 15532 | 15474 | 15450 |   |
|     | 1.07e-01 | 1.01e-01 | 1.12e-01 | 6.25e-02 | 6.59e-02 | 6.74e-02 |   |
|     | 484 | 449 | 368 | 856 | 680 | 1058 |   |
| 10000 | 16302 | 16294 | 16274 | 16367 | 16331 | 16329 |   |
|     | 1.60e-02 | 1.65e-02 | 1.77e-02 | 1.20e-02 | 1.42e-02 | 1.44e-02 |   |
|     | 6601 | 4931 | 4085 | 7575 | 6637 | 10163 |   |
| G28 | 100 | 48 | 6691 | 6615 | 49 | 9725 | 9610 |
|     | 9.97e-01 | 5.92e-01 | 5.97e-01 | 7.39e-01 | 4.25e-01 | 4.14e-01 |   |
|     | 20 | 33 | 26 | 89 | 95 | 100 |   |
| 1000 | 14634 | 14743 | 14534 | 15350 | 15296 | 15285 |   |
|     | 1.08e-01 | 1.01e-01 | 1.14e-01 | 6.42e-02 | 6.75e-02 | 6.81e-02 |   |
|     | 480 | 409 | 349 | 677 | 742 | 1043 |   |
| 10000 | 16156 | 16145 | 16120 | 16204 | 16165 | 16164 |   |
|     | 1.51e-02 | 1.57e-02 | 1.73e-02 | 1.21e-02 | 1.45e-02 | 1.46e-02 |   |
|     | 6620 | 4461 | 3817 | 7461 | 7249 | 10230 |   |
| G29 | 100 | 179 | 6951 | 6923 | 5356 | 10267 | 9717 |
|     | 9.89e-01 | 5.87e-01 | 5.89e-01 | 6.82e-01 | 3.90e-01 | 4.23e-01 |   |
|     | 23 | 30 | 27 | 64 | 70 | 65 |   |
| 1000 | 14989 | 15085 | 14864 | 15732 | 15662 | 15482 |   |
|     | 1.10e-01 | 1.04e-01 | 1.17e-01 | 6.55e-02 | 6.97e-02 | 8.04e-02 |   |
|     | 484 | 411 | 337 | 644 | 669 | 642 |   |
| 10000 | 16539 | 16531 | 16503 | 16606 | 16572 | 16527 |   |
|     | 1.76e-02 | 1.81e-02 | 1.97e-02 | 1.36e-02 | 1.56e-02 | 1.83e-02 |   |
|     | 6608 | 4505 | 3819 | 6538 | 6566 | 6082 |   |
| G30 | 100 | 87 | 6814 | 6671 | 89 | 10271 | 10304 |
|     | 9.95e-01 | 5.96e-01 | 6.04e-01 | 6.94e-01 | 4.04e-01 | 3.89e-01 |   |

23
5.2 Estimating Probability Distributions Arising in Genetics

The following SDP arises in the context of estimating haploid frequencies in a population; see [2] for a discussion. The optimization problem reads as

\[
\begin{align*}
\max & \langle C, X \rangle \\
\text{s.t.} & \quad \text{tr}(X) = 1 \\
& \quad X \succeq 0, X \geq 0
\end{align*}
\]

(5.2)

This defines SDP over the double nonnegative cone. To bring this problem into the form of our convex programming template (P), we set

\[X = \{X \in S^n_{++} | \text{tr}(X) \leq 1\}, g(X) = -\langle C, X \rangle, \text{and } K = \{x \in S^n_{++} | x_{ij} \geq 0, i, j = 1, \ldots, n\}\]

the cone of positive matrices. We take \(P(X) = X\), together with the \(\nu = n(n+1)/2\) logarithmically homogeneous barrier

\[F(X) = -\sum_{i,j} \log(X_{ij}),\]

to arrive at a problem instance where our algorithm can be applied directly.

5.3 The Maximum Stable Set Problem

Let \(G = (V, E)\) be a given undirected graph. A set \(W \subset V\) is an independent set of \(G\) if no two vertices in \(W\) are adjacent. The maximum cardinality of an independent set is denoted by \(\alpha(G)\), known as the stability number of \(G\). The maximum independent set problem is to compute \(\alpha(G)\). This problem is known to be NP-hard. A classical semidefinite programming upper bound has been derived in [20], and is based on the following observation. Let \(S\) be an independent set in a graph \(G\) and let \(x \in \{0,1\}^n\) be its incidence vector. Define the matrix \(X = \frac{1}{|S|} xx^\top\). This matrix satisfies the following conditions:

\[X \in S^n_+, X_{ij} = 0 \text{ for all } (i, j) \in E \text{ and } \text{tr}(X) = 1.\]

Furthermore, \(\text{tr}(X11^\top) = \langle X1, 1 \rangle = |S|\). It is therefore natural to consider the following semidefinite program

\[
\begin{align*}
\max & \text{tr}(X11^\top) \\
\text{s.t.:} & \quad \text{tr}(X) = 1, X \in S^n_+, \\
& \quad X_{ij} = 0 \quad \forall (i, j) \in E \\
\end{align*}
\]

(5.3)

Clearly, this is an example of our convex programming template (P), with \(X = \{X \in S^n_+ | \text{tr}(X) = 1, X_{ij} = 0 \text{ for } (i, j) \in E\}\) and \(P\) being trivial. A tighter relaxation can be obtained by adding non-negativity constraints to the entries of the matrix. This leads to the doubly nonnegative relaxation...
of the Max-Stable-Set problem [6, 31]:
\[
\max_X \text{tr}(X11^T) \\
\text{s.t.: } \text{tr}(X) = 1 \\
X_{ij} = 0 \quad \forall (i, j) \in \mathcal{E} \\
X \succeq 0, X \geq 0
\]

(5.4)

In this formulation, we have the added entry-wise restriction \(P(X) = X \geq 0\), which will be absorbed within the log barrier \(F(X) = -\sum_{(i, j) \notin \mathcal{E}} \log(X_{ij})\).

