Stochastic Approach to a Class of Convex Optimization Problems

Boris T. Polyak, Pavel S. Shcherbakov

***Abstract*— We propose a new approach to solving a wide class of optimization problems which fall into the broad framework of linear matrix inequalities and semidefinite programming. This approach based on randomization and cutting hyperplane ideology also covers robust statements of the problem. The proposed method is easy to implement, and it might be particularly useful in various aspects of functioning and applications of quantum computers.**

I. Introduction

Prospective computers based on quantum logic can be efficiently modelled as a dynamic system subjected to uncertainty, and the architecture ideology of these quantum computers (QC) suggests use of stochastic tools to describe and control their functioning, e.g., see [1]. On the other hand, such new devices are expected to be especially powerful when implementing various randomized algorithms of data processing and structuring, image recognition, clustering, etc. Also, peculiar to these new tasks are massive arrays of information corrupted by exogenous perturbations and subjected to uncertainty in the model description. In other words, in the new line of research related to quantum computations, the development of new *stochastic* optimization methods and *robust* approaches are of great current importance both in optimizing the functioning of a QC and solving typical applied problems for which use of QCs is supposed to be highly advantageous. This is also in agreement with the recent results on use of randomized algorithms in computations, control, and system theory, see [2].

In this note we consider a broad class of optimization problems of this type and propose a new approach to dealing with them. The problems under consideration are formulated in terms of linear matrix inequalities (LMI) and reduce to optimizing a linear function subjected to these constraints, referred to as semidefinite programming (SDP). Many problems in various areas such as optimization, control, estimation of reachability domains of dynamic systems, optimal control, to name just a few, are reducible to such a setup, e.g., see [3].

Our approach is based on estimating the center of gravity x^c of convex bodies by means of *random walk* using the new notion of *boundary oracle*. This estimate of x^c is then used in a new modification of the *cutting hyperplane* method aimed at reducing the value of the objective function.

Of a particular importance is a generalization of the method to robust statements of the problem where the coefficient matrices in the LMI constraints are subjected to additive norm-bounded uncertainties. In this case, the *robust boundary oracle* is devised and the method appropriately modifies.

It should be also noted that the random walk algorithm that we use for estimating the center of gravity is expected to have an independent interest in the QC-related problems.

In this paper, a description of the approach and its main components are given, the implementation issues are discussed and the results of numerical simulations on classes of test problems are presented. We used the well-know MATLAB package as the main computational tool.

Authors are with Institute of Control Science, Profsoyuznaya str., 65, 117997 Moscow, Russia; (boris,sherba)@ipu.ru

II. Problem Formulation

We concentrate our attention at the following problem:

$$\min c^{\mathsf{T}} x \quad \text{s. t.} \quad A(x) \doteq A_0 + \sum_{i=1}^{n} x_i A_i \ \leq \ 0, \qquad (1)$$

where $c \in \mathbb{R}^n$, $A_i \in \mathbb{R}^{m\times m}$, $i = 0, \ldots, n$, are known symmetric matrices and the notation $A \leq 0$ stands for the negative semidefiniteness of the matrix A, i.e., $y^{\mathsf{T}} A y \leq 0$ for all $y \in \mathbb{R}^m$. The inequality constraint in (1) is referred to as a *linear matrix inequality* in the vector variable x, and the convex set

$$D_{feas} = \{x \in \mathbb{R}^n : \ A(x) \leq 0\} \qquad (2)$$

is said to be the *feasibility domain* of the problem.

This is a well-defined constrained convex optimization problem, which plays a central role in the theory of linear matrix inequalities [3], and at present there exist efficient solution methods, e.g., see [4]. This paper is aimed at developing a new approach based on different ideas using randomization of the original setup. Generalization of the approach are also provided; in particular, in robust statements, where the matrices A_i are uncertain, the problem can as well be solved using the proposed approach, which is not often the case with the existing techniques.

III. Description of the Approach

Assuming that D_{feas} is nonempty and bounded, we generate N_{hr} points uniformly distributed inside this domain and take their average x^0 as an estimate of its center of gravity. A specific hyperplane is then drawn through the point x^0 to cut off the "idle" portion of D_{feas} thus reducing it to a smaller set D_1. The next step of the method is performed with the set D_1 instead of D_{feas}, so that we obtain a sequence of embedded sets D_k along with the estimates x^k for their centers of gravity and the converging sequence $f_k = c^{\mathsf{T}} x^k$ estimating the minimum of the objective function.

The structure of the method is schematically presented in Fig. 1.

