Proceedings of the
47th IEEE Conference on Decision and Control
Cancun, Mexico, Dec. 9-11, 2008



# A Randomized Cutting Plane Scheme with Geometric Convergence: Probabilistic Analysis and SDP Applications

F. DABBENE, P. SHCHERBAKOV, B.T. POLYAK

*Abstract*— **We propose a randomized method for general convex optimization problems; namely, the minimization of a linear function over a convex body. The idea is to generate $N$ random points inside the body, choose the best one and cut the part of the body defined by the linear constraint. The method is analyzed under the assumption that a Uniform Generating Oracle (UGO) is available — an algorithm for uniform generation of random points in the convex body. The expected rate of convergence for such method is geometric. Probabilistic estimates of convergence are explicitly derived: we estimate the probability that the method converges slower than the deterministic center-of-gravity algorithm. Since UGO is unavailable in most applications, a practical implementation of the method is discussed, based on Hit-and-Run versions of Markov-chain Monte Carlo algorithms. The crucial notion for this algorithm is a Boundary Oracle, which is available for most optimization problems, including LMIs and many other types of constraints. Numerical results for SDP problems are presented, which confirm that the randomized approach can be a competitor to modern deterministic interior-point algorithms.**

## I. INTRODUCTION

Recent years exhibited the growing interest to randomized algorithms in control and optimization; e.g., see [18]. There are numerous reasons for this interest, from philosophical to computational ones. The present paper continues this line of research for convex optimization. We consider a convex minimization problem of the form

$$\min c^\top x \quad \text{subject to} \quad x \in \mathcal{X}, \tag{1}$$

where the set $\mathcal{X} \subset \mathbb{R}^n$ is a *convex body*, i.e., it is full-dimensional and bounded. The linear cost function is taken without loss of generality; any convex optimization problem can be reduced to this form.

In Section II, we start with the description of deterministic Center-of-Gravity (DCG) method. This method has been proposed simultaneously by Levin and Newman in 1965 [7], [11] for a slightly different problem formulation. Its rate of convergence has been estimated via Grünbaum theorem [6] and happened to be geometric. We provide another result based on Radon theorem [15] which also guarantees geometric convergence. However, despite its very attractive theoretical behavior, the DCG algorithm turned out to be not implementable in practice, for a very simple reason: *for a generic convex set, computing the center of gravity is more difficult than solving the original optimization problem*. That is why, in Section III we propose a Random Cutting Plane (RCP) algorithm based on the assumption that it is possible to generate points uniformly in a convex body; this mechanism is referred to as Uniform Generation Oracle (UGO). We then report a result of [5] that provides an estimate of the expected rate of convergence of the RCP algorithm.

Section IV represents one of the main contributions of the paper; namely, for the first time, a probabilistic analysis of RCP algorithm is provided. In particular, we estimate the probability that RCP converges slower than DCG and calculate the number $N$ of points to be generated in RCP to guarantee a better convergence with high probability.

The main drawback of the RCP is the nonrealistic assumption on the existence of UGO. Our implementation of the algorithm is based on Hit-and-Run (H&R) versions of the Monte Carlo method, which are aimed at *approximately uniform* generation of points in a body via random walks. An H&R algorithm has been applied to convex optimization in [1]; however, the method in [1] differs from the implementable version of RCP presented in Section V. First, the problem formulation differs from (1), and it is this difference that allows us to introduce the concept of *best point*. Second, the method proposed in [1] updates at each step an outer polytope inscribing $\mathcal{X}$, while here we update directly the set $\mathcal{X}$ (in this sense, our method can be considered an interior-point algorithm). Third, in [1], the authors use a so-called Separation Oracle, while we exploit Boundary Oracle (see details below). Results of numerical experiments with the proposed algorithm for semidefinite programming problems are described in Section VI.

## II. DETERMINISTIC CENTER OF GRAVITY ALGORITHM

For a convex body $\mathcal{X}$, define its *center of gravity* as $\mathsf{cg}(\mathcal{X}) \doteq \int_{\mathcal{X}} x\mathrm{d}x / \int_{\mathcal{X}} \mathrm{d}x$. Then, the deterministic cutting plane method based on recursive cutting through the center of gravity can be stated as follows, see also [5], [13].

**Algorithm 1** DCG Algorithm

**Input:** $\mathcal{X}$,
**Output:** optimum $x^*$ for (1)
1: $k \Leftarrow 0$, $\mathcal{X}_k \Leftarrow \mathcal{X}$
2: $x_k \Leftarrow \mathsf{cg}(\mathcal{X}_k)$
3: $\mathcal{X}_{k+1} \Leftarrow \{x \in \mathcal{X}_k : c^\top(x - x_k) \leq 0\}$
4: check Stopping Rule; goto 2.

In words, the method works as follows: Let $\mathcal{X}_0 \equiv \mathcal{X}$, and let $x_0$ be the center of gravity of $\mathcal{X}_0$. Then, proceed by con-

This work was supported by a bilateral project funded by Italian CNR and Russian Academy of Sciences and by a CNR RSTL grant.

