Random projections for semidefinite programming *

Leo Liberti, Benedetto Manca, Antoine Oustry, Pierre-Louis Poirion

Accepted in: P. Cappanera et al (eds), Proceedings of AIRO-ODS 2022, Springer, Cham.

Random projections for semidefinite programming*

Leo Liberti, Benedetto Manca, Antoine Oustry, and Pierre-Louis Poirion

Abstract Random projections can reduce the dimensionality of point sets while keeping approximate congruence. Applying random projections to optimization problems raises many theoretical and computational issues. Most of the theoretical issues in the application of random projections to conic programming were addressed in [1]. This paper focuses on semidefinite programming.

Key words: Johnson-Lindenstrauss Lemma, conic programming, QCQP

1 Introduction

Semidefinite Programming (SDP) problems are linear optimization problems with a matrix variable X over the cone $X \succeq 0$ of positive semidefinite (PSD) matrices:

$$\min\{\langle C,X\rangle \mid \forall i \leq m \ \langle A_i,X\rangle = b_i \wedge X \succeq 0\}, \tag{1}$$

where C,A_i (for $i \leq m$) are given $n \times n$ symmetric matrices, b is a given vector in $\mathbb{R}^m$, and $\langle M,N\rangle = \mathsf{tr}(M^\top N)$ is the Frobenius inner product.

Leo Liberti and Antoine Oustry
LIX CNRS Ecole Polytechnique, Institut Polytechnique de Paris, 91128 Palaiseau, France
e-mail: liberti@lix.polytechnique.fr, antoine.oustry@polytechnique.edu

Benedetto Manca
Dip. Matematica and Informatica, Università degli Studi di Cagliari, Via Ospedale 72, 09124 Cagliari, Italy
e-mail: bmanca@unica.it

Pierre-Louis Poirion
RIKEN Center for Advanced Intelligence Project, Tokyo, Japan
e-mail: pierre-louis.poirion@riken.jp

* The second author (BM) was partly supported by grant STAGE, Fondazione Sardegna 2018.

A *Random Projection* (RP) is a $k \times m$ matrix T, with $k \leq m$ and density $\sigma = |\text{nonzeros}|/(km)$, every component of which is sampled from a normal distribution $\mathsf{Normal}(0, 1/\sqrt{\sigma k})$ [2, §5.1]. RPs are interesting because of the Johnson-Lindenstrauss Lemma (JLL).

Lemma 1 ([3])

Given a finite set of vectors $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^m$ and an $\varepsilon \in (0,1)$, there exist $k = O(\varepsilon^{-2} \ln n) \in \mathbb{N}$, an affine function $\phi(k)$, and constants $\mathscr{C}, \mathscr{C}_2 > 0$ such that:

$$\mathbb{P}\left[\forall i < j \leq n \, (1-\varepsilon) \leq \frac{\|Tx_i - Tx_j\|_2}{\|x_i - x_j\|_2} \leq (1+\varepsilon)\right] \geq 1 - \mathscr{C}_2 e^{-\mathscr{C}\phi(k)}. \qquad (2)$$

Instead of applying RPs to point sets, in this paper we apply them to Mathematical Programming (MP) formulations (also see [4, 5, 2, 1, 6, 7]). More precisely, when RPs are applied to the input data, they yield different types of *projected MP formulations*. For each such type, one must prove that the JLL implies approximate feasibility and approximate optimality, which is nontrivial: for example, decision variables in formulations may well represent uncountable sets of vectors, while the JLL only applies to finite sets. Moreover, the *projected solution* (i.e. a solution of the projected MP) is generally infeasible in the original MP. It was shown in [5] that Linear Programming (LP) $\min\{cx \mid Ax = b \wedge x \geq 0\}$ yields a projected formulation $\min\{cx \mid TAx = Tb \wedge x \geq 0\}$ with feasibility and optimality guarantees in probability. The results about LP were extended to other Conic Programming (CP) problems in [1], which also proposes a *solution retrieval* method for constructing an almost feasible original problem solution from the projected solution.

The present paper focuses on SDP. Since SDPs are CPs, all of the results in [1] apply. The new contributions of this paper build on the theoretical results of [1], and contribute a new computational study. We improve some error bounds from [1] (Prop. 1 and its corollaries); we propose a new solution retrieval algorithm; we obtain computational results on random as well as structured instance.