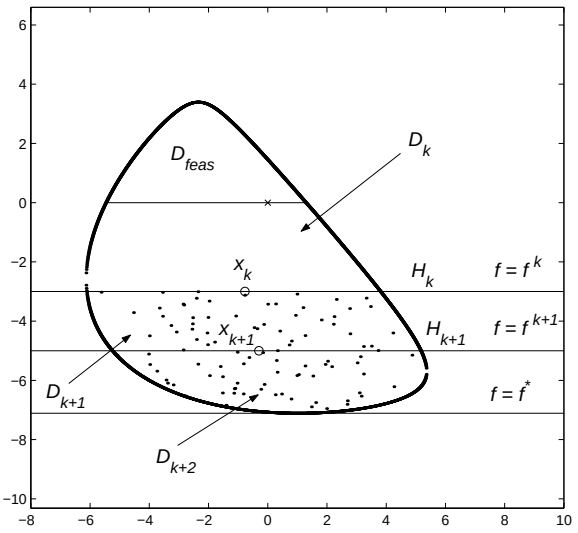

Fig. 1. The basic scheme of the method.

In this section we consider the components of the approach in more detail and later provide certain heuristics towards their validity and efficiency.

A. Randomization: Hit-and-Run

With the first component we are aimed at estimating the center of gravity of the sets D_k. This is performed by means of a random walk algorithm developed in [5] under the name Hit-and-Run (HR). This algorithm applies to a bounded convex body $D \in \mathbb{R}^n$ and produces a random point z distributed "approximately" uniformly over D.

Specifically, let z^0 be an initial point selected at random in the interior of D, and let z^j be the point obtained at the jth step of the algorithm. A random direction $y \in \mathbb{R}^n$ is generated, for instance, in the form $\xi/\|\xi\|_2$, where ξ is a Gaussian random vector with zero mean and identity covariance matrix. The one-dimensional line $z^j + \lambda y$ is considered and the points $\underline{z}^j$ and $\overline{z}^j$ of its intersection with the boundary of D are computed. The next point z^{j+1} is then generated randomly uniformly on the chord $[\underline{z}^j, \overline{z}^j]$.

Under mild conditions on the choice of initial z^0, the shape of the set D, etc., the HR-algorithm yields a practically nice approximation to the uniform distribution over D. The arithmetic mean of the points z^j is then adopted as an estimate of the true center of gravity of D.

Numerous practical aspects of improving the performance of the HR-algorithm as applied to the sets D_k will be discussed in the sections to follow.

B. Semidefinite Boundary Oracle

In order to implement the HR-algorithm, the intersection points of a one-dimensional line with the boundary of the set D_k should be computed. This leads to the notion of *boundary oracle* — a procedure that computes efficiently such points or reports on the absence of intersections. The boundary oracle for sets specified by LMIs was proposed in [6]; it is based on the following lemma.

Lemma 1: Let $A < 0$ and $B = B^{\mathsf{T}}$, 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

$$\underline{\lambda} \;=\; \begin{cases} \max\limits_{\lambda_i<0} \lambda_i, & \\ \\ -\infty, & \text{if all} \quad \lambda_i > 0; \end{cases} \tag{3}$$

and

$$\overline{\lambda} \;=\; \begin{cases} \min\limits_{\lambda_i>0} \lambda_i, & \\ \\ +\infty, & \text{if all} \quad \lambda_i < 0, \end{cases} \tag{4}$$

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

In the setup of this paper, assume that $z^i \in D_{feas}$, and y is a random direction. We have

$$A(z^i + \lambda y) \;=\; A(z^i) + \lambda \sum_{i=1}^{n} y_i A_i \;\doteq\; A + \lambda B,$$

and using Lemma 1, the desired boundary points of D_{feas} are given by $\underline{z}^i = z^i + \underline{\lambda} y$ and $\overline{z}^i = z^i + \overline{\lambda} y$. To compute the boundary points of the current set D_k, it now remains to account for the extra linear constraint provided by the hyperplane (see subsection C below), which is immediate.

Strictly speaking, the intersection points can be found numerically using one-dimensional search. However, this procedure is to be performed at every step of the HR-algorithm; hence it may be time consuming. Instead, the boundary oracle devised requires only the computation of eigenvalues which is extremely fast and accurate as implemented in MATLAB.

We also note that a similar boundary oracle can be formulated for other commonly used constraints such as linear algebraic, quadratic matrix inequalities, etc. In other words, together with the HR-algorithm, the boundary oracle concept can be exploited in a wide range of optimization problems.

C. Cutting Hyperplane

Having an HR-estimate x^0 for the center of gravity of the set $D_0 \doteq D_{feas}$, we proceed by considering the new set

$$D_1 = \{x \in D_0 \colon\, c^{\mathsf{T}}(x - x^0) \le 0\} \subset D_0$$

obtained by cutting off a portion of D_0 using the hyperplane

$$H_0 \;=\; \{x \in \mathbb{R}^n \colon\, c^{\mathsf{T}}(x - x^0) = 0\}$$

through the point x^0. For this convex set D_1 in turn, we perform the HR-algorithm to obtain an estimate x^1 for its center of gravity and construct the hyperplane H_1 through x^1, which defines the set $D_2 \subset D_1$, etc. As a result, we obtain a sequence of embedded sets D_k and a sequence of points x^k having the property $f^{k+1} < f^k$, where $f^k = c^{\mathsf{T}} x^k$ denotes the value of the objective function at the point x^k.