5.4 Finding the Fastest Mixing Markov Chain

We next consider the problem of finding the fastest mixing rate of a Markov chain on a graph [25]. In this problem, a symmetric Markov chain is defined on an undirected graph \(G = ([1, \ldots, n], \mathcal{E})\). Given \(G\) and weights \(d_{ij}\) for \((i, j) \in \mathcal{E}\), we are tasked with finding the transition rates \(w_{ij} \geq 0\) for each \((i, j) \in \mathcal{E}\) with weighted sum smaller than 1, that result in the fastest mixing rate. We assume that \(\sum_{ij} d_{ij}^2 = n^2\). The mixing rate is given by the second smallest eigenvalue of the graph’s Laplacian matrix, described as

\[
L(w)_{ij} = \begin{cases} 
\sum_{j \neq i, j \in \mathcal{E}} w_{ij} & j = i \\
-w_{ij} & (i, j) \in \mathcal{E} \\
0 & \text{otherwise.}
\end{cases}
\]
From $L1 = 0$ it follows that $\frac{1}{n}1$ is a stationary distribution of the process. The mixing time is defined as $T_{\text{mix}} := \sup_{\pi} \|\pi P(t) - \frac{1}{n}1\|_{TV}$, where the supremum is taken with respect to all distributions over the set of nodes, and the norm is the total variation distance. It can be shown that $T_{\text{mix}} \leq \frac{1}{2} \sqrt{n e^{-\lambda_2 t}}$, where $\lambda_2$ is the second-largest eigenvalue of the Laplacian $L$ [29]. To bound this eigenvalue, we follow the strategy laid out in [25]. Thus, the problem can be written as

$$\max_w \lambda_2(L(w))$$

s.t. $\sum_{(i,j) \in E} d_{ij}^2 w_{ij} \leq 1$

$w \geq 0$.

This problem can also alternatively be formulated as

$$\min_w \sum_{(i,j) \in E} d_{ij}^2 w_{ij}$$

s.t. $\lambda_2(L(w)) \geq 1$

$w \geq 0$.

Due to the properties of the Laplacian, the first constraint can be reformulated as

$$\min_w \sum_{(i,j) \in E} d_{ij}^2 w_{ij}$$

s.t. $L(w) \succeq I_{n \times n} - \frac{1}{n}11^\top$

$w \geq 0$. (5.5)

$w \geq 0$. (5.6)

The dual problem is then given by

$$\min_{X \succeq 0} \langle I_{n \times n} - \frac{1}{n}11^\top, X \rangle$$

s.t. $X_{ii} + X_{jj} - 2X_{ij} \leq d_{ij}^2 \quad \{i, j\} \in E$ (5.8)

To obtain an SDP within our convex programming model, we combine arguments from [25] and [2]. Let $X$ be a feasible point for (5.10). Then, there exists a $n \times n$ matrix $V$ such that $X = VV^\top$. It is easy to see that multiplying each row of the matrix $V^\top = [v_1, \ldots, v_n]$ with an orthonormal matrix does not change the feasibility of the candidate solution. Moreover $X_{ij} = v_i^\top v_j$ for all $1 \leq i, j \leq n$, so that $X_{ii} + X_{jj} - 2X_{ij} = \|v_i - v_j\|_2^2$. Without loss of generality, we can normalize the vectors $v_1, \ldots, v_n$ so that $\sum_{i=1}^n v_i = 0$. This implies $\langle I_{n \times n} - \frac{1}{n}11^\top, X \rangle = \text{tr}(X) = \sum_{i=1}^n \|v_i\|^2$. This gives the equivalent optimization problem

$$\max_{v_1, \ldots, v_n} \sum_{i=1}^n \|v_i\|^2$$

s.t. $\|v_i - v_j\|^2 \leq d_{ij}^2 \quad \{i, j\} \in E$, (5.9)

$v_i \in \mathbb{R}^n, \sum_{i=1}^n v_i = 0$.  

26
| Dataset | # Nodes | # Edges | SDPT3 Value |
|---------|---------|---------|-------------|
| 1       | 100     | 1000    | 15.62       |
| 2       | 100     | 2000    | 7.93        |
| 3       | 200     | 1000    | 72.32       |
| 4       | 200     | 4000    | 17.84       |
| 5       | 400     | 1000    | 388.52      |
| 6       | 400     | 8000    | 38.37       |
| 7       | 800     | 4000    | 333.25      |
| 8       | 800     | 16000   | 80.03       |

Table 3: Mixing datasets characteristics.

This is the geometric dual derived in [25], which is strongly connected to the geometric embedding of a graph in the plane [12]. Set \( X = \{ X \in S^n \mid X \succeq 0, \text{tr}(X) \leq \frac{Dn}{2} \} \), where \( D = \max_{ij} d_{ij}^2 \). Then, we can add this trace constraint to the problem formulation, and obtain the equivalent formulation:

Define \( D : S^n \to \mathbb{R}^{|E|} \) given by \( D(X)_{i,j} = X_{ii} + X_{jj} - 2X_{ij} \). Let \( Q = \{ y \in \mathbb{R}^{|E|} \mid d_{ij}^2 \) \ for \( i, j \) \in \mathcal{E} \} \) and \( K = \{ (y, t) \in \mathbb{R}^{|E|} \times \mathbb{R} \mid y \leq d_{ij}^2 \} \). This is a closed convex cone with logarithmically homogeneous barrier

\[
    f(y, t) = -\sum_{(i,j) \in \mathcal{E}} \log(d_{ij}^2 - y_{i,j}) = -\sum_{e \in \mathcal{E}} \log(d_{e}^2 - \frac{1}{t} y_{e}) - |\mathcal{E}| \log(t).
\]

This gives a logarithmically homogeneous barrier \( F(X, t) = f(\mathcal{P}(X, t)) \), where \( \mathcal{P}(X, t) = [D(X); t] \in \mathbb{R}^{|E|} \times \mathbb{R} \).

\[
\begin{align*}
    \min_{X \in S^n, t > 0} & \quad \langle I_n \times n - \frac{1}{n} 11^T, X \rangle \\
    \text{s.t.} & \quad \mathcal{P}(X, t) \in K, t = 1 \\
    & \quad X \in X.
\end{align*}
\]

The Mixing problem was run using Matlab R2021b on an Intel(R) Xeon(R) Gold 6354 CPU @ 3.00GHz server limited to 4 threads per run and 383G total RAM. We generated random connected undirected graphs of various sizes, and for each edge \( (i,j) \) in the graph we generated a random \( d_{ij}^2 \) uniformly in \([0,1]\). Table 3 provides the size of each dataset, and the value obtained by solving the Mixing problem SDP using CVX with SDPT3 solver. All datasets were run for both CG and LCG options with the following choice of parameters \( \eta_0 \in \{0.01 \Omega_g, 0.1 \Omega_g, 0.5 \Omega_g, 1 \Omega_g, 2 \Omega_g\} \) and \( \sigma \in \{0.99, 0.9, 0.5, 0.25\} \). Each of the runs was terminated after it reached at least 10000 iterations at least 3600 seconds running time. Figures 7-8 illustrate the results for some of the parameter values. Table 4 displays the numerical values obtained from our experiments with the best configuration of parameters \( \eta_0 \) and \( \sigma \).