The rest of this paper is organized as follows. Sect. 2 discusses some of the most interesting properties of RPs. Sect. 3 motivates and introduces the projected SDP formulation. In Sect. 4 we recall the theory underlying the application of RPs to SDPs [1]. Sect. 5 presents a computational study on random instances, SDP relaxations of the Distance Geometry Problem (DGP), and SDP relaxations of the Alternating Current Optimal Power Flow (ACOPF) problem. We find that RPs yield useful SDP approximations.

2 Understanding random projections

The JLL shows that the RP T guarantees approximate congruence between X and TX with arbitrarily high probability (wahp). This is surprising because we think of congruence as a dimension-preserving property (e.g. rotations, translations). Allowing an ε error, however, affords approximate congruence. Even so, this RP feature

appears mysterious because it is only realized in high dimensional spaces, and hence it is impossible to "visualize". This is a consequence of the fact that $k = O(\varepsilon^{-2} \ln n)$: if we want ε to represent a low enough error, say 10%, this requires $k \geq \varepsilon^{-2} = 100$ (assuming the $O(\cdot)$ coefficient to be ≥ 1 [10]), meaning that the original space dimension m must be larger than 100.

Note that the target dimension k is independent of the original dimension m, and only depends on n logarithmically. In other words, up to an ε error, the "natural" choice of the data dimensionality m (e.g. the number of pixels in an image) is immaterial: the "correct" dimension only depends on the number of image vectors being projected. This challenges our comprehension: after all, the reason why the original dimension is termed "natural" is that there is a natural, real-life interpretation of m as the most appropriate dimension choice. Another surprising consequence of the JLL (obtained by re-casting the JLL in terms of angles instead of distances) is that $\mathbb{R}^m$ can host exponentially many "almost orthogonal vectors" [8, §7.3.1(7)].

A different result, that of *distance instability* [9], says that if $Z, X_1, \ldots, X_n$ are random vectors in $\mathbb{R}^m$ sampled independently from any "reasonable" and "well-behaved" distribution (check [9] for details), then for any $\varepsilon > 0$ we have

$$\max_{j \leq n} \|Z - X_j\|_2 \leq (1+\varepsilon) \min_{j \leq n} \|Z - X_j\|_2$$

as $m \to \infty$: in short, closest neighbors to Z are indistinguishable (up to ε) from any other point. Intuitively, this suggests that all Euclidean distances are more or less the same in high dimensions, and so the ε-approximate congruence guaranteed by the JLL would not be all that surprising. But the proof settings of the JLL and of the distance instability theorem are very different: the JLL applies to a given finite set of vectors, while distance instability applies to distributions.

Numerous computational tests on clustering images with the k-means algorithm (carried out by one of the authors of this papers) show that the clusterings obtained on original and projected image vectors are similar. The tests exhibited in many JLL-related papers tell a similar story (see e.g. [10]). So, distance instability notwithstanding, and despite its apparent weirdness, the JLL definitely offers a computational value, which we exploit in this paper by applying it to SDP formulations.

3 Projected SDP: motivation and formulation

SDPs are important because they are routinely used as convex relaxations of Quadratic Programs (QP) [2], Quadratically Constrained Programs (QCP) [8, §6.1.3], and Quadratically Constrained Quadratic Programs (QCQP) [11, §4.4]. While convex programming is not necessarily tractable (e.g. MAX CLIQUE, through the Motzkin-Straus formulation, reduces to linear optimization over the completely positive cone [12]), we can solve SDPs to any desired accuracy in polynomial time using the interior point method [13]. Because SDP relaxations are generally tighter than LP relaxations obtained using standard techniques such as McCormick inequal-

ities [14], there is an expectation that their use within spatial Branch-and-Bound (sBB) algorithms (e.g. [15]) is beneficial. Unfortunately, however, SDP solvers are much slower than their LP counterparts, so SDP use in sBB is not mainstream. Because of their remarkable size decrease, resulting in proportional CPU time savings during the solution phase, projected SDP formulations might partly address this issue.

