A Cutting-plane Method for Semidefinite Programming with Potential Applications on Noisy Quantum Devices

J. Mareček and A. Akhriev

1Department of Computer Science, Czech Technical University, Karlovo nam. 13, Prague 2, the Czech Republic
2IBM Quantum, IBM Research – Europe, IBM Dublin Technology Campus, Mulhuddart Dublin 15, Ireland

There is an increasing interest in quantum algorithms for optimization problems. Within convex optimization, interior-point methods and other recently proposed quantum algorithms are non-trivial to implement on noisy quantum devices. Here, we discuss how to utilize an alternative approach to convex optimization, in general, and semidefinite programming (SDP), in particular. This approach is based on a randomized variant of the cutting-plane method. We show how to leverage quantum speed-up of an eigensolver in speeding up an SDP solver utilizing the cutting-plane method. For the first time, we demonstrate a practical implementation of a randomized variant of the cutting-plane method for semidefinite programming on instances from SDPLIB, a well-known benchmark. Furthermore, we show that the RCP method is very robust to noise in the boundary oracle, which may make RCP suitable for use even on noisy quantum devices.

I. INTRODUCTION

Considering that gate-based quantum computers are expected to aid in solving specific optimization problems across many domains, including quantum chemistry [14], machine learning [15], and computational finance [16], it may seem natural to seek quantum algorithms for convex optimization. Quantum speedup in convex optimization seems elusive, in general. Garg et al. [17] have shown that in optimizing a Lipschitz-continuous, but otherwise arbitrary convex function over the unit ball, first-order methods [17] have no quantum speedup over gradient descent, when restricted to the black-box access to the values and gradients of the convex function. One should hence consider other special cases, preferably as broad as possible, and possibly avoiding the black-box access.

Semidefinite programming (SDP) is a broad special case of convex optimization, which has attracted a substantial recent interest. Initially, [18–21] “quantized” the so-called multiplicative-weight-update (MWU) algorithm of Arora and Kale [22] and its variants by Hazan [23]. Subsequently, [25–26] attempted a translation of primal-dual interior-point methods [27] to quantum computers. Finally, in [29–31], the authors study the relationship of several oracles useful in first-order algorithms, but do not claim a run-time of a particular algorithm for SDPs. These results are summarized in Table I. While some of the quantum algorithms [21] are reported as scaling with $O(\sqrt{n} \log(m) \log(n))$ for $m$ constraints in $n \times n$ matrices, this requires the diameter of the convex set to be independent of the dimension, while the dependence is quadratic, in general. Furthermore, none of these algorithms have been implemented in an actual quantum device, or its simulator.

Here, we consider another method for solving SDPs, which can be run in part on the quantum computer. In particular, we “quantize” the so-called randomized cutting plane (RCP) method. The cutting-plane methods [33] have produced a variety of classical theoretical guarantees, as surveyed in Table I, including the first polynomial-time algorithm for linear programming, but yielded little in terms of practical implementations in classical computers. This is because a certain subroutine, known as the boundary oracle, is classically almost as demanding as demanding as the original problem. We show how to leverage quantum speed-up of an eigensolver in speeding up the RCP method. Furthermore, we demonstrate an implementation of the RCP method, which is very robust to noise in the boundary oracle. The robustness to noise may make RCP suitable for use even on noisy quantum devices, which are available within the foreseeable future.

We formalize the problem and discuss the related work in more detail in Sec. II. We present our main result in Sec. III. Finally, we discuss our numerical results in Sec. IV.

II. PRELIMINARIES AND RELATED WORK

A. Convex Optimization

We consider a convex constrained optimization problem [1] of the form

$$\min_{x \in \mathcal{X}} c^T x$$

where $\mathcal{X} \subset \mathbb{R}^n$ is a convex compact set with non-empty interior (“convex body”) and $c \in \mathbb{R}^n$ defines a linear objective. The linear cost function is taken without loss of generality, since any convex constrained optimization problem can be reduced to this form, [25]. Furthermore, we assume that there exist two Euclidean balls $B_r$ and $B_R$ of radii $0 < r < R$, such that $B_r \subseteq \mathcal{X} \subseteq B_R$.

A particularly important class of constrained convex
TABLE I. An overview of recently proposed quantum algorithms for convex optimization on quantum computers, sorted by their appearance in arxiv (listed under Year). In the upper bounds, we drop polylogarithmic terms.

| Reference | Year | Algorithm | Complexity |
|-----------|------|-----------|------------|
| [18]      | 2016 | Multiplicative weights update | \( \sqrt{mns/(Rr/\epsilon)^2} \) |
| [19, 30, 34] | 2017 | Multiplicative weights update | \( \sqrt{mns/(Rr/\epsilon)^8} \) |
| [20]      | 2017 | Multiplicative weights update | \((\sqrt{m}/\epsilon^{10} + \sqrt{m}/\epsilon^{12})s^2\text{poly}(Rr/\epsilon)\) |
| [21, 34]  | 2018 | Multiplicative weights update | \((\sqrt{m} + \sqrt{m}(Rr/\epsilon))s(Rr/\epsilon)^{1/3} \approx Rrs(\sqrt{m}/\epsilon^4 + \sqrt{m}/\epsilon^5)\) |
| [25]      | 2018 | Subgradient | Not given |
| [31]      | 2018 | Subgradient | Not given |
| [36]      | 2021 | Interior-point method | Not given |
| [37]      | 2021 | Interior-point method | Not given |
| [38–40]   | 1979 | Ellipsoid method | \( n^3 \log(nR/\epsilon) \) |
| [45]      | 2004 | Random walk | \( n^3 \log(nR/\epsilon) \) |
| [46]      | 2007 | Random walk probabilistic analyses | \( n^3 \log(nR/\epsilon) \) |
| [47, 48]  | 2010 | Random walk probabilistic analyses | \( n^3 \log(nR/\epsilon) \) |
| [49]      | 2015 | Hybrid center | \( n^3 \log(nR/\epsilon) \) |
| [50]      | 2019 | Volumetric center | \( n^3 \log(nR/\epsilon) \) |

TABLE II. A short history of classical algorithms based on the cutting-plane method (focussing on the feasibility). In the upper bounds (listed under Complexity), we drop polylogarithmic terms and let \( \rho = nR/\epsilon \).

| Reference | Year | Algorithm | Complexity |
|-----------|------|-----------|------------|
| [35, 40]  | 1979 | Ellipsoid method | \( n^3 \log(nR/\epsilon) + n^3 \log(nR/\epsilon) \) |
| [41, 42]  | 1988 | Inscribed ellipsoid | \( n^3 \log(nR/\epsilon) + (n \log(nR/\epsilon))^{2.5} \) |
| [43]      | 1989 | Volumetric center | \( n^3 \log(nR/\epsilon) + n^{4.1} \log(nR/\epsilon) \) |
| [44]      | 1995 | Analytic center | \( n^3 \log^2(nR/\epsilon) + n^{4.1} \log^2(nR/\epsilon) + (n \log(nR/\epsilon))^{3.12}/2 \) |
| [45]      | 2004 | Random walk | \( n^3 \log(nR/\epsilon) + n^7 \log(nR/\epsilon) \) |
| [46]      | 2007 | Random walk probabilistic analyses | \( n^3 \log(nR/\epsilon) + n^7 \log(nR/\epsilon) \) |
| [47, 48]  | 2010 | Random walk probabilistic analyses | \( n^3 \log(nR/\epsilon) + n^7 \log(nR/\epsilon) \) |
| [49]      | 2015 | Hybrid center | \( n^3 \log(nR/\epsilon) + n^7 \log^3(nR/\epsilon) \) |
| [50]      | 2019 | Volumetric center | \( n^3 \log(nR/\epsilon) + n^7 \log^3(nR/\epsilon) \) |

Optimization problems are semidefinite programs [3]:

\[
\inf \langle C, X \rangle \text{ s.t. } AX = b, X \succeq 0 \quad (\text{SDP})
\]

where cone \( \mathcal{K} \) is the cone of positive semidefinite symmetric \( n \times n \) matrices \( \mathcal{S}_+^n \), i.e., \( \{ X = X^T \in \mathbb{R}^{n \times n} : X \text{ is positive semidefinite} \} \), and \( \mathcal{A} : \mathcal{S}^n \rightarrow \mathbb{R}^m \) is a linear operator between \( \mathcal{S}_+^n \) and \( \mathbb{R}^m \):

\[
X \mapsto \left( \langle A_1, X \rangle \cdots \langle A_m, X \rangle \right).
\]

This is a proper generalization of linear programming (LP), second-order cone programming (SOCP), and convex cases of quadratically-constrained quadratic programming (QCQP), which underlie much of operations research. SDPs also have extensive applications in combinatorial optimization, control engineering, (quantum) information theory, machine learning, and statistics. Under the Unique Games Conjecture [38–40], randomized rounding [9] of SDPs obtains the best possible polynomial-time classical algorithms for a variety of problems.

Correspondingly, there have been proposed many classical algorithms for solving constrained convex optimization problems. In summary, SDPs can be classically approximated to any precision in polynomial time. Presently, both the best theoretical bounds [10] and the best practical solvers [11] employ interior-point methods. At least in theoretical models of computation [12] where a real-number arithmetic operation can be performed in unit time, there are classical upper bounds [10] on the run time of the form \( O(\sqrt{m}(mn^2 + n^2) \log(1/\epsilon)) \), where \( O(\cdot) \) indicates the Bachmann–Landau notation, tilde in \( O(\cdot) \) indicates that we drop the polylogarithmic terms, \( n \) is the dimension of the problem, \( \omega \in [2, 2.373] \) is the exponent for matrix multiplication, \( m \) is the number of constraints, and \( n_z \) is the maximal number of non-zero entries per row of the input matrices. For certain smooth instances with sufficient curvature, there are first-order methods [13], which can be faster still. Nevertheless, many of the instances of SDPs encountered in the real-world that are not possible to solve using classical computers in practice.

To introduce the randomized cutting plane method, it is useful to consider perhaps the single most simple optimization algorithm possible: in each iteration, cut a convex body in two parts at its center of gravity, and repeat with the part that yields better objective function. See Algorithm 1 for an outline. This is known as the Deterministic Center-of-Gravity (DCG) algorithm and it has been proposed by Levine [51] and, independently, Newman [52] in 1965.