In the experiments we observe that the line search version LCG outperforms CG by orders of magnitude. One explanation of this is that the step size policy employed in CG is based on global optimization ideas which do not take into account the local structure of the problem. Line search captures these local features of the problem much better, leading to better numerical performance.
6 Complexity Analysis

6.1 Analysis of procedure CG(x^0, ε, t)

Recall \( \Omega_g := \max_{x,y \in \text{dom}(g)} |x| |g(x) - g(y)| \) and \( \mathcal{C} = \text{dom}(F) \cap X \cap \text{dom}(g) \). The following estimate can be established as in [33, Prop. 2.3]. Therefore we omit a proof.

**Lemma 6.1.** For all \((x, t) \in \mathcal{C} \times (0, \infty), we have\)

\[
    t^{-1} e_t(x) \leq \nu/t + \text{Gap}_t(x) + \Omega_g. \tag{6.1}
\]

Observe that

\[
e_t(x) = ||s_t(x) - x||_1 = F'(x) [s_t(x) - x, s_t(x) - x]^{1/2} \geq \nu^{-1/2} |F'(x)| ||s_t(x) - x||
\]

\[
= \frac{t}{\nu} |\text{Gap}_t(x) - g(x) + g(s_t(x))|
\]

\[
\geq \frac{t}{\nu} (\text{Gap}_t(x) - \Omega_g)
\]

6.1.1 Proof of Proposition 3.2

Suppose that \( \text{Gap}_t(x) > \nu + \Omega_g \). Then, it readily follows from the previous display that \( e_t(x) \geq \sqrt{\nu} \geq 1 \). This in turn implies \( \frac{t}{\nu} \frac{\text{Gap}_t(x)}{e_t(x)} < \frac{t}{\nu} \frac{\text{Gap}_t(x)}{e_t(x) + t \text{Gap}_t(x)} < 1 \). This suggests a two-phase analysis defined by the threshold \( a_i := \frac{\nu}{\nu} + \Omega_g \). Let \( \{x_t^k\}_{0 \leq k \leq K_i(x^0)} \) be the sequence obtained when running CG(x^0, ε, t) until reaching a point satisfying the stopping criterion \( \text{Gap}_t(x^k(x^0)) \leq \epsilon \). To that end, define \( \mathbb{I}_i(t) := \{k \in \mathbb{N} | \text{Gap}_t(x^k_t) > a_i\} \) and \( \mathbb{I}_H(t) := \{k \in \mathbb{N} | \text{Gap}_t(x^k_t) \leq a_i\} \).

We start with the analysis of the trajectory on \( \mathbb{I}_i(t) \). By the previous estimates, we know that on this phase the algorithm chooses the step sizes \( a_k = \frac{t}{\nu} \frac{\text{Gap}_t(x_t^k)}{e_t(x_t^k)} \). To reduce notational clutter, we set \( e_t^k \equiv e_t(x_t^k) \) and \( G_t^k \equiv \text{Gap}_t(x_t^k) \). In terms of these quantities, the per-iteration reduction of the potential function can be estimated as follows:

\[
V_t(x_t^{k+1}) \leq V_t(x_t^k) - \frac{G_t^k}{e_t^k} \frac{tG_t^k}{e_t^k + tG_t^k} + \frac{1}{t} \alpha \left( \frac{tG_t^k}{e_t^k} + tG_t^k\right)
\]

\[
= V_t(x_t^k) - \frac{G_t^k}{e_t^k} + \frac{1}{t} \log \left( 1 + \frac{tG_t^k}{e_t^k} \right)
\]

\[
= V_t(x_t^k) - \frac{1}{t} \alpha \left( \frac{tG_t^k}{e_t^k} \right), \tag{6.2}
\]

where \( \alpha(t) = t - \log(1 + t) \). This readily yields \( \Delta_t(x_t^k) - \Delta_t(x_t^{k+1}) \geq \frac{1}{t} \alpha \left( \frac{tG_t^k}{e_t^k} \right) \). This implies that \( \{\Delta_t(x_t^k)\}_k \) is monotonically decreasing. For \( k \in \mathbb{I}_i(t) \), we also see \( \frac{tG_t^k}{v + tG_t^k + \Omega_g} > \frac{1}{2} \). Together with (6.1), this implies

\[
\frac{tG_t^k}{e_t^k} \geq \frac{G_t^k}{v/t + G_t^k + \Omega_g} = \frac{tG_t^k}{v + tG_t^k + t\Omega_g} > \frac{1}{2}.
\]
Define \( q \). Furthermore, using (6.1), the monotonicity of \( \omega \), the fact that the function \( x \mapsto \frac{x}{a + bx} \) is strictly increasing for \( a, b > 0 \), and [33, Prop. 2.1], we arrive at
\[
\Delta_t(x_t^k) - \Delta_t(x_t^{k+1}) \geq \frac{1}{5.3} \frac{G_t^k}{v + tG_t^k + t\Omega_g} \geq \frac{\Delta_t(x_t^k)}{5.3(v + t\Omega_g + t\Delta_t(x_t^0))}.
\]
Coupled with \( \Delta_t(x_t^k) \leq \Delta_t(x_t^0) \), we conclude
\[
\Delta_t(x_t^k) \left[ 1 - \frac{1}{5.3(v + t\Omega_g + t\Delta_t(x_t^0))} \right] \geq \Delta_t(x_t^{k+1}).
\]
Define \( q_t := \frac{1}{5.3(v + t\Omega_g + t\Delta_t(x_t^0))} \in (0, 1) \) (since \( v \geq 1 \)), to arrive at the recursion
\[
\Delta_t(x_t^k) \leq (1 - q_t)^k \Delta_t(x_t^0) \leq \exp(-q_t k) \Delta_t(x_t^0).
\]
Furthermore,
\[
\Delta_t(x_t^k) \geq \Delta_t(x_t^0) - \Delta_t(x_t^{k+1}) \geq \frac{G_t^k}{v + tG_t^k + t\Omega_g} \geq \frac{1}{10.6 t}.
\]
Hence, \( \frac{1}{10.6 t \Delta_t(x_t^0)} \leq \exp(-q_t k) \), so that process \( CG(x_t^0, \epsilon, t) \) must exit phase I in at most \( K_t(x_t^0) := [5.3(v + t\Delta_t(x_t^0)) + t\Omega_g] \log(10.6 t \Delta_t(x_t^0)) \) iterations.