In order to define a projected SDP, we proceed as follows. We note that, for $n \times n$ matrices M,N, $\langle M,N\rangle = \mathsf{vec}(M)^\top \mathsf{vec}(N)$, where, for any M, $\mathsf{vec}(M)$ is the vector in $\mathbb{R}^{n^2}$ constructed by stacking the columns of M. We can therefore re-write $\langle A_i,X\rangle = b_i$ as $\mathsf{vec}(A_i)^\top \mathsf{vec}(X) = b_i$ for each $i \leq m$. Let A be the $m \times n^2$ matrix obtained by stacking the m rows $\mathsf{vec}(A_i)^\top$. We define $A \odot X$ to be the vector $(\mathsf{vec}(A_i)^\top \mathsf{vec}(X) \mid i \leq m) \in \mathbb{R}^m$. Next, we can reformulate Eq. (1) as follows:

$$\min\{\langle C,X\rangle \mid A \odot X = b \wedge X \succeq 0\}. \quad (\mathsf{P}) \tag{3}$$

A projected SDP formulation based on a $k \times m$ RP T, where $k = O(\varepsilon^{-2}\ln n)$, can finally be derived as follows:

$$\min\{\langle C,X\rangle \mid TA \odot X = Tb \wedge X \succeq 0\}. \quad (T\mathsf{P}) \tag{4}$$

4 Theory of projected SDP

The solution $\bar{X}$ of $T\mathsf{P}$ has some interesting approximation properties w.r.t. the original SDP P, under a boundedness condition $\langle \mathbf{1},X\rangle \leq \theta$ for some appropriate $\theta > 0$.

We note that the dual SDP to P is

$$\max\{yb \mid yA \preceq C\}, \quad (\mathsf{D}) \tag{5}$$

where $yA = \sum_{i \leq m} y_i A_i$, and $yA \preceq C$ means that $C - yA$ is PSD.

Theorem 1 (Approximate feasibility)

Let y^* be the optimal solution of D, and $\|A\|_\dagger = \sup_{\|z\|_1=1} \|zA\|_F$. Then there are constants $\mathscr{C},\mathscr{C}_2 > 0$ such that, for $0 < \varepsilon < \|y^*\|_2(\|b\|_2 + \|A\|_\dagger)^{-1}$ and $k = O(\varepsilon^{-2}\ln n)$, we have

$$\mathbb{P}[\mathsf{P} \textit{ is feasible} \Leftrightarrow T\mathsf{P} \textit{ is feasible}] \geq 1 - 2(m+1)\mathscr{C}_2 e^{-\mathscr{C}\varepsilon^2 k}.$$

Proving that $T\mathsf{P}$ is feasible whenever P is feasible simply follows by linearity of T. The probabilistic statement only refers to proving that $T\mathsf{P}$ is infeasible wahp whenever P is infeasible [1, Thm. 3.2].

Theorem 2 (Approximate optimality)

With the above notation, and assuming strong SDP duality holds,

$$\mathbb{P}[\mathsf{val}(T\mathsf{P}) \leq \mathsf{val}(\mathsf{P}) \leq \mathsf{val}(T\mathsf{P}) + \varepsilon\|y^*\|_2(\|b\|_2 + \|A\|_\dagger\theta)] \geq 1-(2m+1)\mathscr{C}_2 e^{-\mathscr{C}\varepsilon^2 k}.$$

Again, the proof has an easy part (proving that $T\mathsf{P}$ is a relaxation of P, which simply follows because aggregating equality constraints always produces a relaxation), and a difficult part [1, Thm. 3.5].

Theorems 1-2 are unsatisfying insofar as the appropriate choice of ε, necessary to define k and therefore to formulate $T\mathsf{P}$, depends on the norm of the optimal solution of D. Of course, if one were to solve D, the solution of $T\mathsf{P}$ to speed up that of P would become a moot point (unfortunately, this is a feature of many theoretical results in RPs applied to MP). What these results really say is that solving projected formulations is likely to provide good approximations, although the choice of parameters requires considerable guesswork.

The projected solution $\bar{X}$ of $T\mathsf{P}$, which satisfies $TA \odot X = Tb$, has probability zero of being feasible in $A \odot X = b$ because the rank of (TA, b) is k, which is smaller than the rank m of (A, b). On the other hand, [1, Prop. 4.2] gives an upper bound to the infeasibility error: for any $u > 0$ we have

$$\mathbb{P}\left[\|A \odot \bar{X} - b\|_2 \leq \varepsilon\theta\|A\|_2(\mathscr{C}_3\mathsf{w}(\mathbb{B}) + u\Delta(\mathbb{B}))/\sqrt{\ln(n)}\right] \geq 1 - 2e^{-u^2}, \quad (6)$$

where $\mathscr{C}_3$ is a universal constant, $\mathbb{B} = \{X \succeq 0 \mid \mathsf{tr}(X) \leq 1\}$, $\mathsf{w}(\mathbb{B})$ is the Gaussian width, and $\Delta(\mathbb{B})$ is the diameter of $\mathbb{B}$.