B. Deterministic Center-of-Gavity (DCG) Algorithm

For a convex body \( \mathcal{X} \subset \mathbb{R}^n \), define its center of gravity as

\[
\text{cg}(\mathcal{X}) = \frac{\int_{\mathcal{X}} x \, dx}{\int_{\mathcal{X}} \, dx}.
\]

Proposition 1 (Grünbaum [53, 54]). Let \( \mathcal{X} \subset \mathbb{R}^n \) be a convex body, and let \( x_G = \text{cg}(\mathcal{X}) \) be its center of gravity. Consider any hyperplane \( \mathcal{H} = \{ x \in \mathbb{R}^n : c^T(x - x_G) = 0 \} \) passing through \( x_G \). This hyperplane divides the set \( \mathcal{X} \)
Algorithm 1: DCG, cf. [51][52]
Input: \( X \)
Output: \( z_k \)
1: \( k = 0, X_k = X \)
2: repeat:
3: \( z_k = \text{cg}(X_k) \)
4: \( X_{k+1} = \{ x \in X_k : c^T(x - z_k) \leq 0 \} \)
5: \( k = k + 1 \)
6: until a stopping criterion is satisfied.

\[
\text{into two subsets}
\]
\[
X_1 = \{ x \in X : c^T x > c^T x_G \},
\]
\[
X_2 = \{ x \in X : c^T x \leq c^T x_G \}.
\]

Then, for \( i = 1, 2 \):
\[
\text{vol}(X_i) \leq \left( 1 - \frac{n}{n+1} \right)^n \text{vol}(X) \leq \left( 1 - \frac{1}{e} \right)^n \text{vol}(X).
\] (2)

Remark 2. Each step of the DCG algorithm guarantees that a given portion of the feasible set is cut out, i.e.,
\[
\text{vol}(X_{k+1}) \leq \left( 1 - \frac{1}{e} \right)^k \text{vol}(X_k).
\]

Applying this inequality recursively, we obtain the volume inequality
\[
\text{vol}(X_k) \leq \left( 1 - \frac{1}{e} \right)^k \text{vol}(X_0) \approx (0.63)^k \text{vol}(X_0),
\] (3)

which proves that DCG has guaranteed geometric convergence in terms of volumes.

By applying Radon’s theorem to the DCG algorithm, we obtain a reduction of cost-function values at each step. Here:

Proposition 3 (Radon, [54]). Let \( X \subset \mathbb{R}^n \) be a convex body and \( x_G = \text{cg}(X) \) be its center of gravity. Denote by \( H \) an arbitrary \((n-1)\)-dimensional hyperplane through \( x_G \), and let \( H_1 \) and \( H_2 \) be the two hyperplane supporting \( X \) and parallel to \( H \). Denote by
\[
r(H) = \frac{\min\{\text{dist}(H, H_1), \text{dist}(H, H_2)\}}{\max\{\text{dist}(H, H_1), \text{dist}(H, H_2)\}}
\]
the ratio of the distances from \( H \) to \( H_1 \) and \( H_2 \), respectively. Then
\[
\min_H r(H) \geq \frac{1}{n}.
\]

Specifically:
\[
f_{k+1} - f^* \leq \frac{n}{n+1} (f_k - f^*).
\] (2.3)

Indeed, let
\[
H_1 = \{ x \in X : c^T x = c^T x_k \}
\]
\[
H = \{ x \in X : c^T x = c^T x_{k+1} \}
\]
\[
H_2 = \{ x \in X : c^T x = c^T x^* \},
\]
then
\[
\text{dist}(H, H_1) = \frac{|c^T x_k - c^T x_{k+1}|}{\|c\|} = \frac{f_k - f_{k+1}}{\|c\|}
\]
and
\[
\text{dist}(H, H_2) = \frac{f_{k+1} - f^*}{\|c\|},
\]

by which proves that \( DCG \) has guaranteed geometric convergence in terms of volumes.

Proposition 4 (Rate of convergence of DCG). Define \( D = f_0 - f^* \). Then the DCG algorithm computes an \( \alpha \)-optimal solution (i.e., such that \( f_k - f^* \leq \alpha \)) in a number of steps bounded as
\[
k = \left\lceil \frac{\ln D}{\ln n + \frac{1}{n}} \right\rceil = O \left( \frac{n \ln D}{\alpha} \right).
\]

This iteration complexity is essentially the same for all cutting-plane methods since 1988, as surveyed in Table 1. We refer to [56] for an in-depth introduction. Notice, however, that computing the deterministic center of gravity (“per-step complexity”) is \#P-hard even for 0-1 polytopes [57].

C. An RCP algorithm

Over the past two decades, there have been developed cutting-plane algorithms [45][48][e.g.] that replace the computing of the center of gravity of a convex body with sampling points uniformly at random from the convex
An overview of geometric random walks: Their mixing times and upper bounds on the per-step complexity of sampling uniform distribution over the spectrahedron in the classical implementations by Chalkis et al. [58].

### Table III

| Reference | Year | Random walk | Mixing time | Per-step complexity for SDP |
|-----------|------|-------------|-------------|-----------------------------|
| 59 [60]   | 1984 | Hit and Run | Fast [60]   | $O(m^{2.697} + 1.3\log L + nm^2)$ |
| 59        | 1984 | Coordinate-directions hit and run | Unknown     | $O(m^{2.697} + 1.3\log L + m^2)$ |
| 61        | 2014 | Billiard walk | Unknown     | $O(m^{2.697} + 1.3\log L + nm^2)$ |
| 62 [64]   | 2015 | Hamiltonian Monte Carlo with reflections | Unknown     | $O(m^{2.697} + 1.3\log L + nm^2)$ |
| 65        | 2019 | Wang-Landau | Fast [65]   | Unknown                     |

Algorithm 2: Randomized cutting plane [45] [47] [60] [60]

**Input:** $X$

**Output:** $z_k$

1: $k = 0, \mathcal{X}_k = X$

2: **repeat**

3: generate $N_k$ uniformly distributed random samples

in $\mathcal{X}_k$, $\{x^{(1)}, \ldots, x^{(N_k)}\}$, e.g., using Algorithm 3

4: $z_k = \arg\min_{x \in \{x^{(1)}, \ldots, x^{(N_k)}\}} c^T x$

5: $\mathcal{X}_{k+1} = \{x \in \mathcal{X}_k : c^T (x - z_k) \leq 0\}$

6: $k = k + 1$

7: **until** a stopping rule is satisfied.

Algorithm 3: Hit-and-run (H&R), cf. [59] [60]

**Input:** $\mathcal{X}, x_0 \in \mathcal{X}, M$ (mixing time)

**Output:** random point $x \in \mathcal{X}$

1: $y(0) = x_0$

2: **for** $i = 0$ to $M - 1$ **do**

3: generate a uniformly distributed random direction $v \in \mathbb{R}^n$

4: $(x, \bar{x}) = BO(\mathcal{X}, y(i), v)$

5: generate a uniform point $y(i+1)$ in the segment $[x, \bar{x}]$

6: **end for**

7: $x = y(M)$

Algorithm 4: RCP with H&R, cf. [44] [60] [67]

**Input:** $\mathcal{X}, x_0 \in \mathcal{X}, M$

**Output:** $z_k$

1: $k = 0, \mathcal{X}_k = X$

2: **repeat**

3: **for** $j = 1$ to $N$ **do**; $x^{(j)} = \text{H&R}(x^{(j-1)}, M)$; **end for**

4: $z_k = \min_{x \in \{x^{(1)}, \ldots, x^{(N)}\}} c^T x$

5: $\mathcal{X}_{k+1} = \{x \in \mathcal{X}_k : c^T (x - z_k) \leq 0\}$

6: $k = k + 1$

7: **until** a stopping rule is satisfied.

D. Boundary Oracle for Hit-and-Run Walks on the Feasible Set of an SDP

Let us now consider the complexity of implementing a boundary oracle for the Random Directions Hit and Run random walk [59] for sampling uniformly at random from the spectrahedron (SDP). For convenience, let us consider the dual of the semidefinite program (SDP), also known as the linear matrix inequality (LMI):

$$
\min c^T x \text{ s.t. } F(x) = F_0 + \sum_{i=1}^n x_i F_i \leq 0, \quad \text{(LMI)}
$$

where $c \in \mathbb{R}^n$ and $F_i = F_i^T \in \mathbb{R}^{m \times m}, i = 0, \ldots, n$, are known symmetric matrices. We then have the convex set

$$
\mathcal{X} = \mathcal{X}_{\text{LMI}} = \{x \in \mathbb{R}^n : F(x) \preceq 0\}.
$$

We assume $\mathcal{X}_{\text{LMI}}$ is nonempty and bounded.

Given $y \in \mathcal{X}_{\text{LMI}}$ such that $F(y) < 0$ and a random direction $v \in \mathbb{R}^n$, how do we find the intersection points of the line $z = y + \lambda v$ and the boundary of $\mathcal{X}_k$ at the $k$-th iteration? First, we have

$$
F(y + \lambda v) = F(y) + \lambda (F(v) - F_0) \equiv A + \lambda B, \quad \lambda \in \mathbb{R}.
$$

Next, we obtain the intersection points with the boundary of $\mathcal{X}_{\text{LMI}}$: $\bar{z} = y + \lambda v$ and $\bar{z} = y + \lambda v$ and test if

- body. An outline of such a randomized cutting-plane method is presented in Algorithm 2.

The uniform sampling is non-trivial, but a breakthrough result of [68] showed that it is possible using certain rapidly-mixing geometric random walks [60]. An overview of the geometric random walks is presented in Table III. For any such random walk, one needs to provide one or more geometric subroutines, such as the test of membership of a point inside the set, a surface separating a point from the set, etc. Several standard subroutines are beautifully surveyed in Chapter 3 of [54]. Our focus in this work will be on the so-called Random Directions Hit and Run random walk [59], wherein the key subroutine is the intersection of a line (or curve, more generally) with the boundary of the feasible set. This subroutine is commonly known as the boundary oracle (BO). See Algorithm 3 for an overview.

In the Supplementary Material, we present some background material concerning the statistical properties of the empirical minimum over a convex body in Appendix A. In Appendix B, we present an iteration complexity of the overall procedure, as captured in Algorithm 4. In particular, we fix minor issues of previous analyses, especially those of Dabbene et al. [47] [48]. In Appendix C, we provide the full pseudo code of the algorithms, specialized to SDPs. We note that the pseudocode and the bounds on the iteration complexity remain the same, independent of whether the boundary oracle is run classically or quantumly.
\( \tilde{z}, \tilde{z} \in X_k = \{ x \in X_{k-1} : c^T(x - z_{k-1}) \leq 0 \} \). If both points are in \( X_k \), then \( \{ \tilde{z}, \tilde{z} \} \) are the intersection points we need. Otherwise, only one of them \( \notin X_k \), so w.l.o.g. assume \( \tilde{z} \notin X_k \), and we need to find the intersection point between the line \( z = y + \lambda v \) and the hyperplane \( \{ x \in X : c^T(x - z_{k-1}) = 0 \} \), which can be easily obtained by solving for \( \lambda \) in \( c^T(y + \lambda v - z_{k-1}) = 0 \). Let \( \lambda^* \) denote the solution and let \( \tilde{z}' = y + \lambda^* v \). Then, \( \{ \tilde{z}, \tilde{z}' \} \) are the desired intersection points.