We now upper bound the time the process spends in phase II, so that \( G_t^k \leq \frac{v}{t} + \Omega_g \). When the algorithm enters this phase, we have to distinguish the iterates using large step sizes \( (\alpha_k = 1) \), from those using short steps \( (\alpha_k < 1) \). Both regimes lead to a universal lower bound on the per-iteration potential reduction achieved, as we illustrate next.

To start with, let us assume \( \alpha_k = 1 \). Then, \( tG_t^k \geq e_t^k(tG_t^k + e_t^k) \), which implies \( e_t^k \in (0, 1) \), and \( G_t^k \geq \frac{(e_t^k)^2}{t(1-e_t^k)} \). Moreover, using (2.7), we readily obtain
\[
V_t(x_t^{k+1}) \leq V_t(x_t^k) - G_t^k + \frac{1}{t} \omega_t(e_t^k) \\
\leq V_t(x_t^k) - G_t^k + \frac{1}{t} (e_t^k)^2 \\
\leq V_t(x_t^k) - \frac{G_t^k}{2}.
\]
Therefore,
\[
\Delta_t(x_t^k) - \Delta_t(x_t^{k+1}) \geq \frac{G_t^k}{2} \geq \frac{1}{2} \frac{(G_t^k)^2}{v + tG_t^k + t\Omega_g} = \frac{1}{2} \frac{t(G_t^k)^2}{v + t\Omega_g} \geq \frac{1}{12} \frac{t(G_t^k)^2}{(v + t\Omega_g)^2}.
\]
On the other hand, in the regime where \( \alpha_k < 1 \), we obtain the estimate \( \Delta_t(x_t^k) - \Delta_t(x_t^{k+1}) \geq \frac{1}{12} \frac{t(G_t^k)^2}{(v + t\Omega_g)^2} \), by following the same reasoning as in [33]. We conclude
\[
\frac{1}{\Delta_t(x_t^{k+1})} - \frac{1}{\Delta_t(x_t^k)} \geq \frac{t}{12(v + t\Omega_g)^2} \quad \forall k \in \mathbb{K}_t(t).
\]
Given the pair \((t, \eta) \in (0, \infty)^2\), denote by \(N \equiv N(x_0^j, \eta, t)\) an upper bound on the total number of iterations of \(\text{CG}(x_0^j, \eta, t)\). We know that at most \(K \equiv K_t(x_0^j)\) of these iterations are in phase I. To estimate the remaining iterations in phase II, i.e. \(M = N - K\), we first telescope the inequality (6.3), to get

\[
\frac{1}{\Delta_t(x_0^j)} \geq \frac{1}{\Delta_t(x_0^j)} + \frac{M t}{12(v + t\Omega g^2)} \geq \frac{1}{\Delta_0(x_0^j)} + \frac{M t}{12(v + t\Omega g^2)}.
\]

Choosing \(M = \lceil 12(v + t\Omega g^2) \right \rfloor \) ensures that \(\Delta_t(x_0^j) \leq \eta\). This bounds the iteration complexity of algorithm \(\text{CG}(x_0^j, \eta, t)\).

### 6.1.2 Proof of Proposition 3.1

Let \(\{k_j(t)\}_j\) denote the increasing set of indices enumerating \(\mathcal{K}_H(t)\). From the analysis of the trajectory given in the previous section, we know that

\[
\Delta_t(x^k_j(t)) - \Delta_t(x^j(t)) \leq -\frac{t(G^k_j(t))^2}{12(v + t\Omega g^2)}, \text{ as well as } G^k_j(t) \geq \Delta_t(x^k_j(t)).
\]

Set \(d_j \equiv \Delta_t(x^k_j(t))\) and \(\frac{1}{M} \equiv \frac{t}{12(v + t\Omega g^2)}\), \(\Gamma_j \equiv G^k_j(t)\). We thus obtain the recursion

\[
d_{j+1} \leq d_j - \frac{\Gamma_j^2}{M}, \quad \Gamma_j \geq d_j.
\]

We can apply [33, Prop. 2.4] directly to the above recursion to obtain the estimates

\[
d_j \leq \frac{M}{j}, \quad \text{and } \min(\Gamma_0, \ldots, \Gamma_j) \leq \frac{2M}{j}.
\]

Therefore, in order to reach an iterate with \(\text{Gap}(x^j_k) \leq \eta\), we need to run the process until the label \(j\) is reached satisfying \(\frac{2M}{j} \leq \eta\). Solving this for \(j\) yields \(j = \lceil \frac{24(v + t\Omega g^2)}{\eta} \rceil\). Combined with the upper bound obtained for the time the process spends in phase I, we obtain the total complexity estimate postulated in Proposition 3.1.

### 6.2 Analysis of the outer loop

Let \(I = I(\eta_0, \sigma, \varepsilon) = \lceil \frac{\log(2M/\varepsilon)}{\log(1/\sigma)} \rceil\) denote the a-priori fixed number of updates of the accuracy and homotopy parameter. We set \(\hat{x}_i \equiv x_i^{R_i}\) the last iterate of procedure \(\text{CG}(x_0^j, \eta_i, t_i)\) satisfying \(\Delta_t(\hat{x}_i) \leq \text{Gap}_{I_i}(\hat{x}_i) \leq \eta_i\). From Proposition 3.1, we deduce that

\[
g(\hat{x}_i) - g(z^*(t_i)) \leq \text{Gap}_{I_i}(\hat{x}_i) + \frac{1}{t_i}F'(\hat{x}_i)[z^*(t_i) - \hat{x}_i] \leq \eta_i + \frac{v}{t_i}.
\]

Hence, using Lemma 2.2, we observe

\[
g(\hat{x}_i) - \text{Opt} = g(\hat{x}_i) - g(z^*(t_i)) + g(z^*(t_i)) - \text{Opt} \leq \eta_i + \frac{2v}{t_i}.
\] (6.4)
Since $t_i = \frac{2\nu}{\eta_i}$, we obtain $g(x_i) - \text{Opt} \leq 2\eta_i$. We next estimate the initial gap $\Delta_{t_i+1}(x_{t_i+1}^0) = \Delta_{t_i+1}(x_i)$ incurred by our warm-start strategy. Observe that

$$V_{t_{i+1}}(x_{t_{i+1}}^0) - V_{t_{i+1}}(z^*(t_{i+1})) = V_{t_i}(x_i) - V_{t_i}(z^*(t_{i+1})) + \left(\frac{1}{t_{i+1}} - \frac{1}{t_i}\right)(F(x_i) - F(z^*(t_{i+1})))$$

$$\leq V_{t_i}(x_i) - V_{t_i}(z^*(t_i)) + (1 - \frac{t_{i+1}}{t_i}) \left(V_{t_{i+1}}(x_{t_{i+1}}^0) - V_{t_{i+1}}(z^*(t_{i+1}))\right)$$

$$+ \left(1 - \frac{t_{i+1}}{t_i}\right)(g(z^*(t_{i+1})) - g(x_{t_{i+1}}^0)).$$

Whence,

$$\frac{t_{i+1}}{t_i}(V_{t_{i+1}}(x_{t_{i+1}}^0) - V_{t_{i+1}}(z^*(t_{i+1}))) \leq V_{t_i}(x_i) - V_{t_i}(z^*(t_i)) + (1 - \frac{t_{i+1}}{t_i})(g(z^*(t_{i+1})) - g(x_{t_{i+1}}^0))$$

