SIAM J. Optim.
Vol. 20, No. 6, pp. 3185–3207

A RANDOMIZED CUTTING PLANE METHOD WITH PROBABILISTIC GEOMETRIC CONVERGENCE[*]

F. DABBENE[†], P. S. SHCHERBAKOV[‡], and 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. We analyze the convergence properties of the algorithm from a theoretical viewpoint, i.e., under the rather standard assumption that an algorithm for uniform generation of random points in the convex body is available. Under this assumption, the expected rate of convergence for such method is proved to be geometric. This analysis is based on new results on the statistical properties of the empirical minimum over a convex body that we obtained in this paper. Moreover, explicit sample size results on convergence are derived. In particular, we compute the minimum number of random points that should be generated at each step in order to guarantee that, in a probabilistic sense, the method performs better than the deterministic center-of-gravity algorithm. From a practical viewpoint, we show how the method can be implemented using hit-and-run versions of Markov-chain Monte Carlo algorithms and exemplify the performance of this implementable modification via a number of illustrative problems. A crucial notion for the hit-and-run implementation is that of boundary oracle, which is available for most optimization problems including linear matrix inequalities and many other kinds of constraints. Preliminary numerical results for semidefinite programs are presented confirming that the randomized approach might be competitive to modern deterministic convex optimization methods.

Key words. convex optimization, randomized algorithms, hit-and-run, linear matrix inequalities

AMS subject classifications. 90C25, 65K05, 65K10, 90C15

DOI. 10.1137/080742506

1. Introduction.

Recent years exhibited a growing interest toward randomized algorithms in computer science, control, and optimization; e.g., see [26, 40, 6]. There are numerous reasons for this interest, from philosophical to computational ones. The present paper continues this line of research for convex optimization. In particular, we consider a convex minimization problem of the form

$$\min c^\top x \quad \text{subject to} \quad x \in \mathcal{X}, \tag{1.1}$$

where the set $\mathcal{X} \subset \mathbb{R}^n$ is a *convex body*; i.e., it is compact with nonempty interior. Specifically, we assume that there exist two Euclidean balls $\mathcal{B}_r$ and $\mathcal{B}_R$, of radii $0 < r < R$, such that $\mathcal{B}_r \subseteq \mathcal{X} \subseteq \mathcal{B}_R$. The linear cost function is taken without loss of generality, since any convex optimization problem can be reduced to this form. Also, for the sake of simplicity, we assume that problem (1.1) admits a unique solution which we denote by x^*, and we let $f^* \doteq c^\top x^*$.

The use of a cutting plane scheme for the solution of convex optimization problems as the one above dates back to 1960 [19]. A center-of-gravity version of a cutting plane,

[*]Received by the editors December 3, 2008; accepted for publication (in revised form) August 27, 2010; published electronically October 21, 2010. A preliminary version of this work was published in the conference paper *A randomized cutting plane scheme for convex optimization*, Proceedings of the IEEE Multiconference on Systems and Control, San Antonio, TX, 2008.
http://www.siam.org/journals/siopt/20-6/74250.html

[†]IEIIT-CNR, Politecnico di Torino, Torino 10129, Italy (fabrizio.dabbene@polito.it).

[‡]Institute of Control Science, Russian Academy of Sciences, Moscow, Russia (sherba@ipu.ru, boris@ipu.ru).

for a slightly different problem formulation, was proposed almost simultaneously by Levin [21] and Newman [28] in 1965. This method can be viewed as a natural extension of the bisection algorithm for one-dimensional (1D) unimodal functions to general convex multidimensional functions and has been proved to have extremely appealing features: it works also when the set $\mathcal{X}$ is described by nonsmooth functions, and its theoretical convergence is among the best known. However, despite its very attractive theoretical behavior, the center-of-gravity algorithm in its pure form has not been used in practice for a very simple reason: *For a generic convex set, computing the center of gravity turns out to be more difficult than solving the original optimization problem.*

As a result, the use of different types of centers, instead of the center of gravity, has been proposed in the literature. Most of the methods devised for circumventing this problem can be classified as *exterior-point* methods, in that they use the center of a specific *localization set*, which is updated at each step and is guaranteed to always contain the feasible set. The first of these techniques is the famous ellipsoid method developed by Shor [37] and Yudin and Nemirovski [43] in the 1970s and used in 1979 by Khachiyan to show polynomial solvability of linear programs [20]. The localization set in this case is represented by an ellipsoid inscribing $\mathcal{X}$. Evolutions of this approach led to methods based on the analytic center of a polytope inscribing $\mathcal{X}$; see [14] and the references therein for an excellent survey.

In recent years, the interest in designing implementable methods based on the center of gravity has gained new interest motivated by the randomized scheme proposed in [3]. In this latter work, the authors compute at each step an *approximate* center of gravity of an outer polytope inscribing $\mathcal{X}$, based on points randomly extracted in the polytope. In the present work, we move a step further and consider the possibility of randomly extracting points *directly* in the feasible set $\mathcal{X}$. This idea has been inspired by the recent results in [31].

The paper is structured as follows. In section 2, we review the classical deterministic center-of-gravity (DCG) method of [21, 28] and analyze its properties. The rate of convergence of the DCG method has been estimated via Grünbaum's theorem (see below) and happened to be geometric. We provide another result based on Radon's theorem [35] which also guarantees geometric convergence. In section 3 we present a theoretical analysis of the statistical properties of the empirical minimum among a set of random points in a convex body. These results pave the road to the introduction in section 4 of a randomized cutting plane (RCP) scheme based on the generation of random points in the feasible set. Namely, at each iteration, N points are generated, the best of them z_k; i.e., the one corresponding to the minimum cost function value $c^\top x$ (empirical minimum) is chosen, and a cut of the half-space $\{x : c^\top x \le c^\top z_k\}$ is performed.

In the sections to follow, we analyze the probabilistic properties of the algorithm. In particular, in section 5.1, we prove convergence in first and second moments and explicitly compute the *expected rate* of convergence of the RCP algorithm; notably, for $N = 1$ this rate coincides with that of the DCG algorithm, while it is shown to improve by a factor of $\ln(N + 1)$ for $N > 1$. Then, in section 5.2, a novel and different type of probabilistic analysis of the RCP algorithm is provided. In particular, we estimate the *probability* that RCP performs worse than DCG and calculate the number N of points to be generated in RCP to guarantee a better performance with prescribed (high) probability.

Section 6 provides discussions and reasonings in support of the implementable modification of RCP scheme, which is based on hit-and-run versions of the Monte Carlo method, a procedure aimed at *approximately uniform* generation of points in a body via random walks.

The application of hit-and-run to convex optimization has been first proposed in [3]. Note that the method in [3] differs from the implementable version of RCP presented in section 6 in the following crucial aspects:

(i) The problem formulation in [3] differs from (1.1), and it is this difference that allows us to introduce the concept of *best point*;

(ii) The method proposed in [3] updates at each step an outer polytope inscribing $\mathcal{X}$, while here we update directly the set $\mathcal{X}$;

(iii) In [3], the authors use a so-called separation oracle, while we exploit here a boundary oracle (BO) for the set $\mathcal{X}$.

Notice that, in view of (ii), our method can be considered an interior-point algorithm, while the method in [3] in fact represents a classical exterior-point localization scheme.

The results of illustrative numerical experiments with the proposed algorithm for semidefinite programming problems are described in section 7.

The material in the two conference papers [9, 8] serves as a basis for the exposition to follow.

2. DCG algorithm.

For a convex body $\mathcal{X}$, define its *center of gravity* as

$$\mathsf{cg}(\mathcal{X}) \doteq \frac{\displaystyle\int_{\mathcal{X}} x\mathrm{d}x}{\displaystyle\int_{\mathcal{X}} \mathrm{d}x}.$$

Then the deterministic cutting plane method based on recursive cutting of the feasible set through the center of gravity can be stated as follows in Algorithm 1; see also [9, 31].

ALGORITHM 1 (DCG ALGORITHM).

```
Input: $\mathcal{X}$
Output: $x_k$
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; $k \Leftarrow k+1$; goto 2.
```

The behavior of the DCG cutting plane is illustrated in Figure 2.1. 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 considering the new set

$$\mathcal{X}_1 \doteq \left\{x \in \mathcal{X}_0 : c^\top(x - x_0) \leq 0\right\} \subset \mathcal{X}_0,$$

obtained by cutting off a portion of $\mathcal{X}_0$ using the hyperplane

$$\mathcal{H}_0 \doteq \left\{x \in \mathbb{R}^n : c^\top(x - x_0) = 0\right\}.$$

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 $\mathcal{H}_1$ passing 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

$$c^\top x_{k+1} < c^\top x_k.$$

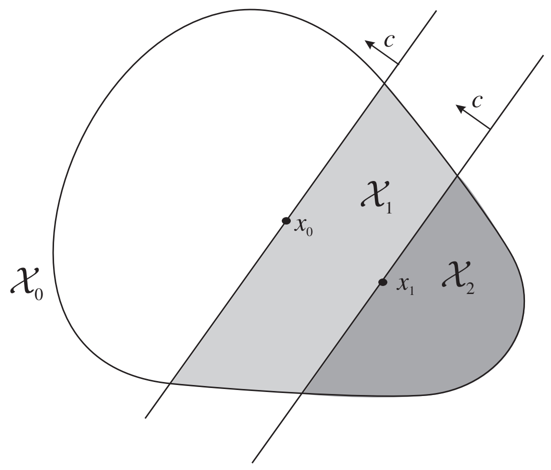

FIG. 2.1. *Illustration of the* DCG *algorithm.*

The convergence analysis of this algorithm can be based on the following proposition due to Grünbaum [16].