The work of [14, 60, 67] can be summarized as follows:

**Lemma 5** (Boundary oracle for LMIs, Lemma 6 in [47]). Let \( A < 0 \) and \( B = B^T \). Then, the minimal and the maximal values of the parameters \( \lambda \in \mathbb{R} \) retaining the negative definiteness of the matrix \( A + \lambda B \) are given by

\[
\lambda = \begin{cases} 
\max_{\lambda_i < 0} \lambda_i \\
-\infty & \text{if all } \lambda_i > 0
\end{cases}
\]

and

\[
\bar{\lambda} = \begin{cases} 
\min_{\lambda_i > 0} \lambda_i \\
+\infty & \text{if all } \lambda_i < 0
\end{cases}
\]

where \( \lambda_i \) are the generalized eigenvalues of the pair of matrices \((A, -B)\), i.e., \( Av_i = -\lambda_i Bv_i \).

The semidefinite generalized eigenvalue problem [70, Chapter 3] could be seen as a special cases of the polynomial eigenvalue problem [71, 72]. There, we wish to compute \( \lambda \in \mathbb{R} \) and \( x \in \mathbb{R}^m \) satisfying

\[
(B_d \lambda^d + \cdots + B_1 \lambda + B_0)x = 0 \tag{PEP}
\]

where \( B_i \in \mathbb{R}^{m \times m} \) are matrices, out of which \( B_d \) and \( B_0 \) are invertible, and all could be seen as coefficients of a univariate matrix polynomial.

Despite much recent progress in computational approach to the polynomial eigenvalue problem [71, 73, 75], and effective computational geometry for surfaces [76], more broadly, a classical implementation of the boundary oracle that would make the hit-and-run walk on the feasible set of SDP (or LMI) is still lacking. In particular, the present best classical run-time bound is:

**Lemma 6** (Chalkis et al., [58]). Consider a PEP of degree \( d \), involving matrices of dimension \( m \times m \), with integer elements of bitsize at most \( \tau \). There is a randomized algorithm for computing the eigenvalues and the eigenvectors of PEP up to precision \( \epsilon = 2^{-L} \), in time \( \tilde{O}((md)^{\omega+3} (md)^3 \tau) \). The arithmetic complexity is \( \tilde{O}(d^2 \cdot 697 + md\log(1/\epsilon)) \). The eigenvectors are roots of the characteristic polynomial of \( C_2 \).

Unfortunately, this is not much easier than solving the original convex constrained optimization problem, which has [10] the arithmetic complexity \( \tilde{O}(\sqrt{n}(mn^2 + m^\omega + \omega^\omega) \log(1/\epsilon)) \), and wherein important special cases [77] e.g., can be solved in matrix-multiplication time.

### III. A Boundary Oracle via Quantum Eigensolvers

Our main result is a family of quantum algorithms for the boundary oracle for hit-and-run walks on the feasible set of an SDP, or rather its dual (LMI). Therein, we transform the generalized eigenvalue problem to an eigenvalue problem on a larger matrix, which makes it possible to use any quantum algorithm for computing the eigenvalues of the larger matrix. Quantum eigensolvers are, in turn, some of the best understood quantum algorithms [78], with practical algorithms [79, 80, e.g.] even for noisy quantum devices. Indeed, one can show [81] that any algorithm for a quantum computer with an exponential speed-up is reducible to an eigensolver.

There are two options for linearising the generalized eigenvalue problem, broadly speaking. Either we utilize the companion linearization [82, 84] to transform the polynomial eigenvalue problem (PEP) into a linear pencil in a higher dimension, or we utilize the congruence transformations [70]. Either way, we express the generalized eigenvalues in the generalized problem as the standard eigenvalues of a larger matrix.

#### A. Companion Linearization

Let us consider the polynomial eigenvalue problem (PEP). Starting from the generalized eigenvalue problem \( C_0 - \lambda C_1 \), where the companion matrices [82, Chapter 4] are:

\[ C_0 = \begin{bmatrix} B_d & 0 & \cdots & 0 \\ 0 & I_m & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & I_m \end{bmatrix}, \quad C_1 = \begin{bmatrix} B_{d-1} & B_{d-2} & \cdots & B_0 \\ -I_m & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & -I_m & 0 \end{bmatrix}, \]

where \( I_m \) denotes the \( m \times m \) identity matrix. We obtain the usual linear eigenvalue problem \((\lambda I_d - C_2)z = 0\), where

\[ C_2 = \begin{bmatrix} B_{d-1}B_d^{-1} & B_{d-2}B_d^{-1} & \cdots & B_0B_d^{-1} \\ -I_m & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & -I_m & 0 \end{bmatrix}. \]

The eigenvectors are roots of the characteristic polynomial of \( C_2 \).

This approach is ready to be used on noisy quantum devices, in the sense that it does not require the implementation of any numerical linear algebra on the quantum device, other than the eigensolver, and moreover, in that it is very robust to errors in the quantum eigensolver.

#### B. Congruence Transformations

An alternative approach is known as the congruence transformations. This stems from the work of Lucas [70].
TABLE IV. A short history of options for translating generalized eigenvalue problems to eigenvalue problems.

| Ref. | Year | Approach / Algorithm |
|------|------|----------------------|
| 1976 | folklore | Companion linearization |
| 1971 | Three RRD (SPEC / SPEC / SVD) |
| 1972 | Three RRD (SPEC / SPEC / QR) |
| 1984 | MDR |
| 1987 | Three RRD (all orthogonal) |
| 1993 | Generalized Upper Triangular (GUPTRI) |
| 2004 | Orthogonal RRD (SPEC / SPEC / SVD) |
| 2004 | Non-orthogonal (Cholesky / LDL^T / QRP) |

Chapter 3], which generalizes earlier work of Fix and Heiberger [86], Parlett [85], and Cao [88]. We refer to Section 3.4.5 of [70] for the discussion of the computational complexity. While this work, summarized in Table IV, is fundamental in (multi)linear algebra, it is surprisingly little known. Having said that, it may be less suited to noisy quantum devices, in the sense that the quantum eigensolver gets compounded up to three times within the quantum boundary oracle.

IV. EXPERIMENTAL RESULTS

We have implemented the random-walk variant of the cutting-plane method specialized to SDPs in Python, with a view of inclusion of the code in Qiskit [91]. The pseudo code of the algorithms is presented in Appendix C, while numerical constants and other details of the implementation are discussed in Appendix D.

We have tested our implementation on SDPLIB [92], a well-known benchmark. Table V presents an overview of the solution quality obtained on a subset of the instances. We should like to stress that the SDPLIB has been tested on any instances from SDPLIB, yet. Likewise, while there has been much effort focussed on implementations [55] [59] of SDP solvers, we believe these to be the first reported results of a cutting plane method on the SDPLIB.

As can be seen in Table V, our method is much slower than classical interior-point methods. For example, on the instance hinf1, which has been originally developed by P. Gahinet within control-theoretic applications, we have not been able to solve the instance hinf1, which has been originally developed by P. Gahinet within control-theoretic applications, using a 14×14 PSD matrix and 13 inequalities, our method converges to 2 significant digits in the objective function within 127 seconds. In contrast, a commonly-used classical solver SCS 2.1.4 [91] [95] solves the hinf1 to 3 significant digits in the objective function within 5.68 seconds on the same hardware; many interior-point methods [83] are much faster still. As we detail in Table V in Appendix E on many other instances, our method terminates after 24 hours without obtaining a solution matching 1 significant digit in the value of the objective function. Despite the appealing iteration-complexity results for cutting-plane methods, cf. Table II, their practical utility remains limited, when executed classically.

In terms of a potential quantum speed-up, much depends on the speed-up of the eigensolver, as discussed in Section III. For instance, on qap6, which features a 37×37 PSD matrix variable, approximately 8 hours and 24 minutes are spent in the eigensolver, classically. A square root of the run-time, which could illustrate a quadratic speed-up in a realistically implementable quantum eigensolver [79] [96], would reduce this to less than 174 seconds. A logarithmic reduction of the run-time, which could illustrate the impact of an exponential quantum speed-up [97], would reduce this to less than 5 seconds. A commonly used classical solver SCS 2.1.4 [91] [95] solves the qap6 within 1.49 seconds on the same classical hardware.

Even the exponential speed-up in the eigensolver would hence yield a speed-up of the overall cutting-plane method only for instances (much) larger than a 37×37 PSD matrix, whilst our current ability to realize any quantum speed-up [79] [98] whatsoever in an eigensolver for an 37×37 matrix is lacking. Indeed, even the state preparation for a 37×37 matrix is presently out of reach. Still, should the state preparation for larger matrices prove feasible, the overall speed-up may be of interest.

On a more positive note, in terms of the robustness to errors in the boundary oracle, which could be implemented with a quantum eigensolver as discussed in Section III, the random-walk variant of the cutting-plane method may be hard to improve upon. As we illustrate in Figure 1, even multiplicative noise in the eigenvalue computation corresponding to the signal-to-noise ratio of approximately 2 dB does not change the performance of the algorithm on qap6, substantially. (See Appendix D for the details of the noise model.) This also has an intuitive interpretation, when one recalls that we use the boundary oracle to estimate the line segment along a sampled random direction that lies within the feasible set. We do not, however, use the estimated end points of
the line segment *per se*: we only sample from the line segment. Unless the error in the eigensolve leads to sampling from beyond the line segment, outside of the feasible set, the error has no discernible impact on the performance and does not propagate further. The fact that we can accommodate a substantial amount of noise in the quantum eigensolver could be seen as a basis of an approach suitable for noisy quantum devices.

V. CONCLUSIONS

We have demonstrated how to utilize eigensolvers in solving semidefinite programs, which are perhaps the broadest widely used class of convex optimization problems. The resulting randomized cutting plane method has several non-trivial steps, with several design choices for each step, as documented in Tables III–IV. This may hence suggest something of a framework for the development of further related algorithms, by varying the design choices we made.

ACKNOWLEDGEMENTS

The authors acknowledge a substantial contribution by Cunlu Zhou during his internship at IBM Research in the summer of 2018, including the first implementation. Cunlu chose not to be listed as a co-author. Jakub acknowledges useful discussions with Joran Van Apeldoorn, András Gilyén, Sander Gribling, Martin Mevissen, and Jiri Vala. We also acknowledge that [24] has appeared at a similar time, with a different algorithm, also applicable to noisy quantum devices, but without any run-time bounds.