Since $t_{i+1} > t_i$ and the definition of the stopping time $R_i = R(x_i, \eta_i, t_i)$, this implies

$$t_{i+1} \Delta_{t_{i+1}}(x_{t_{i+1}}^0) \leq t_i \Delta_{t_i}^R - (t_{i+1} - t_i) \Delta_{t_i} \leq t_i \eta_i + (t_{i+1} - t_i) \Delta_{t_i} = 2\nu + (t_{i+1} - t_i) \Delta_{t_i}.$$  \hspace{1cm} (6.5)

We are now in the position to estimate the total iteration complexity $\text{Comp}(x_i, \epsilon, f) := \sum_{i=0}^{l} R_i(x_i, \eta_i, t_i)$, and thereby prove Theorem 3.3. We do so by using the updating regime of the sequences $\{t_i\}$ and $\{\eta_i\}$ explained in Section 3. These updating mechanisms imply

$$\sum_{i=1}^{l} R_i \leq \sum_{i=1}^{l} 5.3(3\nu + (t_i - t_{i-1}) \Omega_g + t_i \Omega_g) \log(10.6(2\nu + (t_i - t_{i-1}) \Omega_g)) + \sum_{i=1}^{l} \frac{12}{\nu} (\nu + t_i \Omega_g)^2$$

$$\leq \log(10.6(2\nu + t_1 \Omega_g)) \sum_{i=1}^{l} 5.3(3\nu + t_i \Omega_g + (t_i - t_{i-1}) \Omega_g) + \sum_{i=1}^{l} \frac{24}{\nu} (\nu^2 + t_i^2 \Omega_g^2)$$

$$\leq 15.9\nu \log(10.6(2\nu + t_1 \Omega_g)) + 5.3\Omega_g \log(10.6(2\nu + t_1 \Omega_g)) \sum_{i=1}^{l} t_i$$

$$+ 5.3\Omega_g \log(10.6(2\nu + t_1 \Omega_g)) \sum_{i=1}^{l+1} (t_i - t_{i-1}) + 24\nu l + \frac{24\Omega_g}{\nu} \sum_{i=1}^{l} t_i^2$$

$$\leq l \left[24\nu + 15.9\nu \log(10.6(2\nu + t_1 \Omega_g)) + 5.3\Omega_g \log(10.6(2\nu + t_1 \Omega_g))\right]$$

$$+ \frac{24\Omega_g^2}{\nu} \sum_{i=1}^{l} t_i^2 + 5.3\Omega_g \log(10.6(2\nu + t_1 \Omega_g)) \sum_{i=1}^{l} t_i.$$

Let us estimate the terms appearing in this expression. First,

$$t_i \leq t_0 \sigma^{-l} \leq \frac{4\nu}{\epsilon}.$$  

Second,

$$\sum_{i=1}^{l} t_i = t_0 \sigma^{-l} \sum_{i=0}^{l-1} (1/\sigma)^i \leq t_0 \sigma^{-l} \leq \frac{4\nu}{\epsilon(1 - \sigma)}.$$  

31
6.3 Modifications Due to Inexact Oracles

In this section we outline the changes that need to be made in evaluating the iteration complexity under inexact oracle information. Consider algorithm IC\(_G(x^0, \eta, t)\) with accuracy parameter \(\gamma \in [0, 1)\). Fix \(t > 0\) and define \(\bar{e}^k \equiv e_t(x^k)\) as well as \(\bar{\text{Gap}}_t^k(x^k)\). The following modifications have to be made to replicate the results from the exact to the inexact oracle model.

1. Lemma 6.1 is replaced by

\[
t^{-1}\bar{e}_t(x) \leq v/t + \overline{\text{Gap}}_t(x) + \Omega_g.
\]
2. We can show that the sequence of function gaps \( \{\Delta_t(x^k_t)\}_k \) is monotonically decreasing under the sequence \( \{x^k_t\}_k \) generated by ICG\((x^0, \eta, t)\).

3. The phases of the CndG algorithm are characterized by \( K_i(t) := \{k \in \mathbb{N} | \text{Gap}_i(x^k_t) \geq a_i\} \), and \( K_i(t) := \{k \in \mathbb{N} | \text{Gap}_i(x^k_t) \leq a_i\} \).

4. In phase I, the linear convergence is contaminated by a perturbed linear convergence rate in terms of the potential function gap:

\[
\Delta_t(x^k_t) \leq (1 - \bar{q}_t)^k \Delta_t(x^0) + \gamma \eta,
\]

where \( \bar{q}_t := \frac{1}{5.3(v + t(\Omega_g - \gamma \eta) + \Delta_t(x^0))} \). Hence, the bound on the time the process spends in phase I becomes

\[
K_{i, \gamma}(x^0_t) := \begin{cases} 
0 & \text{if } \gamma \eta \geq \frac{1}{10.6t} \\
\lceil 5.3(v + t(\Omega_g - \gamma \eta) + t\Delta_t(x^0)) \rceil \log \left( \frac{10.6t\Delta_t(x^0)}{1 - 10.6t \gamma \eta} \right) & \text{if } \gamma \eta < \frac{1}{10.6t}.
\end{cases}
\]

5. We continue with the analysis on phase II. Following exactly the same arguments used in the model with exact oracles we arrive at the iteration bound for phase II as

\[
\Delta_t(x^k_t) - \Delta_t(x^{k+1}_t) \geq \frac{1}{12} \frac{t(\tilde{G}^k)^2}{(v + t\Omega_g)^2}.
\]