PROPOSITION 2.1 (Grünbaum). *Let $\mathcal{X} \subset \mathbb{R}^n$ be a convex body, and let $x_{\mathrm{G}} = \mathsf{cg}(\mathcal{X})$ be its center of gravity. Consider any hyperplane $\mathcal{H} = \{x \in \mathbb{R}^n : c^\top(x - x_{\mathrm{G}}) = 0\}$ passing through x_{G}. This hyperplane divides the set $\mathcal{X}$ in the two subsets*

$$\mathcal{X}_1 = \{x \in \mathcal{X} : c^\top x > c^\top x_{\mathrm{G}}\},$$
$$\mathcal{X}_2 = \{x \in \mathcal{X} : c^\top x \le c^\top x_{\mathrm{G}}\}.$$

Then the following relations hold for $i = 1, 2$:

$$\mathrm{vol}\,(\mathcal{X}_i) \;\le\; \left(1 - \left(\frac{n}{n+1}\right)^{1/n}\right) \mathrm{vol}\,(\mathcal{X}) \;\le\; \left(1 - \frac{1}{\mathrm{e}}\right) \mathrm{vol}\,(\mathcal{X}), \tag{2.1}$$

where $\mathrm{vol}\,(\mathcal{X})$ *denotes the n-dimensional Lebesgue measure (volume) of $\mathcal{X} \subset \mathbb{R}^n$.*

From this proposition, it follows that each step of the DCG algorithm guarantees that a given portion of the feasible set is cut out, namely, $\mathrm{vol}\,(\mathcal{X}_{k+1}) \le (1 - 1/\mathrm{e})\mathrm{vol}\,(\mathcal{X}_k)$. A recursive application of this inequality immediately leads to the volume inequality

$$\mathrm{vol}\,(\mathcal{X}_k) \le (1 - 1/\mathrm{e})^k \mathrm{vol}\,(\mathcal{X}_0) \approx (0.63)^k \mathrm{vol}\,(\mathcal{X}_0), \tag{2.2}$$

which proves that DCG has guaranteed geometric convergence in terms of *volumes*. Notice that the volume reduction in (2.2) is completely independent of all problem parameters, including the dimension n.

A different type of convergence analysis of the DCG algorithm, which provides an estimated rate of monotone decrease of the *cost-function values* sequence

$$f_k \doteq c^\top x_k,$$

can be deduced from the following result by Radon [35] on the measures of symmetry of convex bodies.

PROPOSITION 2.2 (Radon). *Let $\mathcal{X} \subset \mathbb{R}^n$ be a convex body and $x_{\mathrm{G}} = \mathsf{cg}(\mathcal{X})$ be its center of gravity. Denote by $\mathcal{H}$ an arbitrary $(n-1)$-dimensional hyperplane through*

x_G, and let $\mathcal{H}_1$ and $\mathcal{H}_2$ be the two hyperplanes supporting $\mathcal{X}$ and parallel to $\mathcal{H}$. Denote by

$$r(\mathcal{H}) \doteq \frac{\min\{\mathrm{dist}(\mathcal{H},\mathcal{H}_1);\mathrm{dist}(\mathcal{H},\mathcal{H}_2)\}}{\max\{\mathrm{dist}(\mathcal{H},\mathcal{H}_1);\mathrm{dist}(\mathcal{H},\mathcal{H}_2)\}}$$

the ratio of the distances from $\mathcal{H}$ to $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively. Then

$$\min_{\mathcal{H}} r(\mathcal{H}) \geq 1/n.$$

This result immediately applies to the DCG algorithm. Indeed, with $\mathcal{H}_1$ and $\mathcal{H}$ being the two hyperplanes through the two successive points x_k and x_{k+1} as described above and $\mathcal{H}_2$ being the supporting hyperplane through the optimal point $x^* \doteq \arg\min_{\mathcal{X}} c^\top x$, we arrive at the following estimate:

$$f_k - f^* \leq \frac{n}{n+1}(f_{k-1} - f^*), \tag{2.3}$$

where f^* is the optimal value of the objective function. In particular, the following lemma is an immediate consequence of Proposition 2.2. The proof of the lemma immediately follows from (2.3) by imposing $f_k - f^* \leq \alpha$.

LEMMA 2.3 (convergence of DCG). *Define $D \doteq |f_0 - f^*|$. Then the DCG algorithm computes an α-optimal solution (i.e., such that $f_k - f^* \leq \alpha$) in a number of steps bounded as*

$$k = \left\lceil \frac{\ln \frac{D}{\alpha}}{\ln \frac{n+1}{n}} \right\rceil = \mathcal{O}\left(n \ln \frac{D}{\alpha}\right).$$

In other words, the method is expected to have a guaranteed geometric rate of convergence. Notably, to the best of our knowledge, this result has never been used in optimization, in contrast to similar results on the guaranteed volumetric reduction (in the spirit of Proposition 2.1), which are typical to various modifications of the ellipsoid method.

However, as already mentioned in the introduction, this nice convergence property has a crucial drawback: The practical implementation of the DCG algorithm in its pure original form is hindered by the fact that computing the center of gravity turns out to be an NP-hard problem, even for the simple case when $\mathcal{X}$ is a polytope, as recently proved in [34]. This motivates the introduction of a randomized version of the method in section 4. Prior to describing in detail this scheme, in the next section we develop some important results on the statistical properties of the empirical minimum among a set of random points in a generic convex body. These results, which to the best of our knowledge are novel and original, are not limited to the considered setting and may have a very general application, thus constituting one of the main theoretical contributions of this work.

3. Statistical properties of the empirical minimum over a convex body.

In this section, we consider the situation where N uniform random samples are drawn over a generic convex body $\mathcal{X}$. That is, we assume that a multisample

$$x^{(1\dots N)} \doteq \{x^{(1)}, \dots, x^{(N)}\}$$

is given, where $x^{(i)}$ are independently and identically distributed (i.i.d.) uniform samples drawn over $\mathcal{X}$. More formally, we make the assumption on the availability of a mechanism for generating uniform samples in a convex body.

Assumption 1 (uniform generating oracle). We assume that a *uniform generating oracle* is available such that for any convex set $\mathcal{X} \subset \mathbb{R}^n$, a call to the oracle returns N random i.i.d. points uniformly distributed in $\mathcal{X}$.

Assumption 1 is made only for theoretical analysis purposes and is quite common in the computational geometry community; see, e.g., [33]. Also, we point out that this assumption forms equally sound grounds for the theoretical analysis below, as the assumption on the availability of the center of gravity does in Radon's result. In that respect, this ideal setting allows us to rigorously prove that randomized schemes may outperform classical deterministic ones.

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: $f^{(i)} \doteq c^\top x^{(i)}, \quad i = 1, \ldots, N$. Then we can define the so-called *empirical minimum* (e.g., see [40]) over these random points as

$$f_{[1]} \doteq \min_{i=1,\ldots,N} f^{(i)}. \tag{3.1}$$

Notice that $f_{[1]}$ is also a random variable; it represents the so-called *first order statistics* of $f^{(i)}$, e.g., see [10]. 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 $f^* = \min_{\mathcal{X}} c^\top x$ is bounded from below and from above by constants that depend only on n and N.