F. Dabbene is with IEIIT-CNR, Politecnico di Torino, Italy; fabrizio.dabbene@polito.it

P. Shcherbakov and B.T. Polyak are with IPU, Moscow, Russia; (sherba,boris)@ipu.ru

sidering the new set $\mathcal{X}_1 = \{x \in \mathcal{X}_0 \colon c^\top(x - x_0) \le 0\} \subset \mathcal{X}_0$ obtained by cutting off a portion of $\mathcal{X}_0$ using the hyperplane $\mathcal{H}_0 = \{x \in \mathbb{R}^n : c^\top(x - x_0) = 0\}$. For this convex set $\mathcal{X}_1$ in turn, compute its center of gravity $x_1 = \mathsf{cg}(\mathcal{X}_1)$ and construct the hyperplane $\mathcal{H}_1$ through $x_1$, which defines the set $\mathcal{X}_2 \subset \mathcal{X}_1$, etc. As a result, we obtain a sequence of embedded sets $\mathcal{X}_k \subset \mathcal{X}_{k-1} \subset \cdots \subset \mathcal{X}_1 \subset \mathcal{X}_0$ and a sequence of points $x_k$ having the property $c^\top x_{k+1} < c^\top x_k$. From a classical result by Grünbaum [6], it then follows that

$$\mathrm{vol}\,(\mathcal{X}_{k+1}) \le (1 - 1/\mathrm{e})\mathrm{vol}\,(\mathcal{X}_k), \tag{2}$$

i.e., at each step the DCG algorithm guarantees that a given portion of the set is cut out. Besides that, the method possesses nice convergence properties; specifically, the following estimate for the rate of convergence takes place:

$$f_{k+1} - f^* \le \frac{n}{n+1}(f_k - f^*), \tag{3}$$

where $f_k \doteq c^\top x_k$ and $f^*$ is the optimal value of the cost function. Relation (3) can be proved using the well-known Radon theorem [15], and we formulate this result as the lemma below, see [13], [5] for additional details.

*Lemma 1 (Convergence of DCG):* The DCG algorithm yields an $\alpha$-optimal solution (i.e. such that $f_k - f^* \le \alpha$) in at most $k = \left\lceil \frac{\ln \frac{R}{\alpha}}{\ln \frac{n+1}{n}} \right\rceil = \mathcal{O}(n \ln R/\alpha)$ steps, where $R \doteq f_0 - f^*$.

However, the method has an essential drawback: its practical implementation is hindered by the fact that computing the center of gravity is NP-hard, even for the simple case when $\mathcal{X}$ is a polytope, as recently proved in [14].

## III. A Randomized Cutting Plane Algorithm

In [5], the following randomized modification of the DCG scheme was proposed. The method, formally stated as Algorithm 2, is based on the abstract assumption of the availability of a mechanism for generating uniform samples in convex sets.

**Algorithm 2** RCP Algorithm

**Input:** $\mathcal{X}$,
**Output:** $x^*$

1: $k \Leftarrow 0$, $\mathcal{X}_k \Leftarrow \mathcal{X}$
2: generate $N_k$ uniform samples in $\mathcal{X}_k$, $\{x^{(1)}, \ldots, x^{(N_k)}\} = \mathsf{UGO}(\mathcal{X}_k, N_k)$,
3: $x_k \Leftarrow \arg\min_{x \in \{x^{(1)}, \ldots, x^{(N_k)}\}} c^\top x$,
4: $\mathcal{X}_{k+1} \Leftarrow \{x \in \mathcal{X}_k : c^\top(x - x_k) \le 0\}$
5: check Stopping Rule; go to 2.

*Assumption 1 (Uniform Generating Oracle):* Assume that a *uniform generating oracle* (UGO) is available, such that for any convex set $\mathcal{X} \in \mathbb{R}^n$, a call to $\mathsf{UGO}(\mathcal{X}, N)$ returns $N$ independent random points uniformly distributed in $\mathcal{X}$.

The randomized scheme we propose for computing the query point at step $k$ is very simple: At step $k$, we make recourse to UGO and generate $N_k$ uniform samples in $\mathcal{X}_k$. The next-step query point is selected as the random sample providing the minimum to the cost function, see Figure 1. Clearly, the $k$-th step of the RCP algorithm will perform better than a corresponding step of a deterministic method as long as at least one random point will fall *below* (in terms of the cost function) the center of gravity $x_\mathrm{G}$ of the set $\mathcal{X}_k$.

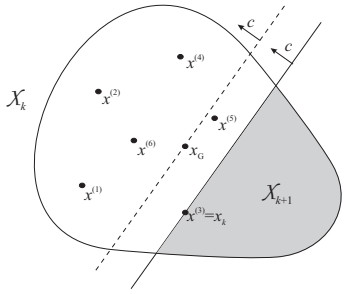


Fig. 1. A sketch of the proposed randomized cutting scheme.

Using the technique similar to the one in [1], it was proved in [5] that the *expected* convergence of the RCP algorithm outperforms the classical DCG one.