This shows that even under inexact oracle information, the sequence \( \{\Delta_t(x^k_t)\}_k \) is monotonically decreasing. If \( \Delta_t(x^{k+1}_t) < \eta \), then we are done. Hence, let us assume that \( \Delta_t(x^{k+1}_t) \geq \eta \). Setting \( \tilde{G}^k = \text{Gap}_i(x^k_t) \), and using the relation \( \tilde{G}^k \geq \text{Gap}_i(x^k_t) - \gamma \eta \) for all \( k \), we can proceed as follows:

\[
\Delta_t(x^k_t) - \Delta_t(x^{k+1}_t) \geq \frac{1}{12} \frac{t(\tilde{G}^k)^2}{(v + t\Omega_g)^2} \geq \frac{1}{12} \frac{t(\tilde{G}^k - \gamma \eta)^2}{(v + t\Omega_g)^2} \geq \frac{t}{12} \frac{(\Delta_t(x^k) - \gamma \eta)^2}{(v + t\Omega_g)^2} = \frac{t}{12} \frac{(\Delta_t(x^k) - \gamma \eta)(\Delta_t(x^{k+1}) - \gamma \eta)}{(v + t\Omega_g)^2}.
\]

Hence,

\[
(\Delta_t(x^k_t) - \gamma \eta) - (\Delta_t(x^{k+1}_t) - \gamma \eta) \geq \frac{t}{12} \frac{(\Delta_t(x^k) - \gamma \eta)(\Delta_t(x^{k+1}) - \gamma \eta)}{(v + t\Omega_g)^2}.
\]

Multiplying both sides by \( \frac{1}{(\Delta_t(x^{k+1}) - \gamma \eta)(\Delta_t(x^k) - \gamma \eta)} \) gives

\[
\frac{1}{\Delta_t(x^{k+1}_t) - \gamma \eta} - \frac{1}{\Delta_t(x^k_t) - \gamma \eta} \geq \frac{t}{12(v + t\Omega_g)^2}.
\]

Telescoping this expression just as in the model with exact oracle feedback, we arrive at the iteration bound for phase II as \( \lceil 12(v + t\Omega_g)^2 \left( \frac{1}{\eta} - \frac{1}{t(\Delta_t(x^k_t) - \gamma \eta)} \right) \rceil \). This proves Proposition 4.1.

6. To prove Proposition 4.2, we argue as follows: Let \( \{k_j(t)\}_j \) denote the increasing set of indices enumerating \( K_{ii}(t) \). From the analysis of the trajectory above, we know that

\[
\Delta_t(x_i^{k_j(t+1)}(t)) - \Delta_t(x_i^{k_j(t)}(t)) \leq - \frac{1}{12} \frac{t(\tilde{G}^{k_j(t)})^2}{(v + t\Omega_g)^2} \leq - \frac{1}{12} \frac{t(\text{Gap}_i(x_i^{k_j(t)}(t)) - \gamma \eta)^2}{(v + t\Omega_g)^2}.
\]
Set $d_j \equiv \Delta_i(x_t^{k(t)}) - \gamma \eta$ and $\frac{1}{M} \equiv \frac{t}{12(\nu + \Omega_j)^2}$, $\Gamma_j \equiv \tilde{G}_j^{k(t)}$. Let $J = \sup \{ |d_j| \geq 0 \} \in \mathbb{N}$. We know that $J$ is finite, since $|\Delta_i(x_t^k)|_k$ is monotonically decreasing. For all $j \in \{1, \ldots, J\}$ we have $d_j \geq 0$ and $\Gamma_j \geq d_j$. For $j = 1, \ldots, J-1$, it thus holds true that

$$d_{j+1} \leq d_j - \frac{\Gamma_j^2}{M}, \quad \Gamma_j \geq d_j.$$  

By [33, Prop. 2.4], we have for $j = 1, \ldots, J$,

$$d_j \leq \frac{M}{j}, \quad \text{and} \quad \min \{ \Gamma_0, \ldots, \Gamma_J \} \leq \frac{2M}{J}.$$  

This shows $J = \beta(\eta, \gamma) := \lfloor \frac{12(\nu + \Omega_j)^2}{\eta \gamma} \rfloor$. In order to reach an iterate with $\text{Gap}_i(x_t^k) \leq \eta$, we need to run the process until the label $j$ is reached satisfying $\frac{2M}{J} \leq (1 - \gamma)\eta$. To see this, it suffices to observe

$$\min \{ \text{Gap}_i(x_t^{k(t)}), \ldots, \text{Gap}_i(x_t^{k(t)}) \} - \gamma \eta \leq \min \{ \tilde{G}_i^{k(t)}, \ldots, \tilde{G}_i^{k(t)} \} \leq \frac{2M}{J} \leq (1 - \gamma)\eta$$

$$\Rightarrow \min \{ \text{Gap}_i(x_t^{k(t)}), \ldots, \text{Gap}_i(x_t^{k(t)}) \} \leq \eta.$$  

Solving this for $j$ yields $j = \lfloor \frac{24(\nu + \Omega_j)^2}{t(1 - \gamma)\eta} \rfloor$. Since $\gamma < 1/3$, we see that this bound is smaller than $\beta(\eta, \gamma)$. Hence, once combined with the upper bound obtained for the time the process spends in phase I, we obtain the total complexity estimate postulated in Proposition 4.2.

7 Conclusion

Solving large scale conic-constrained convex programming problems is still a key challenge in mathematical optimization and machine learning. In particular, SDPs with a massive amount of linear constraints appear naturally as convex relaxations of combinatorial optimization problems. For such problems, the development of scalable algorithms is a very active field of research [10, 32]. In this paper, we introduce a new path-following/homotopy strategy to solve such massive conic-constrained problems, building on recent advances in projection-free methods for self-concordant minimization [7, 9, 33]. Our scheme is competitive with state-of-the-art solvers in terms of its theoretical iteration complexity and gives promising results in practice. Future work will focus on improvements of our deployed CndG solver to accelerate the subroutines.

References

[1] Y. Ye, GSet random graphs, https://www.cise.u.edu/research/sparse/matrices/Gset/ (accessed May, 2022).
[2] S. Arora, E. Hazan, and S. Kale. Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. In 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS’05), pages 339–348, 2005. doi: 10.1109/SFCS.2005.35.
[3] Afonso S. Bandeira, Nicolas Boumal, and Vladislav Voroninski. On the low-rank approach for semidefinite programs arising in synchronization and community detection. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, 29th Annual Conference on Learning Theory, volume 49 of Proceedings of Machine Learning Research, pages 361–382, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR. URL https://proceedings.mlr.press/v49/bandeira16.html.
[4] M. Charikar and A. Wirth. Maximizing quadratic programs: extending grothendieck’s inequality. In 45th Annual IEEE Symposium on Foundations of Computer Science, pages 54–60, 2004. doi: 10.1109/FOCS.2004.39.