THEOREM 3.1. *Let $\mathcal{X} \subset \mathbb{R}^n$ be a convex body. Given $c \in \mathbb{R}^n$, define $h \doteq (\max_{\mathcal{X}} c^\top x - \min_{\mathcal{X}} c^\top x)$ and $f^* \doteq \min_{\mathcal{X}} c^\top x$. Then it holds that*

$$\frac{h}{nN+1} \;\le\; \mathbb{E}\left[f_{[1]} - f^*\right] \le \frac{h}{n} B\left(N+1, \frac{1}{n}\right) \tag{3.2}$$

$$\le h \left(\frac{1}{N+1}\right)^{\frac{1}{n}}, \tag{3.3}$$

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

Proof. Assume, without loss of generality, that $c = [1\,0\,\cdots\,0]^\top$ (that is, $c^\top x = x_1$) and that $x^* = \arg\min_{\mathcal{X}} c^\top x = 0$. We begin by proving the upper bound in (3.2). The first step in the proof of Theorem 3.1 is closely related to the proof of Lemma 4 in [3]. Define the function

$$\phi_{\mathcal{X}}(s) = \frac{1}{\mathrm{vol}\,(\mathcal{X})} \int_{x \in \mathcal{X}, c^\top x = s} \mathrm{d}x$$

that represents the $(n-1)$-dimensional volume of $\mathcal{X}$ intersected with the hyperplane $c^\top x = s$ as a fraction of the n-dimensional volume of $\mathcal{X}$. Then define the set $\mathcal{X}_0$ obtained by replacing each cross-section $\mathcal{X} \cap \{x : x_1 = s\}$ by an $(n-1)$-dimensional ball of volume $\phi_{\mathcal{X}}(s)$ and centered at the point $[s\,0\,\cdots\,0]^\top$, as shown in Figure 3.1. The set $\mathcal{X}_0$ has the same volume as $\mathcal{X}$, and it holds that $\phi_{\mathcal{X}}(s) = \phi_{\mathcal{X}_0}(s)$. Moreover, if we let S_1 and S_2 be the cross-sections of $\mathcal{X}_0$ at s_1 and s_2, respectively, and let S be its cross-section at a point $s = \lambda s_1 + (1-\lambda)s_2$, $\lambda \in [0,1]$, then, by the Brunn–Minkowski inequality (e.g., see [13]), we have

$$\mathrm{vol}\,(S)^{\frac{1}{n-1}} \;\ge\; \mathrm{vol}\,(\lambda S_1 + (1-\lambda)S_2)^{\frac{1}{n-1}} \;\ge\; \lambda \mathrm{vol}\,(S_1)^{\frac{1}{n-1}} + (1-\lambda)\mathrm{vol}\,(S_2)^{\frac{1}{n-1}}. \tag{3.4}$$

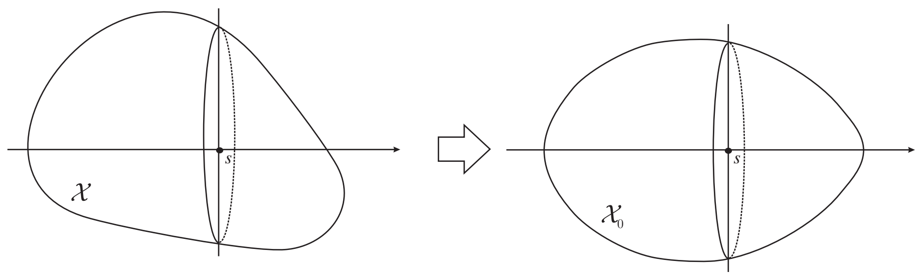

FIG. 3.1. *First step of the proof of Theorem* 3.1.

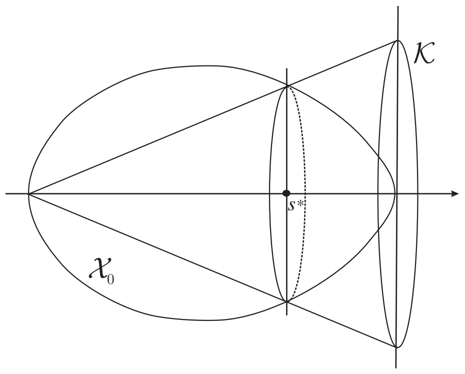

FIG. 3.2. *Second step of the proof of Theorem* 3.1.

Hence, if we denote by $r(s)$ the radius of the $(n-1)$-dimensional ball at s, then (3.4) 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 $\mathcal{X}_0$ is a convex set.

As a second step, define now the cone $\mathcal{K}$ with base area $S = \frac{n}{h}\text{vol}(\mathcal{X})$, the axis directed along c and located as shown in Figure 3.2, with h being the height of $\mathcal{K}$. Then, by construction, we have $\text{vol}(\mathcal{K}) = \text{vol}(\mathcal{X}) = \text{vol}(\mathcal{X}_0)$. Let s^* be the coordinate at which the sets $\mathcal{X}_0$ and $\mathcal{K}$ intersect; see Figure 3.2. Next, for every $s \in [0,h]$, define the sets $\mathcal{X}^+(s) \doteq \{x \in \mathcal{X} : x_1 \geq s\}$, $\mathcal{X}_0^+(s) \doteq \{x \in \mathcal{X}_0 : x_1 \geq s\}$, and $\mathcal{K}^+(s) \doteq \{x \in \mathcal{K} : x_1 \geq s\}$ as shown in Figure 3.3. Then the following chain of inequalities holds:

$$\begin{aligned} \mathbb{P}\{f^{(i)} \geq s\} &= \frac{\text{vol}(\mathcal{X}^+(s))}{\text{vol}(\mathcal{X})} = \frac{\text{vol}\left(\mathcal{X}_0^+(s)\right)}{\text{vol}(\mathcal{X}_0)} \\ &\leq \frac{\text{vol}(\mathcal{K}^+(s))}{\text{vol}(\mathcal{K})} = \frac{h^n - s^n}{h^n}, \end{aligned} \tag{3.5}$$

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

$$\frac{\text{vol}\left(\mathcal{X}_0^+(s)\right)}{\text{vol}(\mathcal{X}_0)} \leq \frac{\text{vol}(\mathcal{K}^+(s))}{\text{vol}(\mathcal{K})},$$

and, for $s < s^*$,

$$\frac{\text{vol}\left(\mathcal{X}_0^+(s)\right)}{\text{vol}(\mathcal{X}_0)} = 1 - \frac{\text{vol}\left(\mathcal{X}_0^-(s)\right)}{\text{vol}(\mathcal{X}_0)} \leq 1 - \frac{\text{vol}(\mathcal{K}^-(s))}{\text{vol}(\mathcal{X}_0)} = \frac{\text{vol}(\mathcal{K}^+(s))}{\text{vol}(\mathcal{K})},$$

where $\mathcal{X}_0^-(s) \doteq \{x \in \mathcal{X}_0 : x_1 < s\}$ and $\mathcal{K}^-(s) \doteq \{x \in \mathcal{K} : x_1 < s\}$.

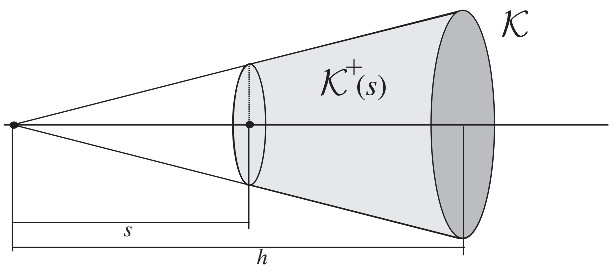

FIG. 3.3. *Third step of the proof of Theorem* 3.1: *the sets* $\mathcal{K}$ *and* $\mathcal{K}^+(s)$.

As a final step for proving the upper bound in (3.2), notice that $f_{[1]}$ is a positive random variable, and hence we may write

$$\begin{aligned}\mathbb{E}\left[f_{[1]}\right] &= \int_0^h \mathbb{P}\{f_{[1]} \geq s\}\mathrm{d}s = \int_0^h \left(\mathbb{P}\{f^{(i)} \geq s\}\right)^N \mathrm{d}s \\ [\text{from (3.5)}] &\leq \int_0^h \left(\frac{h^n - s^n}{h^n}\right)^N \mathrm{d}s \\ [t = s^n/h^n] &= \frac{h}{n}\int_0^1 (1-t)^N t^{\frac{1}{n}-1}\mathrm{d}t = \frac{h}{n}B\left(N+1, \frac{1}{n}\right).\end{aligned}$$

Finally, applying Theorem 3.4. in [1], we get the following bounds

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

thus proving the inequality in (3.3), with the cone $\mathcal{K}$ being the attainable "worst-case" configuration for $\mathcal{X}$.

The lower bound in (3.2) can be proved similarly. Namely, instead of the $\mathcal{K}$ 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:

$$\mathbb{P}\{f^{(i)} \geq s\} \geq \frac{s^n}{h^n}. \tag{3.6}$$

Therefore, we obtain

$$\begin{aligned}\mathbb{E}\left[f_{[1]}\right] &= \int_0^h \mathbb{P}\{f_{[1]} \geq s\}\mathrm{d}s = \int_0^h \left(\mathbb{P}\{f^{(i)} \geq s\}\right)^N \mathrm{d}s \\ [\text{from (3.6)}] &\geq \int_0^h \left(\frac{s^n}{h^n}\right)^N \mathrm{d}s = \frac{h}{nN+1},\end{aligned} \tag{3.7}$$

which concludes our proof. $\square$

From Figure 3.4, it is seen that the upper and lower bounds derived in Theorem 3.1 are indeed tight: the upper bound is attained when the set $\mathcal{X}$ has the ("worst-case") cone configuration ($\triangleleft$), while the lower bound corresponds to the ("best-case") inverted cone configuration ($\triangleright$). In the figure, we also plot the empirical mean values of $f_{[1]}$ (averaged over 10,000 realizations) computed for these two cones as well as for two intermediate configurations, namely, a box and a ball (ℓ_∞- and ℓ_2-norm balls).