*Lemma 2 (Expected convergence of RCP):* The expected number of steps of the RCP algorithm for computing an $\alpha$-optimal solution (i.e. such that $f_k - f^* \le \alpha$) is at most

$$k = \left\lceil \frac{n}{\ln(N+1)} \ln \frac{R}{\alpha} \right\rceil.$$

Notably, the above convergence rate reduces to the one of Lemma 1 when $N = 1$, and improves by a factor of $\ln(N+1)$ when $N > 1$.

## IV. Probabilistic analysis of RCP

We now derive explicit bounds on the number of samples required to guarantee with arbitrarily high *probability* a desired performance of the proposed algorithm. This section constitutes one of the main novel contributions of the paper.

### A. Single step analysis

We first analyze the behavior of a *single step* of the RCP algorithm. To this end, consider the situation at a generic step $k$ and define by $x_\mathrm{G}$ the center of gravity of the set $\mathcal{X}_k$ and by $x_k$ the query point returned by line 3 of Algorithm 2. Formally, define $\mathsf{Worse}_k$ as the event of RCP returning at step $k$ a query point which is worse (i.e. *above* in terms of cost-function value) than $x_\mathrm{G}$. The probability of this latter event can be easily bounded in the following way:

$$\begin{aligned}\mathbb{P}\{\mathsf{Worse}_k\} &= \mathbb{P}\{c^\top x_k \ge c^\top x_\mathrm{G}\} \\ &= \mathbb{P}\{c^\top x^{(i)} \ge x_\mathrm{G}, i = 1, \ldots, N_k\} \\ &= \mathbb{P}\{c^\top x^{(i)} \ge x_\mathrm{G}\}^{N_k} \le (1 - 1/e)^{N_k},\end{aligned}$$

where the last inequality follows from Grünbaum's theorem [6] (cf. (2)). Hence, we have the following lemma showing that, by appropriately selecting the number of samples $N_k$, we can guarantee with a prescribed arbitrarily high probability that the $k$-th step of RCP algorithm performs better than the corresponding DCG step.

*Lemma 3 (Single step analysis of RCP):* Given a probability level $\epsilon > 0$, set $N_k \geq 2.2 \ln \frac{1}{\epsilon}$. Then, with probability at least $1-\epsilon$, it holds

$$c^\top x_k \leq c^\top \mathsf{cg}(\mathcal{X}_k). \tag{4}$$

**Proof.** The lemma is proved noting that $N_k \geq \ln \frac{1}{\epsilon} / \ln \frac{1}{1-1/\mathrm{e}}$ implies $(1-1/e)^{N_k} \leq \epsilon$, and $(\ln \frac{1}{1-1/\mathrm{e}})^{-1} \approx 2.1802 < 2.2$. □

### B. Overall analysis of RCP

We derive a specific bound, based on a Bonferroni-like inequality, on the number of samples required to guarantee (with high probability) that the RCP algorithm will expose the same geometric rate of convergence as that of DCG.

*Theorem 1 (Behavior of RCP):* Given a probability level $\epsilon > 0$, choose

$$N_k \geq N_k{}^{\mathrm{sum}}(\epsilon, k) \doteq 2.2 \ln \frac{1}{\epsilon} + (1.1 + 0.505 \ln k). \tag{5}$$

Then, with probability at least $(1-\epsilon)$, the RCP algorithm will expose a better or equal rate of convergence than DCG.

**Proof.** The probability of the algorithm performing worse than DCG is the probability of the event $\mathsf{Worse}_\infty \doteq \{\text{at some step } k \quad c^\top x_k \geq c^\top \mathsf{cg}(\mathcal{X}_k)\}$. This probability can be bounded as $\mathbb{P}\{\mathsf{Worse}_\infty\} \leq \mathbb{P}\{\mathsf{Worse}_1 \cup \mathsf{Worse}_2 \cup \cdots\} = \sum_{k=1}^{\infty} \mathbb{P}\{\mathsf{Worse}_k\} \leq \sum_{k=1}^{\infty} (1-1/e)^{N_k}$. To make this sum finite and equal to the desired probability level $\epsilon$, it is sufficient to select the sequence $N_k$ such that

$$(1-1/e)^{N_k} = \frac{k^{-\alpha}}{\zeta(\alpha)} \epsilon, \quad \alpha > 1, \tag{6}$$

where $\zeta(\alpha)$ is the Riemann zeta function, for which $\sum_{k=1}^{\infty} k^{-\alpha} = \zeta(\alpha)$. In fact, in this case we have $\sum_{k=1}^{\infty} (1-1/e)^{N_k} = \sum_{k=1}^{\infty} \frac{k^{-\alpha}}{\zeta(\alpha)} \epsilon = \epsilon$. Solving equation (6) with respect to $N_k$, we obtain the bound

$$N_k \geq \frac{\ln \zeta(\alpha) + \alpha \ln k + \ln \frac{1}{\epsilon}}{\ln \frac{1}{1-1/\mathrm{e}}}.$$

Choosing, by trial and error, $\alpha = 1.1$, we get bound (5). □

Table I shows the values of bound (5) for given values of $\epsilon$ and $k$. Note that the number of required random samples is indeed very low, even for very small values of $\epsilon$.

| $k$ \ $\epsilon$ | 0.01 | 0.005 | 0.001 | 0.0005 | 0.00001 |
|---|---|---|---|---|---|
| 1 | 12 | 13 | 17 | 18 | 27 |
| 10 | 13 | 14 | 18 | 19 | 28 |
| 100 | 14 | 16 | 19 | 21 | 29 |
| 1,000 | 15 | 17 | 20 | 22 | 30 |
| 10,000 | 16 | 18 | 21 | 23 | 32 |

TABLE I

THE VALUES OF $\lceil N_k{}^{\mathrm{sum}}(\epsilon, k) \rceil$ FOR VARIOUS VALUES OF $\epsilon$ AND $k$.