IBM, the IBM logo, and ibm.com are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. The current list of IBM trademarks is available at https://www.ibm.com/legal/copytrade.

Jakub Mareček’s research has been supported by the OP VVV project CZ.02.1.01/0.0/0.0/16 019/0000765 Research Center for Informatics.

This work has also received funding from the Disruptive Technologies Innovation Fund (DTIF), by Enterprise Ireland, under project number DTIF2019-090.

[1] Stephen P Boyd and Lieven Vandenberghe, *Convex optimization* (Cambridge university press, 2004).
[2] Indeed, if $f(x)$ is convex,

$$
    f(x) \rightarrow \min_{x \in \mathcal{X}}
$$

by introducing a slack variable $t$, we obtain an equivalent optimization problem of the form [1]

$$
    \min_{x \in \mathcal{X}} \quad f(x) - t \leq 0
$$

[3] Miguel F Anjos and Jean B. Lasserre, *Handbook on semidefinite, conic and polynomial optimization* Vol. 166 (Springer Science & Business Media, 2011).
[4] We still assume there are two Euclidean balls $B_r$ and $B_R$ inscribed and outscribed to the feasible set of SDP, which is known as the spectrahedron. Radius $R$ of the ball $B_R$ above can also be seen as an upper bound on the trace of an optimal primal solution of an SDP. Notice that in the case of a general SDP, parameters $r$ and $R$ are not constants independent of dimension, but do grow with the dimension.
[5] Anirudha Majumdar, Georgina Hall, and Amir Ali Ahmadi, “Recent scalability improvements for semidefinite programming with applications in machine learning, control, and robotics,” *Annual Review of Control Robotics, and Autonomous Systems* 3, 331–360 (2020).
[6] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O’Donnell, “Optimal inapproximability results for MAX-CUT and other 2-variable CSPs?” *SIAM Journal on Computing* 37, 319–357 (2007).
[7] S. Khot, “On the unique games conjecture (invited survey),” in *2012 IEEE 27th Conference on Computational Complexity* (IEEE Computer Society, Los Alamitos, CA, USA, 2010) pp. 99–121.
[8] Subhash A Khot and Nisheeth K Vishnoi, “The unique games conjecture, integrality gap for cut problems and embeddability of negative-type metrics into $l_1$,” *Journal of the ACM (JACM)* 62, 1–39 (2015).
[9] Prabhakar Raghavan and Clark D. Tompson, “Randomized rounding: A technique for provably good algorithms and algorithmic proofs,” *Combinatorica* 7, 365–374 (1987).
[10] Haotian Jiang, Tarun Kathuria, Yin Tat Lee, Swati Padmanabhan, and Zhao Song, “A faster interior point method for semidefinite programming,” in *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)* (IEEE, 2020) pp. 910–918.
[11] APS Mosek, “The mosek optimization software,” (2020), online at http://www.mosek.com.
[12] Lenore Blum, Felipe Cucker, Michael Shub, and Steve Smale, *Complexity and real computation* (Springer Science & Business Media, 2012).
[13] Alp Yurtsever, Joel A Tropp, Olivier Ferozq, Madeleine Udell, and Volkan Cevher, “Scalable semidefinite programming,” *arXiv:1912.02949* (2019).
[14] Marc Ganzhorn, Daniel J. Egger, Panagiotis Kl. Bark-
TABLE V. An overview of the solution quality obtained within a 24-hour time limit on a subset of smaller problems from SDPLIB, a well-known benchmark: Instance name, its variant, number of constraints and size of the matrix variable, reference objective function value in SDPLIB, objective function value obtained by a classical solver (SCS, [34 [35]), and the objective function value at the termination of RCP (in parentheses, if the run terminated with a time out). Reasons for termination are detailed in Appendix [VI].

| Instance | Type | m | n x n | Ref. Obj. | SCS Obj. | RCP Obj. |
|----------|------|---|-------|-----------|----------|----------|
| hin1     | primal | 13 | 14 x 14 | 2.032600  | 2.036422 | 2.253482 |
| hin10    | primal | 21 | 18 x 18 | 108.711800 | 107.808783 | 151.069581 |
| mcp100   | primal | 100 | 100 x 100 | 226.157350 | 226.151147 | 318.123641 |
| mcp124-1 | primal | 124 | 124 x 124 | 141.990480 | 141.988895 | (250.149694) |
| mcp124-2 | primal | 124 | 124 x 124 | 269.880170 | 269.885616 | (383.003507) |
| mcp124-3 | primal | 124 | 124 x 124 | 467.750110 | 467.755324 | (614.279456) |
| mcp124-4 | primal | 124 | 124 x 124 | 864.418600 | 864.413665 | (1120.640088) |
| mcp250-2 | primal | 250 | 250 x 250 | 531.930080 | 531.928325 | (758.00964) |
| mcp250-3 | primal | 250 | 250 x 250 | 981.172570 | 981.185856 | (1272.445273) |
| mcp250-4 | primal | 250 | 250 x 250 | 1681.960100 | 1681.959153 | (2155.741152) |
| truss1   | primal | 6 | 13 x 13 | -8.999996  | -8.999996 | -7.101001 |
| truss2   | primal | 58 | 133 x 133 | -123.380360 | -123.376950 | (21.458773) |
| truss3   | primal | 27 | 31 x 31 | -9.109996  | -9.110169  | 5.232127  |
| truss4   | primal | 12 | 19 x 19 | -9.009996  | -9.010000 | -5.826993 |

-outsos, Pauline Ollitrault, Gian Salis, Nikolaj Moll, Andreas Fuhrer, Peter Mueller, Stefan Woerner, Ivan Tav-ernelli, and Stefan Filipp, “Gate-efficient simulation of molecular eigenstates on a quantum computer,” Phys.

[15] Vojtech Havlicek, Antonio D. Corcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta, “Supervised learning with quantum-enhanced feature spaces,” Nature 567, 209–212 (2019).

[16] Daniel J Egger, Claudio Gambetta, Jakub Marecek, Scott McFaddin, Martin Mevissen, Rudy Raymond, Andrea Simonetto, Stefan Woerner, and Elena Yudurain, “Quantum computing for finance: state of the art and future prospects,” IEEE Transactions on Quantum Engineering (2020).

[17] Ankit Garg, Robin Kothari, Praneeth Netrapalli, and Suhail Sheriff, “No Quantum Speedup over Gradient Descent for Non-Smooth Convex Optimization,” in 12th Innovations in Theoretical Computer Science Conference (ITCS 2021), Leibniz International Proceedings in Informatics (LIPIcs), Vol. 185, edited by James R. Lee (Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2021) pp. 53:1–53:20.

[18] F. G. S. L. Brandao and K. M. Svore, “Quantum speedups for solving semidefinite programs,” in 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS) (2017) pp. 415–426.

[19] J. Van Apelbroeck, A. Gilyén, S. Gribling, and R. de Wolf, “Quantum sdvp solvers: Better upper and lower bounds,” in 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS) (2017) pp. 403–414.

[20] Fernando GSL Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M Svore, and Xiaodi Wu, “Quantum sdvp solvers: Large speed-ups, optimality, and applications to quantum learning,” in 40th International Colloquium on Automata, Languages, and Programming (ICALP 2019) (Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019).

[21] Joran van Apelbroeck and András Gilyén, “Improvements in quantum sdvp solving with applications,” in Proceedings of 46th International Colloquio on Automata, Languages, and Programming (ICALP 2019) (2019).

[22] Sanjeev Arora and Satyen Kale, “A combinatorial, primal-dual approach to semidefinite programs,” in Proceedings of the thirty-ninth annual ACM symposium on Theory of computing (2007) pp. 227–236.

[23] Elad Hazan, “Sparse approximate solutions to semidefinite programs,” in Latin American symposium on theoretical informatics (Springer, 2008) pp. 306–316.

[24] We refer to Algorithm 6 in [18] for a nice overview of the algorithm.

[25] Iordanis Kerenidis and Anupam Prakash, “A quantum interior point method for lps and sdps,” ACM Transactions on Quantum Computing 1 (2020), 10.1145/3406306.

[26] Brandon Augustino, Giacomo Nannicini, Tamás Terlaky, and Luis F. Zuluaga, “An inexact-feasible quantum interior point method for semidefinite optimization,” (2021).

[27] S.J. Wright, Primal-Dual Interior-Point Methods, Other Titles in Applied Mathematics (Society for Industrial and Applied Mathematics, 1997).

[28] We refer to Chapter 1 of [27] for an excellent introduction to primal-dual interior-point methods. Due to the reliance of reliance of interior-point methods on solving linear systems, [7, 9, 15] ended up with a bound dependent on the condition number of a linear system based on the Karush-Kuhn-Tucker (KKT) conditions. As is well known [27, p. 215], this goes to infinity for all instances, by the design of the method, which may be not ideal in practice. Furthermore, there is the issue of the HHL algorithm [99] providing the solution of the linear system only as a quantum state, whereas the interior-point method needs a classical update. The HHL hence needs to be run many times and the quantum state measured many times, to estimate the classical update.

[29] András Gilyén, Srinivasan Arunachalam, and Nathan Wiebe, “Optimizing quantum optimization algorithms
via faster quantum gradient computation," in *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms* (SIAM, 2019) pp. 1425–1444.

[30] Joran Van Apeldoorn, András Gilyén, Sander Gríbling, and Ronald de Wolf, “Quantum SDP-solvers: Better upper and lower bounds.” *Quantum* 4, 230 (2020).

[31] Shouvanik Chakrabarti, Andrew M Childs, Tongyang Li, and Xiaodi Wu, “Quantum algorithms and lower bounds for convex optimization,” *Quantum* 4, 221 (2020).

[32] As it has been shown in [30, Appendix E], in the MWU algorithm, $\frac{\epsilon}{2}$ should be seen as an important parameter, as one can trade-off dependence on one of the three individual parameters for the dependence on the others.

[33] Martin Grötschel, László Lovász, and Alexander Schrijver, *Geometric algorithms and combinatorial optimization*, Vol. 2 (Springer Science & Business Media, 2012).

[34] Joran van Apeldoorn, *A quantum view on convex optimization*, Ph.D. thesis, University of Amsterdam Institute for Language, Logic and Computation (ILLC) (2020).

[35] Joran van Apeldoorn, András Gilyén, Sander Gríbling, and Ronald de Wolf, “Convex optimization using quantum oracles,” *Quantum* 4 (2020), 10.22331/q-2020-07-13-220.