We now estimate the Gaussian width and diameter of $\mathbb{B}$ in terms of n in order to improve the bound in Eq. (6) specifically for SDPs.

Proposition 1 *(i)* $\mathsf{w}(\mathbb{B}) \leq \sqrt{2n}$*; (ii)* $\Delta(\mathbb{B}) \leq 2$.

Proof About (i), by [16, Thm. 1], we have $\mathsf{tr}(AB) \leq \lambda_{\max}(A)\mathsf{tr}(B)$. Thus,

$$\begin{aligned}
\mathsf{w}(\mathbb{B}) &= \mathbb{E}_G(\sup_{X\in\mathbb{B}}\langle G, X\rangle) \quad \text{by definition of Gaussian width} \\
&\leq \mathbb{E}_G(\lambda_{\max}(G)\sup_{X\in\mathbb{B}}\mathsf{tr}(X)) \quad \text{since } G \text{ does not depend on } X \\
&= \mathbb{E}_G(\lambda_{\max}(G)) \quad \text{because } \sup_{X\in\mathbb{B}}\mathsf{tr}(X) = 1.
\end{aligned}$$

Now, by [17, Lemma 2.8], for a symmetric $n \times n$ random matrix variable G distributed like $\mathsf{Normal}(0,1)^{n(n+1)/2}$, we have $\mathbb{E}_G(\lambda_{\max}(G)) \leq \sqrt{2n} \Rightarrow \mathsf{w} \leq \sqrt{2n}$ as claimed. As for (ii), by definition of $\mathbb{B}$ we have:

$$\Delta(\mathbb{B}) = \sup_{X,Y\in\mathbb{B}}\|X-Y\|_F \leq 2\sup_{X\in\mathbb{B}}\|X\|_F \leq 2\sup_{X\in\mathbb{B}}\|X\|_1 = 2\sup_{X\in\mathbb{B}}\mathsf{tr}(X) = 2. \quad \square$$

Corollary 1 *For A, b as defined above, and $\bar{X}$ the solution of $T\mathsf{P}$, we have:*

$$\forall u > 0 \quad \mathbb{P}\left[\|A \odot \bar{X} - b\|_2 \leq \varepsilon\theta\|A\|_2(\mathscr{C}_3\sqrt{2n} + 2u)/\sqrt{\ln(n)}\right] \geq 1 - 2e^{-u^2}. \quad (7)$$

Proof By application of Prop. 1 to Eq. (6). $\square$

The solution retrieval method proposed in [1] consists in constructing a solution $\tilde{X}$ which is a projection of $\bar{X}$ onto $A \odot X = b$:

$$\tilde{X} = \bar{X} + A^\top (AA^\top)^{-1}(b - A \odot \bar{X}). \tag{8}$$

This may easily cause $\tilde{X} \not\succeq 0$, but [1, Thm. 4.4] and Prop. 1 show that this "negativity error" is bounded.

Corollary 2 *With $A, b, \bar{X}, \tilde{X}$ as above and $\kappa(A)$ the condition number of A, we have:*

$$\forall u > 0 \quad \mathbb{P}\left[\lambda_{\min}(\tilde{X}) \geq \lambda_{\min}(\bar{X}) - \varepsilon\theta\kappa(A)(\mathscr{C}_3\sqrt{2n} + 2u)/\sqrt{\ln n}\right] \geq 1 - 2e^{-u^2}.$$