## V. A HIT-AND-RUN IMPLEMENTATION OF RCP

Above, the properties of the RCP algorithm have been analyzed in a very abstract setting, which assumed the existence of a mechanism for generating uniform random samples from a set. Although Assumption 1 is rather common in the computational geometry community, it is quite strong and is made only for theoretical analysis purposes. We show here how this assumption can be relaxed.

### A. Boundary Oracle and Hit-and-Run

We replace Assumption 1 with the more reasonable one on the availability of a *boundary oracle*.

*Assumption 2 (Boundary Oracle):* Assume that a *boundary oracle* (BO) is available such that, given a direction $v \in \mathbb{R}^n$ and a point $z$ belonging to a bounded convex set $\mathcal{X} \in \mathbb{R}^n$, a call to $\mathsf{BO}(x, \mathcal{X}, v)$ returns the two points $\overline{z}$, $\underline{z}$ which are the intersection of the 1D line $x + \lambda v$, $\lambda \in \mathbb{R}$, with the boundary of $\mathcal{X}$.

For a wide range of problems, e.g., such as LMI constrained optimization, a BO can be formulated in closed form. Boundary oracle is crucial for obtaining implementable versions of RCP by means of techniques based on *random walks* in convex bodies (see [20] for a detailed survey on these methodologies). A random walk in $\mathcal{X}$ starts at a point in $\mathcal{X}$ and moves to a neighboring point chosen according to a specific randomized rule that depends on the current point only. Many different techniques have been proposed in the literature to choose the next point to walk in; one of the most promising is the Hit-and-Run (H&R) algorithm, which has been proven to posses the best bounds on the number of steps needed to draw a random sample.

This algorithm has been proposed by Turchin [19] and independently later by Smith [17]. Its properties have been studied in numerous works by Lovász and co-authors (e.g., see the survey [20]). The H&R algorithm is recalled next:

**Algorithm 3** H&R Algorithm

**Input:** $x_0 \in \mathcal{X}$, $T$
**Output:** $x^{(i)}$ random point belonging to $\mathcal{X}$

1: $z \Leftarrow x_0$,
2: **for** $j = 1$ to $T$ **do**
3: generate a uniform random direction $v \in \mathbb{R}^n$,
4: $\{\overline{z}, \underline{z}\} = \mathsf{BO}(\mathcal{X}, z, v)$
5: generate a uniform point $z$ in the segment $[\overline{z}, \underline{z}]$,
6: **end for**
7: $x^{(i)} \Leftarrow z$.

With these premises, the abstract setup of the previous section can be practically implemented by simply substituting line 2 of Algorithm 2 with the H&R procedure. This is summarized in Algorithm 4.

**Algorithm 4** RCP Algorithm – H&R implementation

**Input:** $\mathcal{X}$,
**Output:** $x^*$

1: $k \Leftarrow 0$, $\mathcal{X}_k \Leftarrow \mathcal{X}$
2: **for** $j = 1$ to $N_k$ **do** $x^{(j)} \Leftarrow \mathsf{H\&R}(x^{(j-1)}, T)$; **end for,**
3: $x_k \Leftarrow \arg\min_{x \in \{x^{(1)}, \ldots, x^{(N_k)}\}} c^\top x$,
4: $\mathcal{X}_{k+1} \Leftarrow \left\{x \in \mathcal{X}_k \,:\, c^\top (x - x_k) \leq 0\right\}$
5: check Stopping Rule; go to 2.

### B. Probabilistic analysis of the H&R implementation

In this section we show how the probabilistic analysis of Section IV can be extended to the H&R implementations.

This analysis is based on a seminal property of H&R, which was proved in [8]: After a number of steps $T$ (so-called mixing time) which is *polynomial in the dimension* $n$, the total variation[1] between the density $\mathbb{P}_T$ and the uniform density $\mathbb{P}$ is guaranteed to be bounded by a given constant. This result was later refined by various authors; we report here the following lemma rephrased from [9], which provides a polynomial bound on the mixing time.

*Lemma 4 (Mixing rate of H&R):* Let $\mathcal{X}$ be a convex body that contains a ball of radius $r$ and is contained in a ball of radius $R$. Suppose that the initial distribution is concentrated at a starting point $x_0$ at a distance $d$ from the boundary of $\mathcal{X}$, and let $\mathbb{P}_T$ be the distribution of the current point after $T$ steps of H&R in $\mathcal{X}$. Then,

$$\|\mathbb{P}_T - \mathbb{P}\|_{\mathsf{tv}} \leq \frac{R}{d}\mathrm{e}^{-n^{-3}\frac{r^2}{CR^2}T}, \tag{7}$$

where $C$ is an absolute constant.