[36] Mohammad Hossein Mohammadi Siahroudi, Ramin Fakhami, and Tamás Terály, “Efficient use of quantum linear system algorithms in interior point methods for linear optimization,” (2021) arXiv:2106.03891 [quant-ph].

[37] Kishor Bhati, Tobias Haug, Vladko Vedral, and Leong-Chuan Kwek, “Nisq algorithm for semidefinite programming,” (2021), arXiv:2106.03891 [quant-ph].

[38] Naum Z Shor, “Cut-off method with space extension in convex programming problems,” *Cybernetics* 13, 94–96 (1977).

[39] David B Yudin and Arkadii S Nemirovski, “Evaluation of the information complexity of mathematical programming problems.” *Ekononika i Matematicheskie Metody* 13, 128–142 (1976).

[40] Leonid G Khachiyan, *Polynomial algorithms in linear programming*, USSR Computational Mathematics and Mathematical Physics 20, 53–72 (1980).

[41] Leonid G Khachiyan, Sergei Pavlovich Tarasov, and I. I. Erlikh, “The method of inscribed ellipsoids,” in *Soviet Math. Dokl*, Vol. 37 (1988) pp. 226–230.

[42] Y.E. Nesterov and A.S. Nemirovskii, “Self-concordant functions and polynomial-time methods in convex programming,” (1989), report, Central Economic and Mathematics Institute, USSR Acad. Sci.

[43] Pravin M Vaidya, “A new algorithm for minimizing convex functions over convex sets,” in *30th Annual Symposium on Foundations of Computer Science* (IEEE Computer Society, 1989) pp. 338–343.

[44] David S Atkinson and Pravin M Vaidya, “A cutting plane algorithm for convex programming that uses analytic centers,” *Mathematical Programming* 69, 1–43 (1995).

[45] Dimitris Bertsimas and Santosh Vempala, “Solving convex programs by random walks,” *Journal of the ACM (JACM)* 51, 540–556 (2004).

[46] Giuseppe C. Calafiore and Fabrizio Dabbene, “A probabilistic analytic center cutting plane method for feasibility of uncertain linear systems,” *Automatica* 43, 2022–2033 (2007).

[47] F. Dabbene, P. Shcherbakov, and B. T. Polyak, “A randomized cutting plane scheme with geometric convergence: Probabilistic analysis and sdp applications,” in *2008 47th IEEE Conference on Decision and Control* (2008) pp. 3044–3049.

[48] Fabrizio Dabbene, Pavel S Shcherbakov, and Boris T Polyak, “A randomized cutting plane method with probabilistic geometric convergence,” *SIAM Journal on Optimization* 20, 3185–3207 (2010).

[49] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong, “A faster cutting plane method and its implications for combinatorial and convex optimization,” in *2015 IEEE 56th Annual Symposium on Foundations of Computer Science* (IEEE, 2015) pp. 1049–1056.

[50] Haotian Jiang, Yin Tat Lee, Zhao Song, and Sam Chiu-wai Wong, “An improved cutting plane method for convex optimization, convex-cone games, and its applications,” in *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing* (2020) pp. 944–953.

[51] Anatoly Yu’evich Levin, “An algorithm for minimizing convex functions,” *Doklady Akademii Nauk* 160, 1244–1247 (1965).

[52] Donald J Newman, “Location of the maximum on unimodal surfaces,” *Journal of the ACM (JACM)* 12, 395–398 (1965).

[53] Branko Grünbaum, “Partitions of mass-distributions and of convex bodies by hyperplanes.” *Pacific Journal of Mathematics* 10, 1257–1261 (1960).

[54] Note that some literature [47, 48] restates the lemma incorrectly, with $\frac{1}{n}$ instead of $\frac{1}{n}$.

[55] Johann Radon, “Mengen konvexer körper, die einen gemeinsamen punkt enthalten,” *Mathematische Annalen* 83, 113–115 (1921).

[56] Sébastien Bubeck, “Convex optimization: Algorithms and complexity,” *Found. Trends Mach. Learn.* 8, 231–357 (2015).

[57] Luis A Rademacher, “Approximating the centroid is hard,” in *Proceedings of the twenty-third annual symposium on Computational geometry* (2007) pp. 302–305.

[58] Apostolos Chalkis, Ioannis Emiris, Vissarion Fisikopoulos, Panagiotis Repouskos, and Elias Tsigaridas, “Efficient sampling from feasible sets of sdps and volume approximation,” arXiv preprint arXiv:2010.03817 (2020).

[59] Robert L Smith, “Efficient monte carlo procedures for generating points uniformly distributed over bounded regions,” *Operations Research* 32, 1296–1308 (1984).

[60] László Lovász, “Hit-and-run mixes fast,” *Mathematical Programming* 86, 443–461 (1999).

[61] Boris T Polyak and E. N. Gryazina, “Billiard walk—a new sampling algorithm for control and optimization,” *IFAC Proceedings Volumes* 47, 6123–6128 (2014).

[62] Simon Duane, A.D. Kennedy, Brian J. Pendleton, and Duncan Roweth, “Hybrid monte carlo,” *Physics Letters B* 195, 216 – 222 (1987).

[63] Hadi Mohassel Afshar and Justin Donkey, “Reflection, refraction, and hamiltonian monte carlo,” in *Advances in Neural Information Processing Systems* Vol. 28, edited by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Curran Associates, Inc., 2015) pp. 3007–3015.

[64] Augustin Chevalier, Sylvain Pion, and Frédéric Cazals, *Hamiltonian Monte Carlo with boundary reflections, and application to polytope volume calculations*. 
based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning," in Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020 (Association for Computing Machinery, New York, NY, USA, 2020) p. 387–400.

[102] Ewin Tang, “Quantum principal component analysis only achieves an exponential speedup because of its state preparation assumptions,” Phys. Rev. Lett. 127, 060503 (2021).

[103] Nadiia Chepurko, Kenneth L Clarkson, Lior Horesh, and David P Woodruff, “Quantum-inspired algorithms from randomized numerical linear algebra,” arXiv preprint arXiv:2011.04125 (2020).

[104] Horst Alzer, “Some beta-function inequalities,” Proceedings. Section A, Mathematics-The Royal Society of Edinburgh 133, 731 (2003).

[105] Note that unlike some analyses in the literature [47, 48], we use the correct $h^2$ instead of $h$.

[106] Depending on what data are used for the numerical experiments, we may not need to solve the initial optimization problem. For example, if we choose $F_0$ such that $F_0 \prec 0$, then we can use $x_0 = 0$ as the initial point.
Appendix A: Statistical properties of the empirical minimum over a convex body

Given $c \in \mathbb{R}^n$, we define the following random variables that represent the value of a linear objective evaluated at the random points $x^{(i)}$: $f^{(i)} = c^T x^{(i)}$, $i = 1, \ldots, N$. Then we can define the so-called empirical minimum over these random points as

$$f_{[1]} = \min_{i=1, \ldots, N} f^{(i)}. \quad (A1)$$

Notice that $f_{[1]}$ is also a random variable; it represents the so-called first order statistics of $f^{(i)}$. The key theorem below proves that, for every convex body $\mathcal{X}$, the expected value of the relative distance between the empirical minimum $f_{[1]}$ and the true one $f^* = \min_{x \in \mathcal{X}} c^T x$ is bounded from below and from above by constants that depend only on $n$ and $N$.

**Lemma 7** (Brunn). Let $\mathcal{X} \subset \mathbb{R}^n$ be a convex body. Define a parallel slice $\mathcal{X}_s = \mathcal{X} \cap \{x : x_1 = s\}$ and its $(n-1)$-dimensional volume $v_{\mathcal{X}}(s) = vol(\mathcal{X}_s)$. Then the function $v_{\mathcal{X}}(s)\frac{\lambda}{\pi^{n/2}}$ is concave, and

$$vol(\mathcal{X}_s)\frac{\lambda}{\pi^{n/2}} \geq \lambda vol(\mathcal{X}_s)\frac{\lambda}{\pi^{n/2}} + (1 - \lambda)vol(\mathcal{X}_s)\frac{\lambda}{\pi^{n/2}},$$

where $s = \lambda s_1 + (1 - \lambda)s_2, \lambda \in [0, 1]$.

**Proof.** First note that $\lambda \mathcal{X}_s + (1 - \lambda)\mathcal{X}_s \subseteq \mathcal{X}_s$. Indeed, using convexity of $\mathcal{X}$, for all $(s_1, y_1) \in \mathcal{X}_s$, and $(s_2, y_2) \in \mathcal{X}_s$, we have $(s, \bar{y}) = (\lambda s_1 + (1 - \lambda)s_2, \lambda y_1 + (1 - \lambda)y_2) = \lambda(s_1, y_1) + (1 - \lambda)(s_2, y_2) \in \mathcal{X} \cap \{x : x_1 = s\} = \mathcal{X}_s$. Then, by the Brunn-Minkowski inequality, we have

$$vol(\mathcal{X}_s)\frac{\lambda}{\pi^{n/2}} \geq \lambda vol(\mathcal{X}_s)\frac{\lambda}{\pi^{n/2}} \geq \lambda vol(\mathcal{X}_s)\frac{\lambda}{\pi^{n/2}} + (1 - \lambda)vol(\mathcal{X}_s)\frac{\lambda}{\pi^{n/2}},$$

i.e., $v_{\mathcal{X}}(s)\frac{\lambda}{\pi^{n/2}}$ is concave. \hfill \Box

**Theorem 8.** Let $\mathcal{X} \subset \mathbb{R}^n$ be a convex body. Given $c \in \mathbb{R}^n$, define $h = \max_{\mathcal{X}} c^T x - \min_{\mathcal{X}} c^T x$ and $f^* = \min_{x \in \mathcal{X}} c^T x$. Then it holds that

$$\frac{h}{nN + 1} \leq \mathbb{E}[f_{[1]} - f^*] \leq \frac{h}{n} B \left( N + 1, \frac{1}{n} \right) \leq h \left( \frac{1}{N + 1} \right) \frac{\lambda}{\pi^{n/2}}, \quad (A2)$$

where the expectation is taken with respect to samples $x^{(1: \infty)}$ and $B(\cdot, \cdot)$ is the Euler Beta function.

**Proof.** Assume, without loss of generality, that $c = [1 \ldots 0]^T$ (that is, $c^T x = x_1$) and that $x^* = \arg \min_{\mathcal{X}} c^T x = 0$. (Indeed, if in general $c^T x = \sum_{i=1}^n c_i x_i$, then let $y_1 = \sum_{i=1}^n c_i x_i$, so equivalently, we have $\min c^T y = y_1$, $y \in \mathcal{X}$, where $\hat{c} = [1 \ldots 0]$ and $\hat{\mathcal{X}} = \{y \mid y_1 = \sum_{i=1}^n c_i x_i, x \in \mathcal{X}\}$. Notice that $\hat{\mathcal{X}}$ is still a convex body. If $x^* = \arg \min_{\mathcal{X}} c^T x \neq 0$, then let $y = x - x^*$, so equivalently, we have $y^* = \arg \min_{\{x - x^*\}} \hat{c}^T y = 0$.)