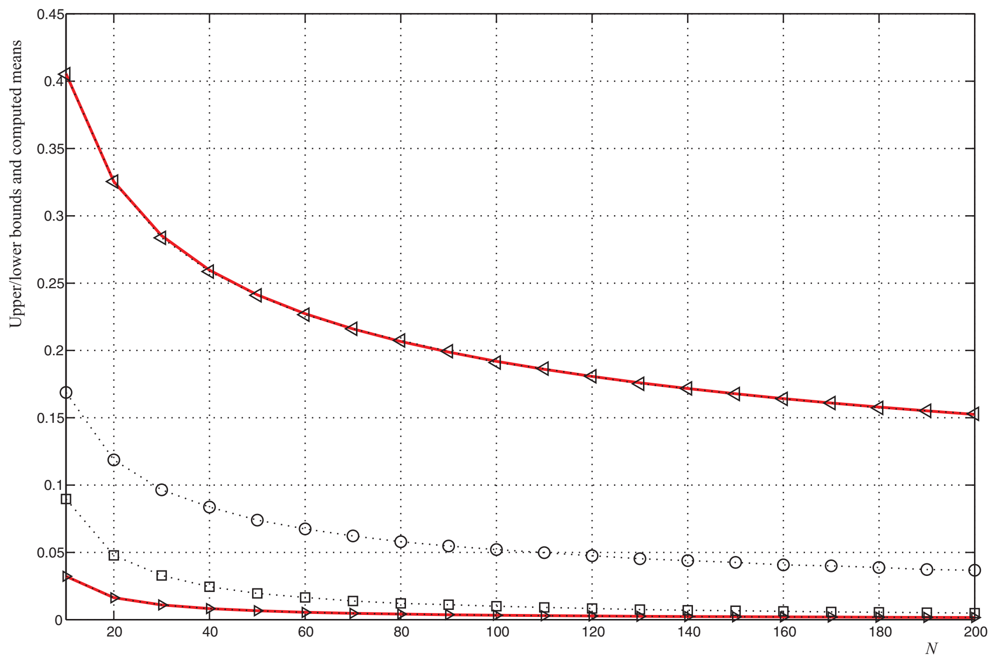

FIG. 3.4. *Illustration of the derived bounds on the mean for $n = 3$: cone configuration, upper bound* ($\triangleleft$)*; inverted cone configuration, lower bound* ($\triangleright$)*; cube configuration* ($\square$)*; ball configuration* ($\circ$)*.*

Remark 1 (connections to Radon's result). Theorem 3.1 constitutes an important and nontrivial extension of Proposition 2.2. Indeed, for $N = 1$, we have $\mathbb{E}\left[f_{[1]}\right] = \mathbb{E}\left[f^{(i)}\right] = c^\top \mathsf{cg}(\mathcal{X})$, and

$$\frac{1}{n}B\left(2, \frac{1}{n}\right) = \frac{n}{n+1},$$

which coincides with the expression in (2.3).

The following corollary extends the results of Theorem 3.1 to the analysis of the variance of the empirical minimum.

COROLLARY 3.2. *Let $\mathcal{X} \subset \mathbb{R}^n$ be a convex body. Define h, f^*, and $f_{[1]}$ as in Theorem* 3.1. *Then it holds that*

$$\mathbb{E}\left[\left(f_{[1]} - f^*\right)^2\right] \leq \frac{2h^2}{n}B\left(N+1, \frac{2}{n}\right) \;\leq\; h^2\left(\frac{1}{N+1}\right)^{\frac{2}{n}}. \tag{3.8}$$

Proof. Assume again, without loss of generality, that $f^* = 0$ and $c = [1\,0\,\cdots\,0]^\top$. Then, $f_{[1]}$ is a positive random variable; hence, we may write

$$\mathbb{E}\left[f_{[1]}^2\right] = \int_0^h \mathbb{P}\{f_{[1]}^2 \geq s\}\mathrm{d}s \;=\; \int_0^h \mathbb{P}\{f_{[1]} \geq \sqrt{s}\}\mathrm{d}s = \int_0^h \left(\mathbb{P}\{f^{(i)} \geq \sqrt{s}\}\right)^N \mathrm{d}s$$

$$[\text{from (3.5)}] \leq \int_0^h \left(\frac{h^n - s^{n/2}}{h^n}\right)^N \mathrm{d}s$$

$$\left[t = \frac{s^{n/2}}{h^n}\right] = \frac{2h^2}{n}\int_0^1 (1-t)^N t^{(2/n)-1}\mathrm{d}t = \frac{2h^2}{n}B\left(N+1, \frac{2}{n}\right).$$

This proves the first inequality in (3.8). The second inequality can be proved by applying Theorem 3.4 in [1]. ◻

The above result shows that the standard deviation of $f_{[1]}$ decreases as the number of samples N increases. However, it should be noted that, contrary to the expected value result, this bound is not tight; indeed, numerical experiments testify to a faster decrease with respect to N.

Remark 2 (uniformity test). Apart from its application in section 5 to the convergence analysis of the specific randomized cutting plane scheme proposed in this paper, we point out that the techniques developed in the proof of Theorem 3.1 may have a broader range of application. In particular, based on a similar reasoning, a general qualitative test for checking the goodness of a given mechanism for generating uniform random points in a given convex body $\mathcal{X}$ can be developed. To this end, notice that inequalities (3.5)–(3.6) represent upper and lower bounds on the distribution of the linear function $c^\top x$, where x is uniform in $\mathcal{X}$. Then, given a set of points from an unknown distribution over $\mathcal{X}$, we construct the empirical distribution for this linear function, and, in the spirit of the classical Kolmogorov–Smirnov test, we discard the points as nonuniform if the empirical distribution falls outside these bounds.

Remark 3 (failure of naive randomized schemes). Analyzing inequality (3.3), we see that for fixed dimension n, as the number of samples N goes to infinity, the expected value of $f_{[1]}$ does converge to f^*, as one may expect. However, we also observe that for fixed N, the expected distance $\mathbb{E}\left[f_{[1]} - f^*\right]$ tends to h as the dimension n grows. Notice that this fact implies that the idea of a simplistic randomized scheme, where we draw N samples in $\mathcal{X}$ and adopt the best one as a candidate optimizer, is bound to fail as the dimension grows. Indeed, if we were to impose a relative precision

$$\mathbb{E}\left[f_{[1]} - f^*\right]/h \le \alpha, \quad \alpha < 1,$$

the number of uniform samples to be drawn in $\mathcal{X}$ would depend exponentially on α as $N = \left\lceil \frac{1}{\alpha^n} \right\rceil$. This is not surprising, since it is well known that in general, an exponential number of samples is needed to approximate with a given precision the minimum of a function; see [2]. This latter reasoning further motivates the RCP approach we are proposing in the next section.

4. An RCP algorithm.

Motivated by the results in previous sections, we propose the following randomized modification of the DCG scheme.

Algorithm 2 (RCP algorithm).

```
Input: $\mathcal{X}$
Output: $z_k$
1: $k \Leftarrow 0$, $\mathcal{X}_k \Leftarrow \mathcal{X}$;
2: generate $N_k$ uniform random samples in $\mathcal{X}_k$, $\{x_k^{(1)}, \dots, x_k^{(N_k)}\}$;
3: $z_k \Leftarrow \arg\min_{x \in \{x_k^{(1)}, \dots, x_k^{(N_k)}\}} c^\top x$;
4: $\mathcal{X}_{k+1} \Leftarrow \left\{x \in \mathcal{X}_k \,:\, c^\top (x - z_k) \le 0\right\}$;
5: check Stopping Rule; $k \Leftarrow k+1$; goto 2.
```

The randomized scheme we propose for computing the query point at step k is very simple and intuitively transparent. At step k, we generate N_k samples in $\mathcal{X}_k$; then the next query point z_k is chosen as the random sample providing the minimum to the objective function. The intuition behind Algorithm 2 is illustrated in Figure 4.1.