Notice that the total number of H&R steps in Algorithm 4 turns out to be $T \cdot N_k$ (that is, at least $T$ H&R steps have to be performed between the two consecutive samples). This mixing time is *necessary* to ensure the total variation bound of Lemma 4. With this result in mind, the reasonings that lead to Lemma 3 modify as follows:

$$\begin{aligned}
\mathbb{P}_T\{\mathsf{Worse}_k\} = \;& \mathbb{P}_T\{c^\top x^{(i)} \geq x_{\mathrm{G}}, i = 1, \ldots, N_k\} \\
\text{from (7)} \leq \;& \mathbb{P}\{c^\top x^{(i)} \geq x_{\mathrm{G}}, i = 1, \ldots, N_k\} \\
& + \frac{R}{d}\mathrm{e}^{-n^{-3}\frac{r^2}{CR^2}T} \\
\leq \;& \Big(\mathbb{P}\{c^\top x^{(i)} \geq x_{\mathrm{G}}\}\Big)^{N_k} + \frac{R}{d}\mathrm{e}^{-n^{-3}\frac{r^2}{CR^2}T} \\
\leq \;& (1 - 1/e)^{N_k} + \frac{R}{d}\mathrm{e}^{-n^{-3}\frac{r^2}{CR^2}T}.
\end{aligned}$$

This proves the following lemma showing that we can still guarantee $\mathbb{P}_T\{\mathsf{Worse}_k\} \leq \epsilon$ with arbitrarily high probability.

*Lemma 5 (Single step analysis of RCP with H&R):* Given a probability level $\epsilon > 0$, choose $\epsilon_1$, $\epsilon_2$ such that $\epsilon_1 + \epsilon_2 = \epsilon$ and set

$$T \geq Cn^3\frac{R^2}{r^2}\ln\frac{R}{d\epsilon_1}; \qquad N_k \geq 2.2\ln\frac{1}{\epsilon_2}. \tag{8}$$

Then, with probability at least $1 - \epsilon$, it holds

$$c^\top x_k \leq c^\top \mathsf{cg}(\mathcal{X}_k). \tag{9}$$

*Remark 1 (Complexity of H&R):* Results like Lemma 4 are of great theoretical importance, — they show that H&R schemes can be implemented in polynomial time. However, from a practical viewpoint, they are still unsatisfactory. The main reason is that the constants involved are usually very large, thus making the method not good in practice. However, numerical experience shows that H&R usually performs much better than predicted theoretically. This fact is acknowledged by many authors, e.g., see [1]. This is also the main reason that convinced us to keep separate the analysis of the pure theoretical setup of Section IV. Moreover, we remark that the quantities $d, r, R$ involved in equation (8) are not constant but vary at every step of the algorithm.

A reasoning identical to the one leading to Lemma 5 can be employed for extending the analysis of Theorem 1. However, we prefer not to dwell further on this analysis, and to dedicate instead some space on different issues related to the practical implementation of the H&R procedure to the solution of standard SDP problems.

[1]For two probability distributions $\mathbb{P}_1$, $\mathbb{P}_2$ on the same underlying $\sigma$-algebra on $\mathcal{X}$, their *total variation* distance is defined as

$$\|\mathbb{P}_1 - \mathbb{P}_2\|_{\mathsf{tv}} \doteq \sup_{A \subseteq \mathcal{X}} |\mathbb{P}_1(A) - \mathbb{P}_2(A)|.$$

## VI. Application to Linear Matrix Inequalities

In this section, we focus our attention at standard semidefinite programs (SDP) of the form

$$\min c^\top x \;\; \text{subject to} \;\; F(x) \doteq F_0 + \sum_{i=1}^{n} x_i F_i \preceq 0, \tag{10}$$

where $c \in \mathbb{R}^n$ and $F_i \in \mathbb{R}^{m,m}$, $i = 0, \ldots, n$, are known symmetric matrices; the notation $F \preceq 0$ stands for negative semidefiniteness of the matrix $F$; the constraint in (10) is called a linear matrix inequality (LMI), and the convex set

$$\mathcal{X}_{\mathrm{LMI}} \doteq \{x \in \mathbb{R}^n \,:\, F(x) \preceq 0\}$$

is referred to as the feasible domain of this LMI. To exclude trivialities, we assume that the set $\mathcal{X}_{\mathrm{LMI}}$ is nonempty and bounded.

The optimization problem (10) is known to be one of the key problems in the theory of LMIs [2]. It has numerous applications in system theory and control, and at present there exist efficient solution techniques based on interior-point methods; e.g., see [10]. For this reason, we exemplify the use of our method for this widely adopted setup, with the belief that future versions of the RCP algorithm presented here will expose better performance than the existing techniques, especially for problems of very high dimensions.

The first key issue in the subsequent exposition is a boundary oracle for the set $\mathcal{X}_{\mathrm{LMI}}$, which we refer to as *Semidefinite Boundary Oracle* (SDBO). This oracle is given by the following lemma, developed in [12], see also the work [3] for a similar result.