We begin by proving the upper bound in **(A2)**, following [48], but correcting several flawed steps. Let

$$\mathcal{X}_s = \mathcal{X} \cap \{x : x_1 = s\}$$

Then by Lemma 7 we have

$$vol(\mathcal{X}_s)\frac{\lambda}{\pi^{n/2}} \geq \lambda vol(\mathcal{X}_s)\frac{\lambda}{\pi^{n/2}} + (1 - \lambda)vol(\mathcal{X}_s)\frac{\lambda}{\pi^{n/2}}$$

for $s = \lambda s_1 + (1 - \lambda)s_2, \lambda \in [0, 1]$. Now define $\mathcal{X}_0$ obtained by replacing each $\mathcal{X}_s$ by an $(n-1)$-dimensional ball $B_s$ of the same volume and centered at the point $[s 0 \ldots 0]^T$, as shown in Figure 3.1. Then $vol(\mathcal{X}_0) = vol(\mathcal{X})$ and

$$vol(B_s)\frac{\lambda}{\pi^{n/2}} \geq \lambda vol(B_s)\frac{\lambda}{\pi^{n/2}} + (1 - \lambda)vol(B_s)\frac{\lambda}{\pi^{n/2}} \quad (A4)$$

Note that the volume of a $n$-dimensional ball $B$ with radius $r$ is

$$vol(S) = \frac{\pi^{n/2}}{\Gamma(n/2 + 1)} r^n.$$
where $\Gamma$ is Euler’s gamma function. Hence, if we denote by $r(s)$ the radius of the $(n-1)$-dimensional ball $B$ at $s$, then (A4) implies that $r(\lambda s_1 + (1-\lambda)s_2) \geq \lambda r(s_1) + (1-\lambda)r(s_2)$. This, in turn, implies that $r(s)$ is a concave function; thus $X_0$ is a convex set (intuitively, think of the shape of $X_0$). Note that now $X_0$ is symmetric about the $x_1$ axis.

As a second step, define now the cone $K$ with base area $S = \frac{\pi}{n}\text{vol}(X)$, the axis directed along $c$ and located as shown in Figure 3.2, with $h$ being the height of $K$. Then, by construction, we have $\text{vol}(K) = \text{vol}(X) = \text{vol}(X_0)$. Let $s^*$ be the coordinate at which the sets $X_0$ and $K$ intersect; see Figure 3.2. Next, for every $s \in [0,h]$, define the sets $X^+(s) = \{x \in X : x_1 \geq s\}$, $X_0^+(s) = \{x \in X_0 : x_1 \geq s\}$ and $K^+(s) = \{x \in K : x_1 \geq s\}$ as shown in Figure 3.3. Then the following chain of inequalities holds:

\[
P\{f^{(i)} \geq s\} = \frac{\text{vol}(X^+(s))}{\text{vol}(X)} = \frac{\text{vol}(X_0^+(s))}{\text{vol}(X_0)} \\
\leq \frac{\text{vol}(K^+(s))}{\text{vol}(K)} = 1 - \frac{\text{vol}(K^-)(s)}{\text{vol}(K)} = 1 - \left(\frac{s}{h}\right)^n = \frac{h^n - s^n}{h^n}.
\]

(A5)

where the last inequality follows from the fact that, for $s \geq s^*$,

$$\frac{\text{vol}(X_0^+(s))}{\text{vol}(X_0)} \leq \frac{\text{vol}(K^+(s))}{\text{vol}(K)},$$

and, for $s \leq s^*$,

$$\frac{\text{vol}(X_0^+(s))}{\text{vol}(X_0)} = 1 - \frac{\text{vol}(X_0^-)(s)}{\text{vol}(X_0)} \leq 1 - \frac{\text{vol}(K^-)(s)}{\text{vol}(X_0)} = \frac{\text{vol}(K^+(s))}{\text{vol}(K)},$$

where $X_0^-(s) = \{x \in X_0 : x_1 < s\}$ and $K^-(s) = \{x \in K : x_1 < s\}$.

As a final step for proving the upper bound in (A2), notice that $f_{[1]}$ is a positive random variable, and hence we may write

\[
\mathbb{E}[f_{[1]}] = \int_{0}^{h} \mathbb{P}\{f_{[1]} \geq s\} ds = \int_{0}^{h} \left(\mathbb{P}\{f^{(i)} \geq s\}\right)^N ds \leq \int_{0}^{h} \left(\frac{h^n - s^n}{h^n}\right)^N ds \leq \frac{h}{n} B\left(\frac{h}{n}\right) \left(N + 1, \frac{1}{n}\right). \tag{A5}
\]

Finally, applying Theorem 3.4 in [104], we get the following bounds

\[
1 - \left(\frac{N}{N + 1}\right)^{\frac{1}{n}} \leq \frac{1}{n} B\left(\frac{h}{n}\right) \left(N + 1, \frac{1}{n}\right) \leq \left(\frac{1}{N + 1}\right)^{\frac{1}{n}},
\]

thus proving the inequality in (A3), with the cone $K$ being the attainable “worst-case” configuration for $X$.

The lower bound in (A2) can be proved similarly. Namely, instead of the $K$ above, consider the “inverted” cone; then with reasonings identical to those above, it proves to be the ”best case” configuration. We hence derive the following inequality:

\[
\mathbb{P}\{f^{(i)} \geq s\} \geq \frac{s^n}{h^n}. \tag{A6}
\]

Therefore, we obtain

\[
\mathbb{E}[f_{[1]}] = \int_{0}^{h} \mathbb{P}\{f_{[1]} \geq s\} ds = \int_{0}^{h} \left(\mathbb{P}\{f^{(i)} \geq s\}\right)^N ds \geq \int_{0}^{h} \left(\frac{s^n}{h^n}\right)^N ds = \frac{h}{nN + 1},
\]

which concludes our proof.
Corollary 9. Let \( X \subset \mathbb{R} \) be a convex body. Define \( h, f^* \), and \( f_{[1]} \) as in Theorem 8. Then it holds that
\[
E[(f_{[1]} - f^*)^2] \leq \frac{2h^2}{n} B \left( N + 1, \frac{2}{n} \right) \leq h^2 \left( \frac{1}{N+1} \right)^{\frac{2}{n}}. \tag{A7}
\]

Proof. Assume again, without loss of generality, that \( f^* = 0 \) and \( c = [1 \ldots 0]^T \). Then \( f_{[1]} \) is a positive random variable; hence, we may write
\[
E[f_{[1]}^2] = \int_0^{h^2} \mathbb{P}\{f_{[1]} \geq s\} ds = \int_0^{h^2} \mathbb{P}\{f \geq \sqrt{s}\} ds = \int_0^{h^2} \left( \mathbb{P}\{f^{(i)} \geq \sqrt{s}\} \right)^N ds
\]

[from \( \text{(A5)} \)]
\[
\left[ t = \frac{s^{\frac{2}{n}}}{h^n} \right] = \frac{2h^2}{n} \int_0^1 (1-t)^{\frac{N}{2}-1} dt = \frac{2h^2}{n} B \left( N + 1, \frac{2}{n} \right).
\]

Appendix B: Expected convergence rate of RCP

At step \( k \) of the RCP algorithm, define the following random variable:
\[
f_k = c^T z_k. \tag{B1}
\]
Then the following corollary of Theorem 3.1 shows that the RCP scheme converges in first and second mean and, more importantly, that the rate of convergence is exponential.

Theorem 10 (Expected convergence of RCP). Consider the RCP algorithm with \( N_k \equiv N \). Then we have
\[
E[f_k - f^*] \leq \left( \frac{1}{N+1} \right)^{\frac{k}{n}} E[f_0 - f^*] \tag{B2}
\]
that is, the RCP algorithm converges in mean with rate \( \left( \frac{1}{N+1} \right)^{\frac{k}{n}} \). Moreover, the RCP algorithm converges also in mean square with
\[
E[(f_k - f^*)^2] \leq \left( \frac{1}{N+1} \right)^{\frac{2k}{n}} E[(f_0 - f^*)^2]. \tag{B3}
\]

Remark 11. From inequality \( \text{(B2)} \) in the corollary we see that the expected number of steps required by the RCP algorithm to compute an \( \alpha \)-optimal solution (i.e., such that \( E[f_k - f^* \leq \alpha] \)) is at most
\[
k = \left\lceil \frac{1}{\ln(N+1)} \frac{n \ln R}{\alpha} \right\rceil,
\]
where \( R = E[f_0 - f^*] \). Interestingly, when \( N = 1 \), \( \text{(B2)} \) becomes inequality \( \text{(2.3)} \). Indeed, when \( N = 1 \),
\[
\frac{1}{n} B \left( 2, \frac{1}{n} \right) = \frac{n}{n+1}
\]
and
\[
E[f_k] = c^T cg(X_k).
\]
Using \( \text{(A2)} \) and the proof of Corollary \( \text{[10]} \) (note that \( h = f_{k-1} - f^* \) in the proof), we obtain
\[
E[f_k - f^*] \leq \frac{n}{n+1} E[f_{k-1} - f^*]
\]
and if \( g_k \equiv c^T cg(X_k) \) and \( g^* \equiv f^* \), we have
\[
g_k - g^* \leq \frac{n}{n+1} (g_{k-1} - g^*),
\]
which is nothing but \( \text{(2.3)} \). Hence, we conclude that the derived convergence rate in Corollary \( \text{[10]} \) reduces to the one in Lemma \( \text{[4]} \) when \( N = 1 \), while it improves by a factor of \( \ln(N+1) \) when \( N > 1 \).
Appendix C: Pseudocode

(a) **Initialization.** We solve the following auxiliary problem

\[
\min \gamma \text{ s.t. } F(x) \preceq \gamma I.
\]

Note that \( \{ x = 0, \gamma = \max \text{eig}(F(0)) \} \) is a feasible solution, therefore we can solve for the optimal solution \((x^*, \gamma^*)\). If \( \gamma^* > 0 \), then \([\text{LMI}]\) is infeasible; otherwise, take \( x_0 = x^* \) as initial feasible point for \([\text{LMI}]\).