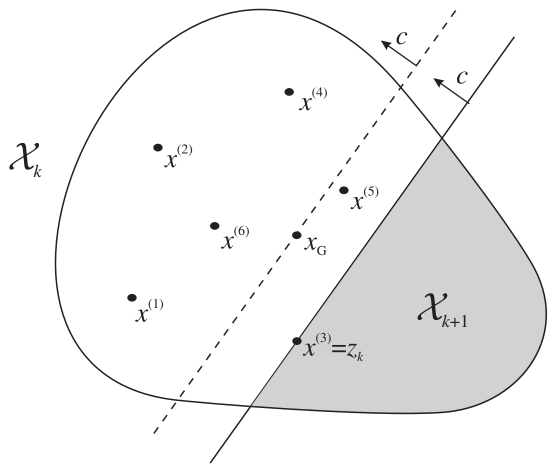

FIG. 4.1. *A sketch of the proposed randomized cutting scheme. The cut at step k is performed at the random point with the smallest cost value (in this case $z_k = x^{(3)}$). The gray area indicates the set $\mathcal{X}_{k+1}$. The dashed line is the hyperplane through x_G, the center of gravity of $\mathcal{X}_k$.*

Clearly, the kth step of the RCP algorithm will perform better than a corresponding step of a deterministic DCG method as long as at least one random point will fall *below* (in terms of the cost function) the center of gravity x_G of the set $\mathcal{X}_k$. Notice that the number of sample points drawn at each iteration is allowed to depend on the iteration index k; this is a key feature that we exploit in section 5.2.2.

In the second part of this paper, we discuss in detail how the proposed scheme can be implemented using techniques based on random walk and present specific applications of this methodology. In the next section we instead concentrate on deriving the theoretical properties of the RCP.

It should be noted that a method of a very similar flavor was proposed in [31], with the cuts being performed through the estimated centers of gravity of the $\mathcal{X}_k$ sets; however, no formal theoretical analysis of the overall scheme has been provided in [31].

5. Probabilistic analysis of RCP.

In this section, we make use of the theory developed so far to perform a probabilistic analysis of the performance of the proposed RCP algorithm.

5.1. Expected convergence rate of RCP.

At step k of the RCP algorithm let us define the following random variable:

$$f_k \doteq c^\top z_k. \tag{5.1}$$

Then the following corollary of Theorem 3.1 shows that the RCP scheme converges in first and second mean and, more important, that the rate of convergence is exponential.

COROLLARY 5.1 (expected convergence of RCP). *Consider the RCP algorithm with $N_k \equiv N$. Then we have*

$$\mathbb{E}\left[f_k - f^*\right] \le \left(\frac{1}{N+1}\right)^{\frac{k}{n}} \mathbb{E}\left[f_0 - f^*\right]; \tag{5.2}$$

that is, the RCP *algorithm converges in mean with rate* $(1/(N+1))^{1/n}$. *Moreover, the* RCP *algorithm converges also in mean square with*

$$\mathbb{E}\left[(f_k - f^*)^2\right] \le \left(\frac{1}{N+1}\right)^{\frac{2k}{n}} \mathbb{E}\left[(f_0 - f^*)^2\right]. \tag{5.3}$$

Proof. Consider step k of the RCP algorithm, and assume that the samples at steps $1, \dots, k-1$ are given. Then f_{k-1} defined as in (5.1) is uniquely determined, and relation (3.3) takes the form (see Theorem 3.1)

$$\mathbb{E}\left[f_k - f^* \,|\, f_{k-1}\right] \le \left(\frac{1}{N+1}\right)^{\frac{1}{n}} (f_{k-1} - f^*), \tag{5.4}$$

where $\mathbb{E}\left[f_k - f^* \,|\, f_{k-1}\right]$ denotes the conditional expectation of $f_k - f^*$ given f_{k-1}, taken with respect to the σ-algebra generated by the multisample at step k.

Then taking expectation with respect to the σ-algebra generated by the samples at previous steps from both sides of the last inequality, by the law of iterated expectations we have

$$\mathbb{E}\left[f_k - f^*\right] \le \left(\frac{1}{N+1}\right)^{\frac{1}{n}} \mathbb{E}\left[(f_{k-1} - f^*\right].$$

Inequality (5.2) is immediately proved by a repeated application of the above relation. The proof of inequality (5.3) follows from Corollary 3.2 by applying exactly the same line of reasoning and is omitted for brevity. $\square$

From inequality (5.2) in the corollary we immediately see that the expected number of steps required by the RCP algorithm to compute an α–optimal solution (i.e., such that $\mathbb{E}\left[f_k - f^*\right] \le \alpha$) is at most

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

where $R \doteq \mathbb{E}\left[f_0 - f^*\right]$. It is interesting to remark that the result is perfectly consistent with intuition; in particular, it is easily verified that (5.2) becomes the probabilistic version of (2.3) when $N = 1$. We conclude that the derived convergence rate reduces to the one in Lemma 2.3 when $N = 1$, while it improves by a factor of $\ln(N+1)$ when $N > 1$.

We note that the results in this subsection are typical of stochastic optimization methods; see for instance [32]. They establish the *property* of convergence in mean for the sequence of estimates *generated by the method*. In particular, it should be stressed that there is no need to compute the expectation analytically. Other types of convergence, e.g., those with *arbitrarily high probability*, can as well be established; they, however, require a different analysis of the method, which is performed in the next section.

5.2. Sample size results. In this section, we perform a different probabilistic analysis of the proposed randomized scheme. This allows us to derive explicit bounds on the number of samples to be drawn at each step of the algorithm in order to guarantee with arbitrarily high probability the desired convergence behavior.

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

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

where the last inequality follows from Proposition 2.1. 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 kth step of the RCP algorithm performs better than the corresponding DCG step.

LEMMA 5.2 (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 that

$$c^\top z_k \leq c^\top \mathsf{cg}(\mathcal{X}_k). \tag{5.5}$$

Proof. The lemma is easily proved by noticing that

$$N_k \geq \frac{\ln \frac{1}{\epsilon}}{\ln \frac{1}{1-1/\mathrm{e}}}$$

implies $(1 - 1/e)^{N_k} \leq \epsilon$ and $(\ln \frac{1}{1-1/\mathrm{e}})^{-1} \approx 2.1802 < 2.2$. ∎

In the next section, we concentrate on the overall behavior of the algorithm and 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 a better performance than DCG.

5.2.2. Overall analysis of RCP. In the theorem below, the probabilistic behavior of the RCP algorithm is characterized. To be more formal, we say that the *RCP algorithm performs better than DCG with probability* $(1 - \epsilon)$ if the probability that at some step k we have that $c^\top z_k \geq c^\top \mathsf{cg}(\mathcal{X}_k)$ is less than ϵ.

THEOREM 5.3 (behavior of RCP). *Given a probability level $\epsilon > 0$, choose*

$$N_k \geq N_{\mathrm{guar}}(\epsilon, k) \doteq 2.2 \ln \frac{1}{\epsilon} + (1.1 + 0.505 \ln k). \tag{5.6}$$

Then the RCP algorithm performs better than DCG with probability $(1 - \epsilon)$.

TABLE 5.1
Behavior of the bound $N_{\rm guar}(\epsilon, k)$ for different values of ϵ and k.

$\lceil N_{\rm guar}(\epsilon, k)\rceil$	$\epsilon = 0.01$	$\epsilon = 0.005$	$\epsilon = 0.001$	$\epsilon = 0.0005$	$\epsilon = 0.00001$
$k = 1$	12	13	17	18	27
$k = 10$	13	14	18	19	28
$k = 100$	14	16	19	21	29
$k = 1,000$	15	17	20	22	30
$k = 10,000$	16	18	21	23	32

Proof. The probability of the algorithm performing worse than DCG is the probability of the event

$$\mathsf{Worse}_\infty \doteq \{\text{at some step } k \text{ it holds } \ c^\top z_k \geq c^\top \mathsf{cg}(\mathcal{X}_k)\}.$$

This probability can be bounded as

$$\begin{aligned} \mathbb{P}\{\mathsf{Worse}_\infty\} &\leq \mathbb{P}\{\mathsf{Worse}_1 \cup \mathsf{Worse}_2 \cup \mathsf{Worse}_3 \cdots\} \\ &= \sum_{k=1}^{\infty} \mathbb{P}\{\mathsf{Worse}_k\} \\ &\leq \sum_{k=1}^{\infty} (1 - 1/e)^{N_k}. \end{aligned}$$

To make this sum finite and equal to the desired probability level ϵ, it is sufficient to select the sequence $\{N_k\}$ such that

$$\left(1 - \frac{1}{e}\right)^{N_k} = \frac{k^{-\alpha}}{\zeta(\alpha)}\, \epsilon, \quad \alpha > 1, \tag{5.7}$$

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

$$\sum_{k=1}^{\infty} \left(1 - \frac{1}{e}\right)^{N_k} = \sum_{k=1}^{\infty} \frac{k^{-\alpha}}{\zeta(\alpha)}\, \epsilon \;=\; \epsilon.$$

Solving (5.7) 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 the bound (5.6). ☐

In other words, Theorem 5.3 guarantees that the overall behavior of the RCP scheme, except for rare runs of the algorithm which can occur with very small probabilty, is better than the one of DCG *at each step*, in the sense that we guarantee with high probability that *at every step*, cutting at z_k, we have a deeper cut than we would have by performing a cut through the center of gravity.