*Lemma 6 (SDBO):* Let $A \prec 0$ and $B = B^\top$. Then, the minimal and the maximal values of the parameter $\lambda \in \mathbb{R}$ retaining the negative definiteness of the matrix $A + \lambda B$ are given by

$$\underline{\lambda} = \begin{cases} \max_{\lambda_i < 0} \lambda_i, \\ -\infty, \text{ if all } \lambda_i > 0; \end{cases} \qquad \overline{\lambda} = \begin{cases} \min_{\lambda_i > 0} \lambda_i, \\ +\infty, \text{ if all } \lambda_i < 0; \end{cases}$$

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

By means of this lemma, the desired endpoints of the segment $[\underline{z}, \overline{z}]$ are computed as $\underline{z} = z + \underline{\lambda}v$ and $\overline{z} = z + \overline{\lambda}v$. In the setup of this paper, assume that $z \in \mathcal{X}_{\mathrm{LMI}}$, and $v \in \mathbb{R}^n$ is a (random) direction. We then have $F(z + \lambda v) = F(z) + \lambda\big(F(v) - F_0\big) \doteq A + \lambda B$, and using Lemma 6, the

desired intersection points of the line and the boundary of $\mathcal{X}_{\rm LMI}$ are given by $\underline{z} = z + \underline{\lambda} v$ and $\overline{z} = z + \overline{\lambda} v$. To compute the intersection points with the boundary, the additional linear condition $c^\top (x - x_k) \leq 0$ defining the feasible set at step $k$ is to be taken into account, which is straightforward to implement. As seen from the aforesaid, this operation should be frequently performed in the process of iterations. The basis of this operation is finding the eigenvalues of matrices, which is for instance efficiently implemented in MATLAB. Hence, the boundary oracle for LMIs is accurate and "cheap" enough for matrices of quite large dimensions, as confirmed by the numerical simulations discussed in the next section.

### A. Implementation issues

Prior to reporting on the results of experiments, we discuss some of the most important implementation issues, including modifications of the basic scheme that showed to be crucial for the method to perform satisfactorily *in practice*. We stress that the implemented procedure does differ from Algorithm 4; it involves certain semi-heuristic tricks and would require a careful theoretical analysis that we aim to perform in the subsequent papers.

*a) Initialization:* To implement the H&R procedure, we need to find an initial feasible point $x_0 \in \mathcal{X}_{\rm LMI}$. This can be done by solving the auxiliary problem

$$\min \gamma \quad \text{subject to} \quad F(x) \preceq \gamma I$$

that admits the couple $\{x = 0, \gamma = \max \mathrm{eig}\big(F(0)\big)\}$ as initial feasible solution. If the optimal solution $\{x^*, \gamma^*\}$ of this problem is such that $\gamma^* > 0$, then the original problem (10) is unfeasible, otherwise we take $x_0 = x^*$ as initial feasible point. Notice also that, for the specific SDP problem, we can also easily force the initial feasible point to be at a distance $d$ from the boundary of $\mathcal{X}$.

The procedure above allows to find a good initial point to pass to Algorithm 4. However, it should be noted that, after the first step of this algorithm, by construction the H&R starting point $x_k$ lies on the boundary of the new set $\mathcal{X}_{k+1}$, and therefore we would have $d = 0$. There are different ways to avoid this degeneracy. For instance, the initialization procedure described in Section 4.1.1 of [4] can be directly applied to our setup. In our case, we adopted a method based on boundary points, as better described in item *d)* below.

*b) Number of points:* As it follows from Lemma 5, the number $N_k$ of random points suggested by Theorem 1 has to be enlarged because of the mixing time required for H&R algorithm; however, for simplicity, in the present implementation, we decided to lean on $N_k \equiv N$ points at every iteration. Also, extra points are required to bring the set $\mathcal{X}_{\rm LMI}$ in near-isotropic position, as described next.

*c) Isotropization:* Note that bound (8) depends explicitly on the quantities $r$ and $R$ which define the radii of two balls respectively inscribed and inscribing the set $\mathcal{X}$. If these quantities can be assumed to be known at the first steps of the RCP algorithm, this does not hold true anymore as the algorithm proceeds. Indeed, at step $k$, the set $\mathcal{X}_k$ is obtained from the original set $\mathcal{X}$ by cutting a portion defined by the hyperplane $\mathcal{H}_k$, and there is no guarantee that it still contains a ball of radius $r$. A well-accepted technique (e.g., see [1]) to overcome this problem, is to perform an affine transformation of $\mathcal{X}_k$ to bring it in near-isotropic position[2].