(b) **Main algorithm** (RCP_H&R)

1: **Input:** \( \mathcal{X}, M, N, x_0 \in \mathcal{X} \)
2: **Output:** \( z_k \)
3: \( k = 0, X_0 = \mathcal{X}, P_0 = \{ \}, Y_0 = I_{n \times n}, x_0^Z = x_0, z_0' = x_0 \)
4: **for** \( j = 1 \) to \( N \) **do**
5: \( (x_{k,j}, y_{k,j}, \bar{x}_{k,j}) = \text{H&R}_\text{SDP}(\mathcal{X}, Y_k, M, x_{k,j-1}^Z, z_k') \)
6: \( P_k = P_k \cup \{ (x_{k,j}, \bar{x}_{k,j}, z_{k,j}) \} \)
7: **end for**
8: \( (z_k, \bar{z}_k, \bar{z}_k) = \arg \min_{(x,\bar{z},\bar{z}) \in P_k} c^T x \)
9: \( (z_k', \bar{z}_k', \bar{z}_k') = \arg \min_{(x,\bar{z},\bar{z}) \in P_k \setminus \{ (z_k, \bar{z}_k, \bar{z}_k) \}} c^T x \) (second best minimum)
10: \( z_{k+1}^Z = z_k \) (initial point for \( X_{k+1} \): \( z_k \in X_{k+1} \), and \( z_k \) is ensured in \( \text{H&R}_\text{SDP} \) to stay in the interior of \( \mathcal{X} \))
11: Calculate affine transformation matrix \( Y_{k+1} \):

\[
\bar{y} = \frac{1}{2N} \sum_{i=1}^N (z_{k,i} + \bar{z}_{k,i}), \quad Y_{k+1} = \frac{1}{2N} \sum_{i=1}^N \left[ (z_{k,i} - \bar{y})(\bar{z}_{k,i} - \bar{y})^T + (\bar{z}_{k,i} - \bar{y})(\bar{z}_{k,i} - \bar{y})^T \right]
\]

12: \( X_{k+1} = \{ x \in X_k : c^T (x - z_k^Z) \leq 0 \} \)
13: check Stopping Rule; \( k \leftarrow k + 1 \); go to 4.

(c) **H&R Algorithm for SDP** (H&R_SDP)

1: **Input:** \( \mathcal{X}, Y, M, x^0 \) (starting point), \( z' \) (cutting point defining the input convex body)
2: **Output:** \( (x, \bar{x}, \bar{x}) \), where \( x \) and \( \bar{x} \) are two intersection points on the boundary of the input convex body defined by \( z' \), and \( x \) is a uniform random point drawn from the line segment \([\bar{x}, \bar{x}]\)
3: \( y^0 = x^0 \)
4: **for** \( i = 0 \) to \( M - 1 \) **do**
5: generate a uniformly distributed random direction \( \eta \in \mathbb{R}^n \) on the unit sphere and apply affine transformation \( Y \) to obtain \( v = Y \hat{x} \eta \)
6: \( (\bar{x}, \bar{x}) = \text{BQ}_\text{SDP}(\mathcal{X}, y^0, v, z') \)
7: generate a uniformly distributed point \( y^{i+1} \) in the line segment \([\bar{x}, \bar{x}]\)
8: **while not** \( F(y^{i+1}) < 0 \) **repeat**
9: **step** 5, 6, 7
10: **end while**
11: (Remark: with an additional count variable, may the above be used as a stopping criterion for RCP_H&R ?)
12: **end for**
13: \( x = y^M \); **return** \( (x, \bar{x}, \bar{x}) \)

(d) **Boundary Oracle** (BO)

1: **Input:** \( \mathcal{X}, y \) (current point), \( v \) (random direction), \( z' \) (cutting point defining the input convex body)
2: Output: \((\vec{x}, \vec{x})\), the intersection points between line \(z = y + \lambda v\) and the boundary of the convex body defined by \(z\).

3: \((\vec{x}, \vec{x}) = BO_{SDP}(\mathcal{X}, y, v)\) (see Lemma 5)

4: if \(c^T (\vec{x} - z') > 0\) then solve for \(\lambda:\ c^T (y + \lambda v - z') = 0; \vec{x} = y + \lambda v\)

5: elseif \(c^T (\vec{x} - z') > 0\) then solve for \(\bar{\lambda}: c^T (y + \bar{\lambda}v - z') = 0; \bar{x} = y + \bar{\lambda}v\)

6: else break

7: return \((\vec{x}, \vec{x})\)

(c) Data

\(F_i\) are generated to guarantee non-emptiness of \(\mathcal{X}_{LMI}\), and \(F_0\) is generated such that \(F_0 \prec 0:\)

I.

\[
M = 2 \text{rand}(m) - 1; F_0 = -M \times M^T - \text{eye}(m)
\]

II.

\[
M = 2 \text{rand}(\frac{m}{2}) - 1; M = M + M^T; F_i = \text{blkdiag}(M; -M)
\]

III. Worst-case geometry:

\[
\mathcal{X}_{LMI} = \{x \in \mathbb{R}^n : \|x\|_1 \leq 1, x_1 < 0\}
\]

Appendix D: Details of the Implementation

For simplicity we compute the initial solution using SCS solver from CVXPY package. Namely, we iterate SCS until any feasible solution emerged, does not matter how far it is from the reference value of objective function. This kind of solution serves as an initial one in all our simulations. Alternatively, one can use the approach suggested in [47], Section 6.1 but it would take even longer simulation time.

We model noisy quantum eigensolver by adding noise to the exactly computed eigenvalues. We have considered two noise models. In both models, the noise was defined by its signal to noise ratio (SNR), expressed in dB. Here “signal” is an absolute value of generalised eigenvalue (Lemma 5 in the main text) and “noise” is its disturbance. In this account, small SNR means strong noise and poor estimation of eigenvalues, while high SNR implies that reliable estimates of the eigenvalues are available.

The first noise model is multiplicative. It respects the spectrum of the generalised eigenproblem in the sense that every eigenvalue \(\lambda_i\) is disturbed in proportion to its amplitude:

\[
\lambda_i^{\text{noisy}} = \lambda_i^{\text{exact}} \left(1 + \frac{\varepsilon}{10^\left(SNR/20\right)}\right), \quad \varepsilon \sim \mathcal{N}(0, 1).
\]

Fig. 2 shows convergence profiles for various noise levels.

The second noise model is additive. It adds random Gaussian noise scaled by root-mean-square eigen-value to all others:

\[
\lambda_i^{\text{noisy}} = \lambda_i^{\text{exact}} + \varepsilon \sqrt{\frac{1}{10^2} \sum_{i=1}^{N} (\lambda_i^{\text{exact}})^2}, \quad \varepsilon \sim \mathcal{N}(0, 1).
\]

Mind denominator in \(SNR/10\) expression, where 10 comes from the fact that we operate on squared “signal” \((\lambda_i^2)\), as opposed to the first model, where just a “signal” amplitude is involved. Since eigenvalues typically differ in their orders of magnitude, the algorithm does not converge to the desired minimum \((\approx -380\) for the \texttt{qap6} problem) for high noise levels (low SNR values). The Fig. 3 demonstrates early termination in many cases.
a. *Experimental setup*  We have been using SDPLib, the standard benchmark in SDP. Not all the instances were suitable for the RCP-based solver, which is slow, comparing to classical ones. Out of the full list of 80+ instances, we picked up a few “good” ones, according to the following simple criteria:

- primal problems only.
- those solved successfully by SCS solver with final objective function close to the reference one (provided by SDPLib).
- small enough, so as to be fast to solve: namely, those which SCS solver managed to solve in less than 200 seconds.
The list of “good” instances appears in the left-most column of the table of results. Note that not every “good” instance was actually solved by RCP method because of the size. One can hence notice that the final value of the objective function is sometimes substantially different from the reference one, although the timeout parameter was 86400 seconds. Table VII lists all the results obtained for 1 day timeout (86400 seconds). Namely, we interrupt the execution and return solution reached so far as soon as the timeout has been exceeded but simulation is still running. On some instances, the RCP method converged close to the reference value. Table VII lists all similar results obtained for 7 day timeout (604800 seconds). A few instances did not make any substantial progress beyond the very first iteration of the hit & run algorithm because of their large size, such as theta6, where initial and final objective function values are the same.

One possibility we have not explored yet, would be to employ sparse matrices for very large problems.

All numerical experiments have been conducted on a server equipped with 44 cores / 88 hardware threads of Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz, hundreds of gigabytes of memory, using RedHat 7 OS and Anaconda environment with Python 3.9 and Numpy, Scipy, CVXPY packages installed. The entire code was written in Python.

During our experimentation, we noticed that enabling multi-threading actually significantly slows down the RCP solver. We attribute this behaviour to some software issues, which we did not investigate in depth. Instead, we run all the instances in a separate process without multi-threading, utilising a single CPU core per problem.

b. Implementation details In practice, we faced certain challenges trying to implement RCP as presented. A few modifications in the main algorithms have been made to address those issues:

(a) In the current implementation, the boundary points are computed by the classical generalised eigensolver from Scipy package, which expects a positive definite matrix $B$ (as in Lemma 5), where the problem $A v_i = -\lambda_i B v_i$ is considered. The (imperfect) quantum eigensolver is modelled by adding an artificial noise to the eigenvalues. In our case, the matrix $A$ is negative (semi)definite, while $B$ is sign indefinite. We actually solve the problem $\mu_i (-A) v_i = B v_i$ and then take the reciprocal $\lambda_i = 1/\mu_i$. Moreover, when $A$ is close to semi-definiteness, the eigensolver becomes unstable. If that case, we add a tiny value to diagonal elements of $A$ in order to make it strictly positive definite and repeat the eigensolve one more time.

(b) In Section C, the pseudocode of Algorithm H&R in Line 5 performs the isotropization via computation of a square root of matrix $Y$. The same result can be obtained faster via Cholesky decomposition, which is actually done in our code.

(c) In Line 7 of the same algorithm (H&R SDP), a point $y^{(i+1)}$ is generated inside the segment $[\bar{x}, \bar{x}]$. In our implementation, we prevent the point from taking end values by a small margin (about 0.001 of the segment size).

(d) In Lines 8 to 10 of the same algorithm (H&R SDP), the loop is repeated a number of times (by default up to 200 attempts before we claim no further improvements can be done). Instead of generating a new couple of boundary points $\bar{x}, \bar{x}$ in every iteration of the inner loop, which is very expensive, we repeat Line 7 five times. If $y^{(i+1)}$ remains infeasible, Line 6 is activated.

(e) In the same algorithm (H&R SDP), on Line 3 instead of using the same starting point $x^{(0)}$, we use the last feasible $y$ obtained after previous invocation of H&R SDP algorithm.