Table 5.1 shows various values of bound (5.6) for given values of ϵ and k. Note that the number of required random samples is indeed very low, even for very small values of ϵ.

6. A hit-and-run implementation of RCP.

In the previous sections, the probabilistic properties of the proposed algorithm have been analyzed in an idealized setting which assumed the existence of a mechanism for generating uniform random samples from the current set $\mathcal{X}_k$. We now briefly discuss how this assumption can be softened using random walks. To this end, we replace Assumption 1 of the uniform oracle with the more implementable one on the availability of a *boundary oracle* as follows.

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

In fact, the boundary oracle assumption can be relaxed to the assumption of having a *membership oracle*. Indeed, from the latter one, a boundary oracle can be always constructed by means of a binary search (e.g., see [24]). However, for a wide range of problems, a BO can be easily formulated in closed form, and in section 7 we present a particular BO for sets specified by linear matrix inequalities.

The availability of a BO is crucial for obtaining implementable versions of the RCP algorithm by means of techniques based on *random walks* in convex bodies (e.g., see [42] for a very detailed survey on these methodologies).

In general, the problem of sampling from a convex body $\mathcal{X}$ has received increasing attention in the last decade. The main reason is that it provides an efficient (polynomial-time) way for estimating the volume of $\mathcal{X}$; see for instance [23, 17, 25]. Specifically, a random walk in $\mathcal{X}$ starts at some 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; for instance, the grid or lattice walk [12], the ball walk [17], and the hit-and-run algorithm.

We adopt the latter methodology as the most promising one, since it has been proven to possess the best known bounds on the number of steps needed to obtain a random sample. The hit-and-run algorithm has been proposed by Turchin [41] and independently later by Smith [38] and is aimed at generating points with approximately uniform distribution in a body. Its properties have been studied in numerous works by Lovász and coauthors (e.g., see the survey paper [42]). This algorithm in its simplest form is recalled next in Algorithm 3.

ALGORITHM 3 (hit-and-run algorithm).

```
Input: $x_0 \in \mathcal{X}$, $M$
Output: $x$ random point belonging to $\mathcal{X}$
1: $y^{(1)} \Leftarrow x_0$;
2: for $i = 1$ to $M$ do
3:     generate a uniform random direction $v \in \mathbb{R}^n$,
4:     $\{\overline{y}, \underline{y}\} = \mathsf{BO}(\mathcal{X}, y^{(i)}, v)$,
5:     generate a uniform point $y^{(i+1)}$ in the segment $[\overline{y}, \underline{y}]$,
6: end for;
7: $x \Leftarrow y^{(M)}$.
```

The random direction v in line 3 can be easily generated as $\xi / \|\xi\|_2$, where $\xi \in \mathbb{R}^n$ is a Gaussian random variable with zero mean and identity covariance matrix.

Among the attractive features of hit-and-run is the so-called polynomial-time mixing property, which was proved by Smith in [38] and later refined by various authors (e.g., see [23, 25]). Roughly speaking, it formulates as follows: Under certain conditions on the geometry of $\mathcal{X}$ and the initial hit-and-run point, in order to obtain a point such that the total variation[1] between its distribution and the uniform distribution on $\mathcal{X}$ is bounded by a given constant, the number of steps M required (mixing time) is *polynomial in the dimension* n. Hence, the results in Theorem 5.3 can be directly adapted to obtain bounds on the sample size which are still polynomial in the dimension n. Specifically, in the paper [8] a theoretical analysis is undertaken showing how the sample-size bounds derived in section 5 can be readily extended to the hit-and-run setup. For completeness, we remark that the analysis [8] assumes that the samples generated by the hit-and-run procedure are independent and exactly uniform. However, the random walk gives only us samples from (i) a distribution that is close to the right one, and (ii) they are only nearly independent. However, there exist standard tricks to circumvent these obstacles. The reader is referred to the exhaustive discussion in [42], where it is shown that the bounds derived assuming independency can be easily extended without loosing the polynomial dependence. In particular, difficulty (i) can be handled by methods sometimes known as divine intervention (see, e.g., [42]). Similarly, (ii) can be tackled by introducing the notion of μ-independence.[2]

Remark 4 (hit-and-run implementation of the RCP algorithm). We point out that, in a hit-and-run implementation of the RCP algorithm, Algorithm 3 has to be run N_k times—one for every random point. This is necessary to theoretically guarantee (almost)-independent hit-and-run samples. As a result, in this process we generate MN_k points, where M is the mixing time. However, in a practical implementation, nothing prevents us to consider all the points $y^{(i)}$ generated by the hit-and-run procedure, calculating the objective function for each of them and selecting the best one. In this way, one obtains a practical improvement, still guaranteeing a performance at least as good as the original scheme.

We prefer not to dwell further on this topic, which has been studied in the recent literature. The interested reader is referred to the paper [3], which provides a general exposition on the construction of cutting plane schemes based on hit-and-run and discusses at a theoretical level several implementation issues.

We remark that results along this line are of great theoretical importance, showing that hit-and-run schemes can be implemented in polynomial time, but they are still unsatisfactory from a practical viewpoint. The main reason is that the constants involved in the sample size bounds available so far are usually very large, thus making the method not good in practice.

On the other hand, numerical experience shows that hit-and-run usually performs much better than predicted theoretically. This fact is acknowledged by most authors and practitioners. For instance, Bertsimas and Vempala in [3] state that *"In practice, drawing random samples from convex sets might be much faster than the known worst-case bounds; also, sampling convex sets is an active research area and there might well be faster sampling methods in the future that would directly improve the complexity of the random walk algorithm."*

This is also the main reason that convinced us to keep separate the analysis of the theoretical setup of section 5. Equally importantly, in the RCP setup, not

[1]For two probability distributions $\mathbb{P}_1$, $\mathbb{P}_2$ on the same underlying σ-algebra on $\mathcal{X}$, their *total variation* distance is defined as $\|\mathbb{P}_1-\mathbb{P}_2\|_{\mathrm{tv}} \doteq \sup_{A\subseteq\mathcal{X}} |\mathbb{P}_1(A)-\mathbb{P}_2(A)|$, where A range over measurable subsets of $\mathcal{X}$.

[2]Two random variables ξ, ζ are said to be μ-independent if $\sup_{A,B} |P(\xi \in A; \zeta \in B) - P(\xi \in A)P(\zeta \in B)| \leq \mu$ and A and B range over measurable subsets of the ranges of ξ and ζ, respectively.

only the theoretical bounds on the number of samples are pessimistic in practice, but the above-mentioned conditions on the geometry of the feasible set are also very hard to guarantee in the process of cutting. As a result, implementable hit-and-run–based versions of the algorithm do differ from Algorithm 2, as better described in section 7.1. In particular, the so-called isotropization performed over the $\mathcal{X}_k$ sets (see below) improves the convergence considerably.

7. Application to linear matrix inequalities.

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

$$\min c^\top x \qquad \text{subject to} \qquad F(x) \doteq F_0 + \sum_{i=1}^{n} x_i F_i \preceq 0, \tag{7.1}$$

where $c \in \mathbb{R}^n$ and $F_i = F_i^\top \in \mathbb{R}^{m\times m}$, $i = 0, \ldots, n$, are known symmetric matrices; the notation $F \preceq 0$ stands for negative semidefiniteness of the matrix F. The constraint inequality in (7.1) is called a linear matrix inequality, and the convex set

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

is referred to as the feasible domain of this linear matrix inequality. To exclude trivialities, we assume that the set $\mathcal{X}_{\text{LMI}}$ is nonempty and bounded (this is equivalent to assuming that $\sum_{i=1}^{n} x_i F_i$ is sign indefinite for all x).

The optimization problem (7.1) is known to be one of the key problems in modern convex optimization [4]. It has numerous applications in various fields of system theory, control, and optimization, and at present there exist efficient solution techniques based on interior-point methods; e.g., see [27]. It is because of this generality and availability of "alternative" solvers that we exemplify here the use of our method for this widely adopted convex optimization setup and compare the numerical results with those obtained using one of the "standard" MATLAB-based toolboxes, SeDuMi [39] together with the most popular interfaces YALMIP [22] and `cvx` [15].

To apply our method, we need a semidefinite BO for the set $\mathcal{X}_{\text{LMI}}$, i.e., the intersections of a 1D line and the boundary of $\mathcal{X}_{\text{LMI}}$ are to be computed efficiently. This is accomplished via the following lemma developed in [30]; see also the work [7] for a similar result.