4.1 A new solution retrieval method

The issue with the solution retrieval method in Eq. (8) is that it yields a solution $\tilde{X}$ which has some (hopefully small) eigenvalue negativity, which stems from a projection of $\bar{X}$ onto $A \odot X = b$. We note that this eigenvalue negativity can be "projected away" by zeroing all of the negative eigenvalues of $\tilde{X}$, a technique known as classic Multidimensional Scaling (MDS) [18], which essentially yields a projection of $\tilde{X}$ on the PSD cone $X \succeq 0$. Every time we project on $A \odot X = b$ we may leave the PSD cone, and every time we project back into the PSD cone we may leave the subspace $A \odot X = b$. This suggests using the well-known Alternating Projection Method (APM) [19, Thm. 13.10] based on the two convex sets $A \odot X = b$ and $X \succeq 0$. For two closed convex sets S_1, S_2 with non-empty intersection, the APM converges (possibly in infinite time) to a point in the intersection $S_1 \cap S_2$. The new retrieval method is presented in Alg. 1. The loop in Alg. 1 is executed at most a given number

Algorithm 1 APM$(A, b, \tilde{X}, \delta)$

```
X̂ ← X̄
for i ≤ MaxIterations do
    X̂ ← X̂ + Aᵀ(AAᵀ)⁻¹(b − A ⊙ X̂)
    if X ⪰ 0 then
        return X̂    // X̂ feasible
    end if
    X̂ ← MDS(X̂)    // perform Multidimensional Scaling on X̂
    if ‖A ⊙ X̂ − b‖₂ ≤ δ then
        return X̂    // X̂ almost feasible
    end if
end for
return X̂    // X̂ is closer than X̃ to being feasible
```

MaxIterations of times. It alternates between achieving feasibility w.r.t. $A \odot X = b$ and w.r.t. $X \succeq 0$. Alg. 1 returns a solution $\hat{X} \leftarrow \text{APM}(A, b, \hat{X}, \delta)$ which is closer to the feasible set of P than the initial matrix $\tilde{X}$ was.

5 Computational results

The Mosek 9.1.5 [20] SDP solver was used in all experiments, carried out on a 2.1GHz Intel Xeon E5-2620 with 32 8-core CPUs and 64GB RAM running Linux. Most code was written in Python 3. The number of iterations for the APM Alg. 1 was set to 50 for DGP instances (Sect. 5.2) and 20 otherwise.

A computational validation of projected SDPs was carried out in [1, §5]. RPs were sampled as in [21]. Fourty random SDPs were generated, ten of which infeasible and thirty feasible. In all cases, $C = I_n$ and $A \sim \mathsf{Uniform}(0,1)^{m\times d}$, where $d = n(n+1)/2$ are the degrees of freedom of the linear system. In infeasible cases, $b \sim \mathsf{Uniform}(0,1)^m$ (all candidate infeasible instances were verified to be infeasible). In feasible cases, $X_0 \sim \mathsf{Uniform}(-1,1)^{n\times n}$ so that X was diagonally dominant (DD) and hence PSD, and $b = A \odot X$. The actual SDP formulation tested in infeasible cases was $\min\{\mathsf{tr}(X) \mid A \odot X = b \wedge X_0 - \theta\mathbf{1} \leq X \leq X_0 + \theta\mathbf{1} \wedge X \succeq 0\}$, to make sure instances were not unbounded, and take the θ bound into account explicitly. Infeasible tests were all successful with $\varepsilon = 0.13$ (every infeasible P mapped to an infeasible $T\mathsf{P}$), and all unsuccessful with $\varepsilon = 0.2$. Feasible tests yielded very poor values of $\mathsf{val}(T\mathsf{P})$. The retrieved solutions had excellent quality, although we later discovered a bias in the random generation process.

5.1 New tests on random instances

The differences of the new benchmark w.r.t. [1] are as follows. (a) We only test feasible instances. (b) We use $k \times m$ sparse RPs T with given density σ, sampled componentwise from $\mathsf{Normal}(0, 1/\sqrt{\sigma k})$. (c) Instead of taking $C = I_n$, we sample $C \sim \mathsf{Uniform}(0,1)^{n^2}$. (d) We do not artificially change the instance with X_0 and θ as in [1], since neither are actually available in practice. Instead, we impose a doubly non-negative (DNN) constraint on X, yielding

$$\min\{\langle C, X\rangle \mid A \odot X = b \wedge X \geq 0 \wedge X \succeq 0\}. \tag{9}$$