(f) The mixing time $M$ is equal to 10 in all simulations. The number $N$ of samples generated in every outer iteration is equal to the number of variables multiplied by 100. The latter factor is accountable for the slow performance on large problem instances, although the big number of samples provides better convergence rate.

Sometimes, a slightly deeper minimum can be attained without steps (c) to (e), but this depends on randomisation. For the hard instances, these steps bring a noticeable improvement of convergence.

c. Observations For the Gap6 instance, the time spent by the different modules is distributed as follows (as percentage of the total execution time):

- Computation of $F(x) = F_0 + \sum_{i=1}^{n} x_i F_i$: 41 \%
- Computation of generalised eigen-values: 22.3 \%
- Cholesky decomposition: 6 \%
- Generation of random vectors: 5.2 \%
- Other subroutines: 25.5 \%

Cholesky decomposition is used to check feasibility of the constraint $F(x) \leq 0$ and for isotropization. Surprisingly, the computation of $F(x)$ dominates the total run time, although it is implemented very rationally (one line of Python code) with full utilisation of Numpy optimised backend.
FIG. 4. Here we demonstrate the process of shrinking of convex body volume over iterations. The volume is roughly proportional to determinant of the covariance matrix of a cloud of sampled points. Since determinant of high-dimensional covariance matrix is either very big or very small, a better visual experience can be drawn from the behaviour of extreme eigen-values as well as the mean one. The data were obtained for qap6 problem.

\[d. \text{Volume shrinkage}\]

It might be insightful to see how the volume of convex body is shrinking as iteration process progresses. Again, we selected the qap6 instance for demonstration purposes. This particular instances is challenging (sensitive to sampling scheme), but large enough and solvable in a reasonable time. Fig. 4 shows minimum, maximum and mean eigenvalues of the covariance matrix of a cloud of points sampled on every iteration of hit&run algorithm.
### Table VI

An overview of the behaviour of RCP on a subset of smaller problems from SDPLIB, a well-known benchmark, within a 24-hour time limit (86400 seconds): Instance name, dimensions (first is the number of constraint matrices), reference objective function value, objective function value at termination of RCP, initial objective function at the beginning of RCP, DIMACS Error 1 [92, 93] at termination, DIMACS Error 2 [92, 93] at termination, and run-time in seconds.

| Instance | size           | Ref. | RCP(T) | Initial | Err1(T) | Err2(T) | Time [s] |
|----------|----------------|------|--------|---------|---------|---------|----------|
| gpp100   | 101x100x100    | -44.94 | -44.94 | -18.40  | 950.45  | 0.0     | 86400.32 |
| gpp124-1 | 125x124x124    | -7.34  | -6.34  | 18.08   | 152.92  | 0.0     | 86400.30 |
| gpp124-2 | 125x124x124    | -46.86 | -45.77 | -19.86  | 718.02  | 0.0     | 86400.26 |
| gpp124-3 | 125x124x124    | -153.01| -150.81| -97.13  | 815.00  | 0.0     | 86400.36 |
| gpp124-4 | 125x124x124    | -418.99| -407.43| -267.03 | 1551.43 | 0.0     | 86400.34 |
| hinf1    | 13x14x14       | 2.03   | 2.09   | 2.25    | 7.30    | 0.0     | 84.31    |
| hinf10   | 21x18x18       | 108.71 | 122.41 | 151.07  | 4998.56 | 0.0     | 294.37   |
| mcp100   | 100x100x100    | 226.16 | 226.27 | 318.12  | 0.11    | 0.0     | 86400.23 |
| mcp124-1 | 124x124x124    | 141.99 | 160.95 | 250.15  | 0.08    | 0.0     | 86400.27 |
| mcp124-2 | 124x124x124    | 269.88 | 282.22 | 383.00  | 0.11    | 0.0     | 86400.25 |
| mcp124-3 | 124x124x124    | 467.75 | 486.34 | 614.28  | 0.15    | 0.0     | 86400.24 |
| mcp124-4 | 124x124x124    | 864.41 | 901.95 | 1120.64 | 0.23    | 0.0     | 86400.73 |
| mcp250-2 | 250x250x250    | 531.93 | 741.00 | 758.01  | 0.12    | 0.0     | 86400.63 |
| mcp250-3 | 250x250x250    | 981.17 | 1238.95| 1272.45 | 0.17    | 0.0     | 86400.76 |
| mcp250-4 | 250x250x250    | 1681.96| 2100.18| 2155.74 | 0.26    | 0.0     | 86400.76 |
| qap5     | 136x26x26      | -436.00| -355.98| 256.19  | 82.88   | 0.0     | 12641.77 |
| qap6     | 229x37x37      | -381.44| -379.29| 187.39  | 20.77   | 0.0     | 86400.19 |
| qap7     | 358x50x50      | -424.82| -178.42| 290.45  | 10.08   | 0.0     | 86400.27 |
| qap8     | 529x65x65      | -756.96| -12.57 | 420.27  | 8.71    | 0.0     | 86400.32 |
| qap9     | 748x82x82      | -1409.94| -38.38 | 745.86  | 10.78   | 0.0     | 86400.40 |
| theta1   | 104x50x50      | 23.00  | 23.00  | 53.33   | 106.62  | 0.0     | 5967.03  |
| theta2   | 498x100x100    | 32.88  | 145.51 | 152.09  | 826.16  | 0.0     | 86400.38 |
| theta3   | 1106x150x150   | 42.17  | 82.34  | 82.34   | 568.98  | 0.0     | 86401.44 |
| theta4   | 1949x200x200   | 50.32  | 147.05 | 147.05  | 1186.88 | 0.0     | 86403.63 |
| theta5   | 3028x250x250   | 57.23  | 255.70 | 255.70  | 2327.25 | 0.0     | 86407.97 |
| theta6   | 4375x300x300   | 63.48  | 410.29 | 410.29  | 4106.66 | 0.0     | 86420.12 |
| truss1   | 6x13x13        | -9.00  | -9.00  | -7.10   | 5.90    | 0.0     | 17.30    |
| truss2   | 58x133x133     | -123.38| -123.00| -21.46  | 75.54   | 0.0     | 86400.21 |
| truss3   | 27x31x31       | -9.11  | -9.11  | -5.83   | 4.78    | 0.0     | 794.89   |
| truss4   | 12x19x19       | -9.01  | -9.00  | -5.83   | 5.38    | 0.0     | 89.30    |
TABLE VII. An overview of the behaviour of RCP on a subset of smaller problems from SDPLIB, a well-known benchmark, within a week long time limit (604800 seconds): Instance name, dimensions (first is the number of constraint matrices), reference objective function value, objective function value at termination of RCP, initial objective function at the beginning of RCP, DIMACS Error 1 [92, 93] at termination, DIMACS Error 2 [92, 93] at termination, and run-time in seconds.

| Instance | size       | Ref. | RCP(T) | Initial | Err1(T) | Err2(T) | Time [s] |
|----------|------------|------|--------|---------|---------|---------|----------|
| gpp100   | 101x100x100| -44.94 | -44.94 | -18.40  | 950.46  | 0.0     | 82692.28 |
| gpp124-1 | 125x124x124| -7.34  | -7.34  | 18.08   | 212.81  | 0.0     | 205997.33|
| gpp124-2 | 125x124x124| -46.86 | -46.86 | -97.13  | 862.31  | 0.0     | 153632.17|
| gpp124-3 | 125x124x124| -153.01| -153.01| -18.40  | 950.46  | 0.0     | 149677.75|
| gpp124-4 | 125x124x124| -418.99| -418.99| -267.03 | 1667.97 | 0.0     | 164294.46|
| hin1     | 13x1x14    | 2.03   | 2.09   | 2.25    | 7.30    | 0.0     | 60.85    |
| hin10    | 21x18x18   | 108.71 | 122.41 | 151.07  | 4998.56 | 0.0     | 240.51   |
| mcp100   | 100x100x100| 226.16 | 226.16 | 318.12  | 0.11    | 0.0     | 74340.28 |
| mcp124-1 | 124x124x124| 141.99 | 141.99 | 250.15  | 0.07    | 0.0     | 188645.49|
| mcp124-2 | 124x124x124| 269.88 | 269.88 | 383.00  | 0.10    | 0.0     | 178055.59|
| mcp124-3 | 124x124x124| 467.75 | 467.75 | 614.28  | 0.14    | 0.0     | 171032.22|
| mcp124-4 | 124x124x124| 864.41 | 864.41 | 1120.64 | 0.20    | 0.0     | 168804.84|
| mcp250-2 | 250x250x250| 531.93 | 579.59 | 758.01  | 0.08    | 0.0     | 604800.46|
| mcp250-3 | 250x250x250| 981.17 | 1036.79| 1272.45 | 0.12    | 0.0     | 604800.48|
| mcp250-4 | 250x250x250| 1681.96| 1775.46| 2155.74 | 0.17    | 0.0     | 604800.49|
| qap5     | 136x26x26  | -436.00| -435.98| 256.19  | 82.88   | 0.0     | 11396.36 |
| qap6     | 229x37x37  | -381.44| -379.67| 187.39  | 20.75   | 0.0     | 127062.87|
| qap7     | 358x50x50  | -424.82| -423.03| 290.45  | 11.47   | 0.0     | 410249.13|
| qap8     | 529x65x65  | -756.96| -744.99| 420.27  | 10.04   | 0.0     | 604800.31|
| qap9     | 748x82x82  | -1409.94| -965.46| 745.86  | 11.90   | 0.0     | 604800.37|
| theta1   | 104x50x50  | 23.00  | 23.00  | 53.33   | 106.62  | 0.0     | 5988.24  |
| theta2   | 498x100x100| 32.88  | 50.36  | 152.09  | 287.07  | 0.0     | 604800.34|
| theta3   | 1106x150x150| 42.17 | 76.34  | 82.34   | 531.18  | 0.0     | 604800.62|
| theta4   | 1949x200x200| 50.32 | 143.99 | 147.05  | 1188.87 | 0.0     | 604802.40|
| theta5   | 3028x250x250| 57.23 | 255.70 | 2327.25 | 0.0     | 604805.45|
| theta6   | 4375x300x300| 63.48 | 410.29 | 410.29  | 4106.66 | 0.0     | 604811.28|
| truss1   | 6x13x13    | -9.01 | -9.00  | -7.10   | 5.90    | 0.0     | 16.35    |
| truss2   | 58x13x133 | -123.38| -123.00| -21.46  | 75.54   | 0.0     | 57755.23 |
| truss3   | 27x31x31   | -9.11 | -9.11  | -5.23   | 4.78    | 0.0     | 684.62   |
| truss4   | 12x19x19   | -9.01 | -9.00  | -5.83   | 5.38    | 0.0     | 81.22    |