LEMMA 7.1 (semidefinite boundary oracle). *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*

$$\underline{\lambda} = \begin{cases} \max\limits_{\lambda_i<0} \lambda_i, & \\ -\infty & \text{if all } \lambda_i > 0, \end{cases} \qquad \overline{\lambda} = \begin{cases} \min\limits_{\lambda_i>0} \lambda_i, & \\ +\infty & \text{if all } \lambda_i < 0, \end{cases}$$

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

In the setup of this paper, assume that $y \in \mathcal{X}_{\text{LMI}}$, and $v \in \mathbb{R}^n$ is a (random) direction. We then have

$$F(y + \lambda v) = F(y) + \lambda\big(F(y) - F_0\big) \doteq A + \lambda B,$$

and using Lemma 7.1, the desired intersection points of the line and the boundary of $\mathcal{X}_{\text{LMI}}$ at step 4 of Algorithm 3 are given by $\underline{y} = y + \underline{\lambda} v$ and $\overline{y} = y + \overline{\lambda} v$. To compute the intersection points with the boundary of a current set $\mathcal{X}_k$, the additional linear condition $c^\top(x - z_{k-1}) \leq 0$ 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 symmetric matrices (which is for instance efficiently implemented in MATLAB). Hence, the BO for linear matrix inequalities is accurate and "cheap" enough for matrices of quite large dimensions, as confirmed by the numerical simulations discussed in the next section.

7.1. Numerical experiments. 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 remark again that, in the spirit of the discussion at the end of section 6, the implemented procedure differs from Algorithm 2; it involves certain semiheuristic tricks and would require a careful theoretical analysis that we aim to perform in the subsequent papers.

(a) *Initialization.* To implement the hit-and-run procedure, we need to find an initial feasible point $x_0 \in \mathcal{X}_{\mathrm{LMI}}$. 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 $\{x = 0, \gamma = \max \operatorname{eig}\big(F(0)\big)\}$ as an initial feasible solution. If the optimal solution $\{\tilde{x}, \tilde{\gamma}\}$ of this problem is such that $\tilde{\gamma} > 0$, then the original problem (7.1) is infeasible; otherwise we take $x_0 = \tilde{x}$ as initial feasible point. Notice also that, for the specific semidefinite programs, 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 us to find a good initial point to pass to the algorithm. However, it should be noted that, after the first step of the RCP algorithm, by construction, the hit-and-run starting point z_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 [5] can be directly applied to our setup. In our case, we adopted a method based on boundary points, as better described in point (d) below.

(b) *Number of points.* As it follows from the discussion in section 6, the number N_k of random points suggested by Theorem 5.3 has to be enlarged because of the mixing time required for hit-and-run algorithm. Also, extra points are required to bring the set in near-isotropic position, as described below. For these reasons, in the present implementation, we decided to lean on $N_k \equiv N \gg N_{\mathrm{guar}}(0.01, 10000) = 16$ points at every iteration.

(c) *Isotropization.* The theoretical bounds on the required number of hit-and-run points depend explicitly on the radii of two balls, respectively, inscribed and inscribing the set $\mathcal{X}$. While 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 $(k-1)$, the set $\mathcal{X}_k$ is obtained from the original set $\mathcal{X}$ by cutting a portion defined by the hyperplane $\mathcal{H}_{k-1} = \big\{x \in \mathbb{R}^n : c^\top(x - z_{k-1}) = 0\big\}$, and there is no guarantee that it still contains a ball of a given radius. A well-accepted technique (see for instance [3]) to overcome this problem is to perform an affine transformation of $\mathcal{X}_k$ to bring it in near-isotropic position.[3]

[3]A convex set $\mathcal{X} \subset \mathbb{R}^n$ 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 $\mathsf{cg}(\mathcal{X}) = 0$ and the covariance matrix of the uniform distribution over $\mathcal{X}$ has eigenvalues between $\frac{1}{2}$ and $\frac{3}{2}$.

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

$$\bar{y} \doteq \frac{1}{N} \sum_{i=1}^{N} y^{(i)} \quad \text{and} \quad Y \doteq \frac{1}{N} \sum_{i=1}^{N} (y^{(i)} - \bar{y})(y^{(i)} - \bar{y})^\top \tag{7.2}$$

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

$$\widetilde{\mathcal{X}} \doteq \left\{ x \in \mathbb{R}^n \; : \; Y^{\frac{1}{2}} x + \bar{y} \in \mathcal{X} \right\}. \tag{7.3}$$

It was shown by Rudelson [36] that the complexity of bringing a set in near-isotropic position with high probability is of the order of $\mathcal{O}(n \log^2 n)$. In this paper, we adopted the implicit isotropization technique proposed by Kannan, Lovász, and Simonovits in [17]. In that case, the set $\mathcal{X}$ is left unchanged, and the direction vector for step 2 of Algorithm 3 is taken in the form $v = Y^{1/2}\eta$, where Y is computed as (7.2) and the vector $\eta \in \mathbb{R}^n$ is uniformly distributed on the surface of the unit hypersphere.

(d) *Heuristics.* Various heuristic schemes proved effective in our simulations.

First, as regards the isotropization step, we notice that, as a byproduct of the hit-and-run algorithm at the kth iteration, we have $2N$ points on the boundary of the set $\mathcal{X}_k$ (endpoints $\underline{y}$, $\overline{y}$). The transformation matrix Y in (7.2) 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 kth step that fall into $\mathcal{X}_{k+1}$ were used to perform the initialization phase of point (a). More precisely, the initial point for the hit-and-run 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" hit-and-run point as described above, then there exist $N_b \geq 1$ such points (at least the endpoint of the chord associated with the "best" hit-and-run 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.

Finally, a negative definite matrix was considered singular if its largest eigenvalue was greater than -10^{-10}; i.e., $\max \operatorname{eig}(F(z_k)) > -10^{-10}$ was adopted as a stopping rule.

We do not discuss here the choice of other numerical constants involved in computations and various extra modifications of the basic scheme (e.g., regularization constants, alternative ways for computing the transformation matrix Y, etc.); this would be a subject of a separate paper primarily focused on implementation issues.

(e) *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 $\mathcal{X}_{\text{LMI}}$; for simplicity, we chose $F_0 \prec 0$ in the form

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

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

$$\begin{aligned} &\text{(a)} \quad M = 2\,\texttt{rand}(m/2) - 1, \\ &\text{(b)} \quad M = M + M^\top, \\ &\text{(c)} \quad F_i = \texttt{blkdiag}(M; -M) \end{aligned}$$

in order to guarantee $\mathcal{X}_{\rm LMI}$ to be bounded and to ensure the existence of a finite solution. In a number of examples, step (b) above was implemented as

$$M = \texttt{triu}\, M + (\texttt{triu}\, M)^\top - \texttt{diag}(\texttt{diag}(M)).$$

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 $n = 1,000$. Without loss of generality, the vector c in the objective function was taken as $c = (1\ 0\ \dots\ 0)^\top$.

We note that the test problems chosen are illustrative, thus giving a first impression on the potentials of our method; of course, a deeper analysis is needed, which will be performed in the subsequent papers. In the experiments, the method demonstrated pretty stable performance; as far as the widely used MATLAB-based realizations of interior-point methods (the `solvesdp` routine in SeDuMi toolbox) are concerned, it 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 obtained 7 to 9 exact digits after 45 to 55 iterations with $N = 200$ hit-and-run points at each step. In other words, the observed rate of convergence $(f_k - f^*)/(f_{k-1} - f^*)$ was approximately 0.65 to 0.7 as compared to the expected theoretical rate 0.59 for this dimension $n = 10$ (see (5.4)). The same accuracy was typically observed after 30 to 40 steps if $N = 500$ hit-and-run points were used.

In the second set of experiments the method was tested on semidefinite programs having large dimensions of the design vector, $n = 1{,}000$ and $m = 10$. Typically, the method reproduced 7 to 8 exact decimal digits for the function value after 20 to 30 iterations ($N = 1{,}500$ points were used). Hence, for such high-dimensional problems, a much faster convergence was observed than the predicted theoretical one 0.99. Also, 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” optimal point x^*, e.g., $\max \operatorname{eig}(F(x^*)) \approx 2 \cdot 10^{-7} > 0$ was observed.

The third set of experiments was conducted with problems having worst-case geometry, where the feasible domain was the worst-case cone configuration derived in section 3. For such set, the bound on the rate of convergence of the RCP algorithm given by (5.4) is known to be attained. In the experiments, the matrices F_i were chosen so as to yield

$$\mathcal{X}_{\rm LMI} \ = \ \{x \in \mathbb{R}^n\colon\ \|x\|_1 \le 1,\ x_1 < 0\},$$

so that 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 hit-and-run points that would be needed to correctly restore the Y matrix.

Specifically, for the worst-case geometry problem with $n = 10$ optimization variables and quite large dimension $m = 1,024$ 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 comparable to 0.69, which is given by (5.4).