[5] Alexandre d’Aspremont, Francis Bach, and Laurent El Ghaoui. Optimal solutions for sparse principal component analysis. Journal of Machine Learning Research, 9(7), 2008.

[6] E. de Klerk and D. V. Pasechnik. Approximation of the stability number of a graph via copositive programming. SIAM Journal on Optimization, 12(4):875–892, 2002. doi: 10.1137/S1052623401383248. URL https://doi.org/10.1016/S1052-6234(02)80473-8.

[7] Pavel Dvurechensky, Petr Ostroukhov, Kamil Safin, Shimrit Shtern, and Mathias Staudigl. Self-concordant analysis of Frank-Wolfe algorithms. In Hal Daume III and Aarti Singh, editors, Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 2814–2824, Virtual, 13–18 Jul 2020. PMLR. URL http://proceedings.mlr.press/v119/dvurechensky20a.html. arXiv:2002.04320.

[8] Pavel Dvurechensky, Shimrit Shtern, and Mathias Staudigl. First-order methods for convex optimization. EURO Journal on Computational Optimization, 9:100015, 2021. ISSN 2192-4406. doi: https://doi.org/10.1016/j.ejco.2021.100015. URL https://www.sciencedirect.com/science/article/pii/S2192440621001428. arXiv:2101.00935.

[9] Pavel Dvurechensky, Kamil Safin, Shimrit Shtern, and Mathias Staudigl. Generalized self-concordant analysis of Frank-Wolfe algorithms. Mathematical Programming, 2022. doi: 10.1007/s10107-022-01771-1. URL https://doi.org/10.1007/s10107-022-01771-1.

[10] Khaid Elbassioni, Kazuhiro Makino, and Waleed Najy. Finding sparse solutions for packing and covering semidefinite programs. SIAM Journal on Optimization, 32(2):321–353, 2022. doi: 10.1137/20M137570X. URL https://doi.org/10.1137/20M137570X.

[11] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. Naval Research Logistics Quarterly, 3(1-2):95–110, 2019/05 1956. doi: 10.1002/nav.3800030109. URL https://doi.org/10.1002/nav.3800030109.

[12] Frank Göring, Christoph Helmberg, and Markus Wappler. Embedded in the shadow of the separator. SIAM Journal on Optimization, 19(1):472–501, 2008. doi: 10.1137/050639430. URL https://doi.org/10.1137/050639430.

[13] Elaine T. Hale, Wotao Yin, and Yin Zhang. Fixed-point continuation for $\ell_1$-minimization: Methodology and convergence. SIAM Journal on Optimization, 19(3):1107–1130, 2008. doi: 10.1137/070698920. URL https://doi.org/10.1137/070698920.

[14] Zaid Harchaoui, Anatoli Juditsky, and Arkadi Nemirovski. Conditional gradient algorithms for norm-regularized smooth convex optimization. Mathematical Programming, 152(1):75–112, 2015. doi: 10.1007/s10107-014-0778-9. URL https://doi.org/10.1007/s10107-014-0778-9.

[15] Garud Iyengar, David J Phillips, and Cliff Stein. Feasible and accurate algorithms for covering semidefinite programs. Scandinavian Workshop on Algorithm Theory, pages 150–162, 2010.

[16] Martin Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In International Conference on Machine Learning, pages 427–435, 2013.

[17] Guanghui Lan. First-order and stochastic optimization methods for machine learning. Springer, 2020.

[18] Gert RG Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan. Learning the kernel matrix with semidefinite programming. Journal of Machine learning research, 5(Jan):27–72, 2004.

[19] Monique Laurent and Franz Rendl. Semidefinite programming and integer programming. Handbooks in Operations Research and Management Science, 12:393–514, 2005.

[20] L. Lovasz. On the Shannon capacity of a graph. IEEE Transactions on Information Theory, 25(1):1–7, 1979. doi: 10.1109/TIT.1979.1055985.

[21] Yurii Nesterov. Lectures on Convex Optimization, volume 137 of Springer Optimization and Its Applications. Springer International Publishing, 2018.

[22] Yurii Nesterov and Arkadi Nemirovski. Interior Point Polynomial methods in Convex programming. SIAM Publications, 1994.

[23] Jiming Peng and Yu Wei. Approximating k-means-type clustering via semidefinite programming. SIAM Journal on Optimization, 18(1):186–205, 2007. doi: 10.1137/050641983. URL https://doi.org/10.1137/050641983.

[24] Benjamin Recht, Maryam Fazel, and Pablo A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. SIAM Review, 52(3):471–501, 2010. doi: 10.1137/070697835. URL https://doi.org/10.1137/070697835.

[25] Jun Sun, Stephen Boyd, Lin Xiao, and Persi Diaconis. The fastest mixing markov process on a graph and a connection to a maximum variance unfolding problem. SIAM Review, 48(4):681–699, 2022/01/12/ 2006. URL http://www.jstor.org/stable/2453871.
[26] Ryan J. Tibshirani and Jonathan Taylor. The solution path of the generalized lasso. *Ann. Statist.*, 39(3):1335–1371, 2011. doi: 10.1214/11-AOS878. URL https://projecteuclid.org:443/euclid.aos/1304514656.

[27] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2), 2009.

[28] Zaiwen Wen, Wotao Yin, Donald Goldfarb, and Yin Zhang. A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization, and continuation. *SIAM Journal on Scientific Computing*, 32(4):1832–1857, 2010. doi: 10.1137/090747695. URL https://doi.org/10.1137/090747695.

[29] EL Wilmer, David A Levin, and Yuval Peres. Markov chains and mixing times. *American Mathematical Soc.*, Providence, 2009.

[30] Lin Xiao and Tong Zhang. A proximal-gradient homotopy method for the sparse least-squares problem. *SIAM Journal on Optimization*, 23(2):1062–1091, 2013. doi: 10.1137/120869997. URL https://doi.org/10.1137/120869997.

[31] A. Yoshise and Y. Matsukawa. On optimization over the doubly nonnegative cone. In 2010 IEEE International Symposium on Computer-Aided Control System Design, pages 13–18, 2010. ISBN 2165-302X. doi: 10.1109/CACSD.2010.5612811.

[32] Alp Yurtsever, Joel A. Tropp, Olivier Fercoq, Madeleine Udell, and Volkan Cevher. Scalable semidefinite programming. *SIAM Journal on Mathematics of Data Science*, 3(1):171–200, 2021. doi: 10.1137/19M1305045. URL https://doi.org/10.1137/19M1305045.