We test SDPs with different densities $\sigma \in \{0.05, 0.1, 0.4\}$ in the constraint matrix A (sampled as in [1]), and sample the RP T with the same density. We chose instance sizes $(m,n) \in \{(1000,50),(1500,57),(2000,65),(2500,72),(3000,80)\}$, yielding degrees of freedom $d \in \{1275, 1653, 2145, 2628, 3240\}$; we fixed $k = 0.001\,m$ for the whole benchmark. We tested 6 random instances per size and density. Average results are given in Table 1. RPs appear to provide best results on sparse instances. Given that projection on $A \odot X = b$ very often preserved $X \succeq 0$, the solution retrieval method was adapted to only alternate between $A \odot X = b$ and $X \geq 0$, terminating on the former, and limited to 20 iterations to prevent excessive CPU time usage, but clearly more iterations would have further decreased the error w.r.t. $X \geq 0$. The objective values follow a trend to [1]: $\mathsf{val}(T\mathsf{P})$ provides a poor relaxation, but $\langle C, \tilde{X}\rangle$ is a good approximation of $\mathsf{val}(\mathsf{P})$.

Instance					Objective		Feasibility			CPU
σ	m	k	n	d	$\bar{F}/F^*$	$\tilde{F}/F^*$	$\frac{\|A\odot\tilde{X}-b\|_1}{m}$	$\frac{\sum\tilde{X}_{ij}^-}{n^2}$	$\lambda_{\min}(\tilde{X})$	$\tilde{t}/t^*$
0.05	1000	10	50	1275	0.0614	0.8131	0.000	0.640	0.668	0.73
0.05	1500	15	57	1653	0.0621	0.8748	0.001	0.446	0.002	0.70
0.05	2000	20	65	2145	0.0882	0.9356	0.001	0.397	0.000	0.70
0.05	2500	25	72	2628	0.0837	0.9458	0.001	0.391	0.000	0.79
0.05	3000	30	80	3240	0.0784	0.9396	0.001	0.529	0.000	0.71
0.10	1000	10	50	1275	0.0520	0.7676	0.000	0.618	0.570	0.69
0.10	1500	15	57	1653	0.0740	0.9425	0.001	0.475	0.001	0.79
0.10	2000	20	65	2145	0.0795	0.9278	0.001	0.457	0.000	0.83
0.10	2500	25	72	2628	0.0721	0.9381	0.001	0.361	0.000	0.85
0.10	3000	30	80	3240	0.0879	0.9217	0.001	0.521	0.000	0.85
0.40	1000	10	50	1275	0.0542	0.7702	0.001	0.591	0.385	0.92
0.40	1500	15	57	1653	0.0736	0.9155	0.002	0.427	0.000	1.01
0.40	2000	20	65	2145	0.0707	0.9341	0.002	0.417	0.000	1.12
0.40	2500	25	72	2628	0.0866	0.9549	0.003	0.367	0.000	1.06
0.40	3000	30	80	3240	0.0963	0.8953	0.003	0.491	0.000	0.95

Table 1 Computational results on random SDPs. $F^*, \bar{F}, \tilde{F}$ are the objective function values of the original, projected, retrieved ($\tilde{X}$) solution; $\tilde{t}$ is the CPU time taken to sample T, compute TA, Tb, construct and solve $T\mathsf{P}$, retrieve $\tilde{X}$; t^* is the time taken to solve P. Best objective ratios should approach 1; best CPU time ratios should approach 0.

5.2 Tests on SDP relaxations of the DGP

The DGP is the following decision problem: given an integer $K > 0$ and a simple edge-weighted graph $G = (V, E, d)$, decide whether there is a realization $x : V \to \mathbb{R}^K$ such that $\forall \{i, j\} \in E \ \|x_i - x_j\|_2^2 = d_{ij}^2$. This is a pure-feasibility, **NP**-hard nonconvex QCP [22]. Its SDP relaxation is based on the identity $\|x_i - x_j\|_2^2 = \langle x_i, x_i\rangle + \langle x_j, x_j\rangle - 2\langle x_i, x_j\rangle$. By replacement of the $\langle x_i, x_j\rangle$ with new variables X_{ij}, we have the SDP: $\min\{\mathsf{tr}(X) \mid \forall \{i, j\} \in E \ (X_{ii} + X_{jj} - 2X_{ij} = d_{ij}^2) \wedge X \succeq 0\}$ (†). The objective function aims at (heuristically) reducing the rank of X, since, by spectral decomposition, $\mathsf{tr}(X) = \mathsf{tr}(P\Lambda P^\top) = \mathsf{tr}(\Lambda PP^\top) = \mathsf{tr}(\Lambda) = \sum_{v \in V} \lambda_v$. By minimizing the sum of (non-negative) eigenvalues of X, we hope that at least some of them will be set to zero.