Isotropization techniques stem from the consideration that, for a set in near-isotropic position, the diameter of $\mathcal{X}$ is less or equal than $n$ and the ratio $R/r$ is $\mathcal{O}(\sqrt{n})$. Hence, in this case the bound (8) explicitly rewrites as

$$T \geq C n^4 \ln \frac{n}{d\epsilon_1}.$$

The procedure for bringing a set in near isotropic position is quite straightforward: For a general convex set $\mathcal{X}$, let $y^{(1)}, \ldots, y^{(M)}$ be random samples from $\mathcal{X}$. Compute the quantities

$$\bar{y} \doteq \frac{1}{M} \sum_{i=1}^{M} y^{(i)} \text{ and } Y \doteq \frac{1}{M} \sum_{i=1}^{M} (y^{(i)} - \bar{y})(y^{(i)} - \bar{y})^\top \quad (11)$$

and apply the affine transformation defined by $\bar{y}$ and $Y$ to the set $\mathcal{X}$, obtaining the new set

$$\widetilde{\mathcal{X}} \doteq \left\{ x \in \mathbb{R}^n \,:\, Y^{\frac{1}{2}} x + \bar{y} \in \mathcal{X} \right\} \quad (12)$$

It was shown by Rudelson [16] that one can bring a set in near-isotropic position with high probability in $\mathcal{O}(n \log^2 n)$.

*d) Heuristics:* Various heuristic schemes proved effective in our simulations. As regards the isotropization step, we noticed that, as a by-product of H&R algorithm at the $k$th iteration we have $2N$ points on the boundary of the set $\mathcal{X}_k$ (endpoints $\underline{z}$, $\overline{z}$). The transformation matrix $Y$ in (11) is composed from these boundary points. Based on the assumption that the subsequent sets $\mathcal{X}_k$ and $\mathcal{X}_{k+1}$ have "similar" geometry, we perform isotropization of $\mathcal{X}_{k+1}$ by means of this matrix $Y$ obtained at the previous step.

Moreover, those of the boundary points obtained at the $k$th step that fall into $\mathcal{X}_{k+1}$ where used to perform the initialization phase of item *a)*. More precisely, the initial point for the H&R algorithm at the $(k+1)$st step was taken as the arithmetic mean of these selected points. If the cut is performed through the "best" H&R point as described above, then there exist $N_b \geq 1$ such points (at least the endpoint of the chord associated with the "best" H&R point). For the sake of numerical safety, we performed the cut through the "second best" point to make sure $N_b \geq 3$ so that averaging yields an interior initial point.

### B. Preliminary Numerical Experiments

The method was tested over a range of problems whose data were generated randomly as described next. Symmetric matrices $F_i$ were generated so as to guarantee nonemptiness of $\mathcal{X}_{\rm LMI}$; for simplicity, we chose $F_0 < 0$ in the form

$$M = 2\,\mathtt{rand}(m) - 1; \quad F_0 = -M \cdot M^\top - \mathtt{eye}(m),$$

[2]A convex set $\mathcal{X} \in \mathbb{R}^n$ is said to be in isotropic position if, given a uniform random point $x \in \mathcal{X}$, it holds $\mathbb{E}[x] = 0$ and $\mathbb{E}[xx^\top] = I$, i.e. its covariance matrix is identity. We say that a convex set $\mathcal{X}$ is in near-isotropic position if $\mathtt{cg}(\mathcal{X}) = 0$ and the covariance matrix of the uniform distribution over $\mathcal{X}$ has eigenvalues between $\frac{1}{2}$ and $\frac{3}{2}$.

so that $x_0 = 0$ was adopted as the initial point for iterations. The rest of the matrices $F_i$, $i > 0$, were computed as (a) $M = 2\,\mathtt{rand}(m/2) - 1$; (b) $M = M + M^\top$; (c) $F_i = \mathtt{blkdiag}(M; -M)$ in order to guarantee $\mathcal{X}_{\rm LMI}$ to be bounded and to ensure the existence of a finite solution. Without loss of generality, the vector $c$ in the cost function was taken as $c = (0\ 0\ \ldots\ 0\ 1)$. Such data were generated for dimensions of the $F_i$ matrices as large as $m = 100$, and the dimensions of the design vector $x$ as high as $n = 1,000$. The method demonstrated pretty stable performance; as far as the widely used computer MATLAB-based realizations of interior-point methods (the `solvesdp` routine in SeDuMi Toolbox) are concerned, it has showed comparable performance, sometimes exceeding the classical methods in accuracy. To illustrate, we briefly describe some preliminary results of simulations.

For moderately sized problems with $n = 10$, $m = 10$, we obtain 7 to 9 exact digits after 45 to 55 iterations with $N = 200$ H&R points at each step. In other words, the observed rate of convergence $(f_{k+1} - f^*)/(f_k - f^*)$ was approximately 0.65 to 0.7 as compared to the theoretical rate 0.9 for this dimension $n = 10$ (see (3)). The same accuracy is typically observed after 30 to 40 steps if $N = 500$ H&R points were used.

In the second set of experiments the method was tested on SDP problems having large dimensions of the design vector, $n = 1,000$ and $m = 10$. Typically, the method reproduces 7 to 8 exact decimal digits for the function value after 20–30 iterations ($N = 1,500$ points were used). As a rule, for these high-dimensional problems, the `solvesdp` routine exhibits slightly lower accuracy (6 to 7 digits); moreover, sometimes it yields a formally infeasible point $x^*$, e.g., $\max \mathrm{eig}\big(F(x^*)\big) \approx 2 \cdot 10^{-7} > 0$ was observed.