[33] Renbo Zhao and Robert M. Freund. Analysis of the Frank–Wolfe method for convex composite optimization involving a logarithmically-homogeneous barrier. *Mathematical Programming*, 2022. doi: 10.1007/s10107-022-01820-9. URL https://doi.org/10.1007/s10107-022-01820-9.
| Dataset | Iter. | Alg. | $\eta_0 (x \cdot \Omega_g)$ | $\sigma$ | Obj. Value | Gap   | Time  |
|---------|-------|------|-----------------------------|--------|------------|-------|-------|
| 1       | 1000  | CG   | 0.01                        | 0.99   | 14.18      | 9.23e-02 | 57.33 |
|         |       | LCG  | 2                           | 0.99   | 14.67      | 6.07e-02 | 61.07 |
|         | 5000  | CG   | 1                           | 0.25   | 14.34      | 8.23e-02 | 299.84 |
|         |       | LCG  | 2                           | 0.99   | 14.99      | 4.08e-02 | 315.06 |
|         | 10000 | CG   | 2                           | 0.9    | 14.36      | 8.06e-02 | 590.61 |
|         |       | LCG  | 2                           | 0.99   | 15.12      | 3.22e-02 | 630.42 |
| 2       | 1000  | CG   | 0.01                        | 0.99   | 7.24       | 8.65e-02 | 59.50 |
|         |       | LCG  | 2                           | 0.99   | 7.4       | 6.65e-02 | 47.08 |
|         | 5000  | CG   | 1                           | 0.25   | 7.27      | 8.33e-02 | 297.11 |
|         |       | LCG  | 2                           | 0.99   | 7.54      | 4.89e-02 | 264.89 |
|         | 10000 | CG   | 1                           | 0.25   | 7.27      | 8.26e-02 | 589.45 |
|         |       | LCG  | 2                           | 0.99   | 7.6        | 4.12e-02 | 561.60 |
|         | 50000 | CG   | 2                           | 0.99   | 7.75      | 2.29e-02 | 3037.44 |
|         |       | LCG  | 2                           | 0.99   | 7.77      | 2.29e-02 | 3037.44 |
| 3       | 1000  | CG   | 0.01                        | 0.99   | 8.78       | 8.79e-01 | 125.78 |
|         |       | LCG  | 2                           | 0.99   | 67.95      | 6.04e-02 | 91.18 |
|         | 5000  | CG   | 0.01                        | 0.99   | 67.31      | 6.93e-02 | 717.50 |
|         |       | LCG  | 2                           | 0.99   | 69.68      | 3.65e-02 | 485.01 |
|         | 10000 | CG   | 0.01                        | 0.99   | 67.4        | 1.04e-01 | 1588.48 |
|         |       | LCG  | 2                           | 0.99   | 70.3        | 2.74e-02 | 1020.81 |
| 4       | 1000  | CG   | 0.01                        | 0.99   | 9.7          | 4.56e-01 | 170.83 |
|         |       | LCG  | 2                           | 0.99   | 16.37      | 8.26e-02 | 84.17 |
|         | 5000  | CG   | 0.01                        | 0.99   | 15.98      | 1.04e-01 | 822.57 |
|         |       | LCG  | 2                           | 0.99   | 16.82      | 5.72e-02 | 454.63 |
|         | 10000 | CG   | 0.01                        | 0.99   | 15.99      | 1.04e-01 | 1588.48 |
|         |       | LCG  | 2                           | 0.99   | 17.04      | 4.59e-02 | 940.77 |
| 5       | 1000  | CG   | 0.01                        | 0.99   | 130.86     | 6.63e-01 | 229.34 |
|         |       | LCG  | 2                           | 0.99   | 362.16     | 6.78e-02 | 160.07 |
|         | 5000  | CG   | 0.01                        | 0.99   | 356.87     | 8.15e-02 | 1020.93 |
|         |       | LCG  | 2                           | 0.99   | 373.61     | 3.84e-02 | 819.19 |
|         | 10000 | CG   | 0.01                        | 0.99  | 367.97     | 5.29e-02 | 2216.81 |
|         |       | LCG  | 2                           | 0.99   | 377.77     | 2.77e-02 | 1668.29 |
| 6       | 1000  | CG   | 0.01                        | 0.99   | 19.71      | 4.86e-01 | 211.59 |
|         |       | LCG  | 1                           | 0.99   | 35.29      | 8.03e-02 | 172.43 |
|         | 5000  | CG   | 0.01                        | 0.99   | 34.91      | 9.01e-02 | 1143.19 |
|         |       | LCG  | 1                           | 0.99   | 36.17      | 5.72e-02 | 942.42 |
|         | 10000 | CG   | 0.5                         | 0.25   | 35.03      | 8.69e-02 | 2469.83 |
|         |       | LCG  | 1                           | 0.99   | 36.57      | 4.67e-02 | 1904.46 |
| 7       | 1000  | CG   | 0.01                        | 0.99   | 89.68      | 7.31e-01 | 669.59 |
|         |       | LCG  | 2                           | 0.99   | 311.85     | 6.42e-02 | 441.59 |
|         | 5000  | CG   | 0.01                        | 0.99   | 263.38     | 2.10e-01 | 3589.64 |
|         |       | LCG  | 2                           | 0.99   | 320.39     | 3.86e-02 | 2231.72 |
|         | 10000 | CG   | 0.01                        | 0.99   | 312.09     | 6.35e-02 | 6995.37 |
|         |       | LCG  | 2                           | 0.99   | 323.44     | 2.95e-02 | 4661.37 |
| 8       | 1000  | CG   | 0.01                        | 0.99   | 10.91      | 8.64e-01 | 667.97 |
|         |       | LCG  | 1                           | 0.99   | 72.62      | 9.25e-02 | 427.37 |
|         | 5000  | CG   | 0.01                        | 0.99   | 49.59      | 3.80e-01 | 3859.94 |
|         |       | LCG  | 2                           | 0.99   | 75.02      | 6.26e-02 | 2262.00 |
|         | 10000 | CG   | 0.01                        | 0.99   | 71.97      | 1.01e-01 | 7479.36 |
|         |       | LCG  | 2                           | 0.99   | 76.04      | 4.98e-02 | 4792.46 |
Figure 7: Mixing datasets relative gap from SDPT3 solution vs. iteration for various parameter choices.
Figure 8: Mixing datasets relative gap from SDPT3 solution vs. time for various parameter choices.