The tested instances were taken from the Protein Data Bank (PDB): each realization was transformed into a distance-weighted ball graph with radius 5.5Å, as discussed in [23, 3.1]. The resulting linear system $A \odot X = b$ in (†) is very sparse and loosely constrained. We chose $k = 0.01\,m$ for this benchmark. The results in Table 2 show a good trend in the CPU times, at the expense of the sizable approximation errors (optimality and feasibility w.r.t. the PSD cone). Despite the errors, further tests (not included in this paper) show that good quality solutions of the DGP can nonetheless be obtained by using $\tilde{X}$, after dimensionality reduction (which loses the negative eigenvalues), as a starting point for a local descent using a nonlinear optimization solver on a standard primal DGP formulation.

	Instance					Objective		Feasibility		CPU
name	σ	m	k	n	d	$\bar{F}/F^*$	$\tilde{F}/F^*$	$\frac{\|A\odot\tilde{X}-b\|_1}{m}$	$\lambda_{\min}(\tilde{X})$	$\tilde{t}/t^*$
`tiny`	0.00540	335	58	38	741	0.0190	0.4999	0.000	0.730	0.62
`names`	0.00104	849	67	87	3828	0.0122	0.2182	0.000	1.007	0.45
`1guu`	0.00004	955	69	427	91378	0.0000	0.1290	0.000	0.203	0.23
`1guu-1`	0.00035	959	69	150	11325	0.0025	0.1152	0.000	0.574	0.42
`2kxa`	0.00007	2711	79	333	55611	0.0020	0.1718	0.000	0.765	0.22
`100d`	0.00003	5741	87	491	120786	0.0013	0.0692	0.000	1.217	0.26

Table 2 Computational results on SDPs derived from DGP instances.

5.3 Tests on SDP relaxations of the ACOPF

The ACOPF is a QCQP with parameters and decision variables over $\mathbb{C}$ [11], which can be reformulated to a QCQP over $\mathbb{R}$ four times the size. It relies on Ohm's and Kirchhoff's laws adapted to AC, plus some technical constraints. We test an ACOPF variant of the form $\min\{\langle C,X\rangle \mid A\odot X = b \wedge L \leq X \leq U \wedge X \succeq 0\}$ that minimizes $\|X - X_0\|_\infty$ with fixed generating power levels, where X_0 is a target solution. See [24] about the tested instances. We chose $k = 0.001\,m$ for this benchmark.

	Instance					Objective		Feasibility			CPU
name	σ	m	k	n	d	$\bar{F}/F^*$	$\tilde{F}/F^*$	$\frac{\|A\odot\tilde{X}-b\|_1}{m}$	`rng`	$\lambda_{\min}(\tilde{X})$	$\tilde{t}/t^*$
`case57_ieee`	0.00366	3363	3	114	6555	0.0002	0.9152	0.000	0.007	0.092	0.74
`case73_ieee`	0.00205	5475	5	146	10731	0.0028	0.8228	0.000	0.018	0.094	0.58
`case89_pegase`	0.00377	8099	8	178	15931	0.0001	0.0147	0.000	0.281	0.022	0.14
`case118_ieee`	0.00136	14160	14	236	27966	0.0006	0.8456	0.000	0.010	0.092	0.35
`case162_ieee`	0.00072	26568	27	324	52650	0.0015	0.8868	0.000	0.012	0.099	0.32
`case179_goc`	0.00059	32399	32	358	64261	0.0010	0.0632	0.000	0.261	0.017	0.28

Table 3 Tests on SDPs derived from ACOPF instances; `rng` is the average range error.

The results in Table 3 show good trends in optimality and feasibility (especially in the `ieee` instances), as well as CPU time (across the benchmark).