The third set of experiments was conducted with problems having so-called worst-case geometry, where the feasible domain is "pyramid-like." For such sets, the bound $n/(n+1)$ on the rate of convergence of the DCG algorithm given by (3) is known to be attained. In the experiments, the matrices $F_i$ were chosen in such a way as to yield $\mathcal{X}_{\rm LMI} = \{x \in \mathbb{R}^n : \ \|x\|_1 \le 1\}$, so the minimum function value is $f^* = -1$ and $m = 2^n$ by construction. Notably, for such a geometry, the isotropization procedure described above very often *does not* bring $\mathcal{X}_{\rm LMI}$ to near-isotropic position, and is therefore omitted. As a result, in practice we lean on a much smaller number $N$ of H&R points that would be needed to correctly restore the $Y$ matrix. Specifically, for the worst-case geometry problem with $n = 10$ variables and quite large dimension $m = 1024$ of the $F_i$ matrices, using only $N = 40$ points yields 6 to 8 exact digits after 75–85 iterations depending on realization. This means that, in accordance with the theory developed here, the observed rate of convergence (approximately 0.8) is usually better than 0.9, which is given by (3).

## VII. Conclusions

In this paper, we study a novel randomized scheme for convex optimization. The theoretical behavior of the method is analyzed in the abstract framework in which a Uniform Generating Oracle is assumed to be available. A novel probabilistic analysis is carried out, and our method is shown to outperform the classical DCG methods. In particular, we explicitly derive bounds on the number of random samples $N$ necessary to guaranteed with prescribed high probability the desired convergence behavior of the algorithm. It turns out that this number is very small, thus guaranteeing practical implementability of the proposed scheme. These bounds are then specialized to the case in which UGO is implemented using H&R. The second part of the paper is devoted to numerical simulations for the special case of SDP optimization. To the best of our knowledge, this is the first time that a practical implementation of the method is discussed in such detail.

## References


[1] D. Bertsimas and S. Vempala. Solving convex programs by random walks. *Journal of the ACM*, 51(4):540–556, 2004.

[2] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. SIAM, Philadelphia, 1994.

[3] G. Calafiore. Random walks for probabilistic robustness. In *Conf. Decison and Control*, pages 5316–5321, 2004.

[4] G. Calafiore and F. Dabbene. A probabilistic analytic center cutting plane method for feasibility of uncertain LMIs. *Automatica*, 43(12):2022–2033, 2006.

[5] F. Dabbene. A randomized cutting plane scheme for convex optimization. In *Multiconf. on Systems and Control*. San Antonio, 2008.

[6] B. Grünbaum. Partitions of mass-distributions and convex bodies by hyperplanes. *Pacific J. Math.*, 10:1257–1261, 1960.

[7] A.Y. Levin. On an algorithm for the minimization of convex functions over convex functions. *Soviet Math. Doklady*, 6:286–290, 1965.

[8] L. Lovász. Hit-and-Run mixes fast. *Mathematical Programming*, 86:443–461, 1999.

[9] L. Lovász and S. Vempala. Hit-and-Run from a corner. *SIAM Journal on Computing*, 35(4):985–1005, 2006.

[10] Y. Nesterov and A.S. Nemirovski. *Interior Point Polynomial Algorithms in Convex Programming*. SIAM Journal on Applied Mathematics, Philadelphia, 1994.

[11] D.J. Newman. Location of the maximum on unimodal surfaces. *Journal of the ACM*, 12(3):395–398, 1965.

[12] B.T. Polyak and P.S. Shcherbakov. The $D$-decomposition technique for linear matrix inequalities. *Automat. Remote Control*, 67(11):1847–1861, 2006.

[13] B.T. Polyak and P.S. Shcherbakov. A randomized method for solving semidefinite programs. In *Proc. of the 9th IFAC Workshop ALCOSP'07*, 2007. Available from *IPACS Electronic Library* at `http://lib.physcon.ru/`.

[14] L. Rademacher. Approximating the centroid is hard. In *Proc. of the 24th Annual Symposium on Computationa Geometry (SoCG-07)*, 2007.

[15] J. Radon. Über eine Erweiterung des Begriffs der konvexen Functionen, mit einer Anwendung auf die Theorei der konvexen Körper. *S. B. Akad. Wiss. Wien*, 125:241–258, 1916.

[16] M. Rudelson. Random vectors in the isotropic position. *J. Funct. Anal.*, 164:60–72, 1999.

[17] R.L. Smith. Efficient Monte-Carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32:1296–1308, 1984.

[18] R. Tempo, G. Calafiore, and F. Dabbene. *Randomized Algorithms for Analysis and Control of Uncertain Systems*. Communications and Control Engineering Series. Springer-Verlag, London, 2004.

[19] V.F. Turchin. On the computation of multidimensional integrals by the Monte-Carlo method. *Theory of Probability and Appl.*, 16(4):720–724, 1971.

[20] S. Vempala. Geometric random walks: a survey. *Combinatorial and Computational Geometry*, 52:573–611, 2005.