In view of the numerical results above, we mention the paper [31], where a similar randomized algorithm was proposed. At every step, the cut was performed through the estimated center of gravity (averaged hit-and-run points) rather than through the best point as above, and an additional “projection” step was used that largely accelerated the convergence. The results of numerical tests with the same data were similar to those reported here; however, no formal theoretical claims were made in [31] regarding the convergence properties.

Summarizing, the presented examples testify to a good numerical performance of the proposed algorithm. In our computational experience, the approach is comparable with interior point methods in terms of convergence rate and accuracy. Moreover, it is felt that the schemes proposed here are quite promising to outperform classical deterministic optimization methods in the following two directions. The first one relates to large-dimensional problems; e.g., those where the linear matrix inequalities are specified in matrix, rather than canonical vector form. For this case, ad hoc matrix implementations of the boundary oracle can be devised, as discussed in [31]. The second direction is robust optimization, as for instance semidefinite programs with linear matrix inequality constraints where the matrix coefficients subject to norm-bounded uncertainties. In this case, the only component of the overall scheme that needs to be modified is BO. Such *robust* BO is also discussed in [31].

On the other hand, it should be noted that the hit-and-run procedures still suffer from several implementation difficulties and require ad hoc arrangements which currently limit their practical use. However, the field of random sampling by random walk is rapidly evolving, e.g., see [11], and increasingly efficient schemes are being devised. In this regard, we stress again that the theoretical results presented in sections 3–5 are of full generality and apply to any random point generation scheme.

Finally, we remark that the idea of combining random walk schemes with classical optimization methods is gaining increasing interest. For instance, in [29] the use of logarithmic barrier functions is combined with a hit-and-run scheme to solve convex optimization problems, while [18] proposes a specific combination of interior point methods and random walks in a polytope for solving linear programs.

Acknowledgment. This work was partially supported by a bilateral project between the Italian National Research Council (CNR) and the Russian Academy of Sciences (RAS). The support of RSTL-CNR is also acknowledged.

REFERENCES

[1] H. Alzer, *Some Beta function inequalities*, Proc. R. Soc. Edinburgh Sect. A, 113 (2003), pp. 731–745.

[2] E.-W. Bai, R. Tempo, and M. Fu, *Worst-case properties of the uniform distribution and randomized algorithms for robustness analysis*, Math. Control Signals Systems, 11 (1998), pp. 183–196.

[3] D. Bertsimas and S. Vempala, *Solving convex programs by random walks*, J. ACM, 51 (2004), pp. 540–556.

[4] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, SIAM Stud. Appl. Math. 15, SIAM, Philadelphia, 1994.

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

[6] G. Calafiore and F. Dabbene, *Probabilistic and Randomized Methods for Design under Uncertainty*, Springer, London, 2006.

[7] G. Calafiore, *Random walks for probabilistic robustness*, in Proceedings of the IEEE Conference on Decision and Control, 2004, pp. 5316–5321.

[8] F. Dabbene, P. Shcherbakov, and B. Polyak, *A randomized cutting plane scheme with geometric convergence: Probabilistic analysis and SDP applications*, in Proceedings of the 47th IEE Conference on Decision and Control, 2008.

[9] F. Dabbene, *A randomized cutting plane scheme for convex optimization*, in Proceedings of the IEEE Multiconference on Systems and Control, San Antonio, TX, 2008.

[10] H. David and H. Nagaraja, *Order Statistics*, 3rd ed., Wiley, New York, 2003.

[11] P. Diaconis, *The Markov chain Monte Carlo revolution*, Bull. Amer. Math. Soc. (N.S.), 46 (2009), pp. 179–205.

[12] M. Dyer, A. Frieze, and R. Kannan, *A random polynomial-time algorithm for approximating the volume of convex bodies*, J. ACM, 38 (1991), pp. 1–17.

[13] R. Gardner, *The Brunn–Minkowski inequality*, Bull. Amer. Math. Soc. (N.S.), 39 (2002), pp. 355–405.

[14] J.-L. Goffin and J.-P. Vial, *Convex non-differentiable optimization: A survey focused on the analytic center cutting plane method*, Optim. Methods Softw., 17 (2002), pp. 805–867.

[15] M. Grant, S. Boyd, and Y. Ye, *Disciplined convex optimization*, in Global Optimization: From Theory to Implementation, Nonconvex Optim. Appl., L. Liberti and N. Maculan, eds., Springer, New York, 2006, pp. 155–210.

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

[17] R. Kannan, L. Lovász, and M. Simonovits, *Random walks and an $O^*(n^5)$ volume algorithm for convex bodies*, Random Structures Algorithms, 11 (1997), pp. 1–50.

[18] R. Kannan and H. Narayanan, *Random walks on polytopes and an affine interior point method for linear programming*, in Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC 2009), 2009, pp. 561–570.

[19] J. Kelley, Jr., *The cutting plane method for solving convex programs*, J. Soc. Indust. Appl. Math., 8 (1960), pp. 703–712.

[20] L. Khachiyan, *A polynomial time algorithm for linear programming*, Soviet Math. Dokl., 20 (1979), pp. 191–194.

[21] A. Levin, *On an algorithm for the minimization of convex functions*, Soviet Math. Dokl., 6 (1965), pp. 286–290 (in Russian).

[22] J. Löfberg, *YALMIP: A toolbox for modeling and optimization in MATLAB*, in Proceedings of the CACSD Conference, Taipei, Taiwan, 2004, pp. 284–289.

[23] L. Lovász and M. Simonovits, *Random walks in a convex body and an improved volume algorithm*, Random Structures Algorithms, 4 (1993), pp. 359–412.

[24] L. Lovász, *Random walks on graphs: A survey*, in Combinatorics, Paul Erdös is Eighty, V. T. S. D. Miklós and T. Szönyi, eds., János Bolyai Mathematical Society, Budapest, 1996, pp. 353–398.

[25] L. Lovász, *Hit-and-Run mixes fast*, Math. Program., 86 (1999), pp. 443–461.

[26] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, Cambridge, MA, 1995.

[27] Y. Nesterov and A. Nemirovski, *Interior point polynomial algorithms in convex programming*, SIAM Stud. Appl. Math. 13, SIAM, Philadelphia, 1994.

[28] D. Newman, *Location of the maximum on unimodal surfaces*, J. ACM, 12 (1965), pp. 395–398.

[29] B. Polyak and E. Gryazina, *Markov chain Monte Carlo method exploiting barrier functions with applications to control and optimization*, in Proceedings of the IEEE Multiconference on Systems and Control, Yokohama, 2010, pp. 1553–1557.

[30] B. Polyak and P. Shcherbakov, *The D-decomposition technique for linear matrix inequalities*, Autom. Remote Control, 67 (2006), pp. 159–174.

[31] B. Polyak and P. Shcherbakov, *A randomized method for solving semidefinite programs*, in Proceedings of the 9th IFAC Workshop—Adaptation and Learning in Control and Signal Processing (ALCOSP'07), 2007; also available online from IPACS Electronic Library at http://lib.physcon.ru/.

[32] B. Polyak, *Introduction to Optimization*, Optimization Software, New York, 1987.

[33] L. Rademacher and S. Vempala, *Testing geometric convexity*, Foundations of Software Technology and Theoretical Computer Science (FSTTCS-04), Lect. Notes Comput. Sci. 3328, Springer, 2005, pp. 207–224.

[34] L. Rademacher, *Approximating the centroid is hard*, in Proceedings of the 24th Annual Symposium on Computational Geometry, 2007.

[35] 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 (1916), pp. 241–258.

[36] M. Rudelson, *Random vectors in the isotropic position*, J. Funct. Anal., 164 (1999), pp. 60–72.

[37] N. Shor, *Cut-off method with space dilation in convex programming problems*, Cybernet., 13 (1977), pp. 94–96.

[38] R. Smith, *Efficient Monte-Carlo procedures for generating points uniformly distributed over bounded regions*, Oper. Res., 32 (1984), pp. 1296–1308.

[39] J. Sturm, *Using SeDuMi* 1.02*, a MATLAB toolbox for optimization over symmetric cones*, Optim. Methods Softw., 11–12 (1999), pp. 625–653.

[40] R. Tempo, G. Calafiore, and F. Dabbene, *Randomized Algorithms for Analysis and Control of Uncertain Systems*, Comm. Control Engrg. Ser., Springer, London, 2004.

[41] V. F. Turchin, *On calculation of multi-dimensional integrals via the Monte Carlo method*, Theory Probab. Appl., 16 (1971), pp. 720–724.

[42] S. Vempala, *Geometric random walks: A survey*, Combin. Comput. Geom., 52 (2005), pp. 573–611.

[43] D. Yudin and A. Nemirovski, *Informational complexity and efficient methods for solving complex extremal problems*, Matekon, 13 (1977), pp. 25–45.