5.4 Closing remarks

Although $\tilde{X}$ is supposed to be approximately feasible, the ratios $\tilde{F}/F^*$ in the results tables are ≤ 1 because $\tilde{X}$ is obtained by applying a heuristic (Sect. 4.1) to the solution $\bar{X}$ of an SDP relaxation. That the ratios $\tilde{t}/t^*$ are sometimes ≥ 1 does not mean that RPs are useless: the point of RPs is to solve instances so large they cannot be solved any other way. Here we are interested in comparing the performances, so instances are small enough so we can also solve them exactly. But the trend given by $k = O(\ln n)$ is that $\tilde{t}/t^*$ decrease with increasing n.

References

1. L. Liberti, P.-L. Poirion, and K. Vu. Random projections for conic programs. *Linear Algebra and its Applications*, 626:204–220, 2021.
2. C. D'Ambrosio, L. Liberti, P.-L. Poirion, and K. Vu. Random projections for quadratic programs. *Mathematical Programming B*, 183:619–647, 2020.
3. W. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In G. Hedlund, editor, *Conference in Modern Analysis and Probability*, volume 26 of *Contemporary Mathematics*, pages 189–206, Providence, RI, 1984. AMS.
4. M. Pilanci and M. Wainwright. Randomized sketches of convex programs with sharp guarantees. *IEEE Transactions on Information Theory*, 61(9):5096–5115, 2015.
5. K. Vu, P.-L. Poirion, and L. Liberti. Random projections for linear programming. *Mathematics of Operations Research*, 43(4):1051–1071, 2018.
6. L. Liberti and B. Manca. Side-constrained minimum sum-of-squares clustering: Mathematical Programming and random projections. *Journal of Global Optimization*, accepted.
7. C. Cartis, E. Massart, and A. Otemissov. Global optimization using random embeddings. Technical Report 2107.12102, arXiv, 2021.
8. L. Liberti. Distance geometry and data science. *TOP*, 28:271–339, 220.
9. K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In C. Beeri and P. Buneman, editors, *Proceedings of ICDT*, volume 1540 of *LNCS*, pages 217–235, Heidelberg, 1998. Springer.
10. S. Venkatasubramanian and Q. Wang. The Johnson-Lindenstrauss transform: An empirical study. In *Algorithm Engineering and Experiments*, volume 13 of *ALENEX*, pages 164–173, Providence, RI, 2011. SIAM.
11. D. Bienstock, M. Escobar, C. Gentile, and L. Liberti. Mathematical programming formulations for the alternating current optimal power flow problem. *4OR*, 18(3):249–292, 2020.
12. I. Bomze, M. Dür, E. De Klerk, C. Roos, A. Quist, and T. Terlaky. On copositive programming and standard quadratic optimization problems. *Journal of Global Optimization*, 18:301–320, 2000.
13. C. Helmberg, F. Rendl, R. Vanderbei, and H. Wolkowicz. An interior-point method for semidefinite programming. *SIAM Journal on Optimization*, 6(2):342–361, 1996.
14. K. Anstreicher. Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming. *Journal of Global Optimization*, 43:471–484, 2009.
15. P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24(4):597–634, 2009.
16. Y. Fang, K. Loparo, and X. Feng. Inequalities for the trace of matrix product. *IEEE Transactions on Automatic Control*, 39(12):2489–2490, 1994.
17. D. Song and P. Parrilo. On approximations of the psd cone by a polynomial number of smaller-sized psd cones. Technical Report 2105.02080v1, arXiv, 2021.
18. T. Cox and M. Cox. *Multidimensional Scaling*. Chapman & Hall, Boca Raton, 2001.
19. J. von Neumann. *Functional Operators. Volume II: The geometry of orthogonal spaces*. Number 22 in Annals of Mathematics Studies. Princeton University Press, Princeton NJ, 1950.
20. Mosek ApS. *The `mosek` manual, Version 9*, 2019.
21. D. Kane and J. Nelson. Sparser Johnson-Lindenstrauss transforms. *Journal of the ACM*, 61(1):4, 2014.
22. J. Saxe. Embeddability of weighted graphs in k-space is strongly NP-hard. *Proceedings of 17th Allerton Conference in Communications, Control and Computing*, pages 480–489, 1979.
23. L. Liberti, C. Lavor, N. Maculan, and A. Mucherino. Euclidean distance geometry and applications. *SIAM Review*, 56(1):3–69, 2014.
24. IEEE PES PGLib-OPF Task Force. The power grid library for benchmarking ac optimal power flow algorithms. Technical Report 1908.02788, arXiv, 2019.