The estimated *rate* of monotone decrease of the sequence f^k can be deduced from the following result on the measures of symmetry of convex bodies [8].

Lemma 2: Let $D \subset \mathbb{R}^n$ be a convex bounded set and $g \in D$ be its center of gravity. Denote by P an arbitrary $(n-1)$-dimensional hyperplane through g, and let P_1 and P_2 be the two hyperplanes supporting to D and parallel to P. Denote by

$$r(P) \;\doteq\; \frac{\min\{\mathbf{dist}(P,P_1),\, \mathbf{dist}(P,P_2)\}}{\max\{\mathbf{dist}(P,P_1),\, \mathbf{dist}(P,P_2)\}}$$

the ratio of the distances from P to P_1 and P_2, respectively. Then

$$\min_P r(P) \;\geq\; \frac{1}{n}\,.$$

This result immediately applies to our method. Indeed, with P_1 and P being the two hyperplanes through the two successive points x^k and x^{k+1} as described above, and P_2 being the supporting hyperplane through the optimal point $x^* = \arg\min c^{\mathsf{T}}x$, and assuming that the *exact* value of the center of gravity is available, we arrive at the following estimate:

$$f^{k+1} - f^* \;\leq\; \varkappa(f^k - f^*), \quad \varkappa = \frac{n}{n+1}\,, \qquad (5)$$

where f^* is the optimal value of the objective function. In other words, the method is expected to have a guaranteed geometric rate of convergence.

Importantly, this lemma had never been used in optimization, in contrast to similar results on the guaranteed *volumetric* reduction, which are typical to various modifications of the ellipsoid method.

Note however that direct application of Lemma 2 to the performance evaluation of the method is complicated by the fact that the quantity x^k obtained as an outcome of the HR-algorithm is represented by the sum of dependent random vectors. Hence, the properties of this statistical estimate of the center of gravity of D_k are hard to establish. As a result, the approach in its present form lacks severe theoretical justification, and in this paper we provide an experimental study of the method.

IV. Presence of Uncertainty

In practice, the data defining the structure of the problem are unavoidably corrupted by unknown noise, uncertainty in the model description, etc. Here we consider the situation where the matrix coefficients in the linear matrix inequality in (1) are not known precisely but only to the accuracy of additive norm-bounded matrix uncertainty. This gives raise to the *robust* statement of the SDP problem, and in this section we propose the appropriate robust modification of the method described above.

Namely, the matrices A_i are given by

$$A_i = A_i^0 + \Delta_i,$$

where A_i^0 are the known *nominal* values, and the real symmetric $m \times m$ matrix uncertainties $\Delta_i = \Delta_i^{\mathsf{T}}$ are not known but bounded in the spectral norm:

$$\|\Delta_i\| \;\leq\; \varepsilon_i, \quad i = 1, \dots n,$$

where the numbers $\varepsilon_i \geq 0$ are given. Such uncertainties will be referred to as *admissible*. Respectively, the *robustly feasible* domain of the uncertain LMI is defined as

$$D_{feas}^{rob} \;=\; \big\{x \in \mathbb{R}^n\colon\; A(x,\Delta) \leq 0 \quad \forall \text{ admissible } \Delta\big\},$$

where it is denoted

$$A(x,\Delta) \;=\; A_0^0 + \Delta_0 + \sum_{i=1}^n x_i(A_i^0 + \Delta_i).$$

In the robust statement, the SDP problem (1) is formulated as the minimization of the same objective function $c^{\mathsf{T}}x$ over the robustly feasible domain. For simplicity, we do not address the issue of nonemptiness of D_{feas}^{rob} assuming that the nominal LMI is feasible and the levels ε_i of uncertainty are small enough to guarantee $D_{feas}^{rob} \neq \emptyset$.

For robust statements of the SDP problem, there are only limited results available in the literature; e.g., see [9]. The robust modification of the method described above remains the same as in the original uncertainty-free case with the only difference that a *robust semidefinite oracle* should be developed. This oracle is based on the result in [6] on the radius of nonsingularity for symmetric matrices, and we summarize it in the lemma below.

Lemma 3: Let $A(x,0) < 0$. For any $y \in \mathbb{R}^n$, the maximal and minimal values of λ retaining the negative definiteness of the matrix $A(x+\lambda y, \Delta)$ for all admissible perturbations Δ are given by the two solutions $\underline{\lambda}^{rob}$, $\overline{\lambda}^{rob}$ of the nonlinear equation in the scalar variable λ

$$\left\|\Big(A_0 + \sum_{i=1}^n (x_i + \lambda y_i)A_i\Big)^{-1}\right\| \;=\; \frac{1}{\varepsilon_0 + \sum\limits_{i=1}^n |x_i + \lambda y_i|\,\varepsilon_i}$$

on the interval $[\underline{\lambda},\, \overline{\lambda}]$ (3)–(4).

Similarly to the non-robust statement, this result is interpreted to mean that, given a point $x \in D_{feas}^{rob}$ and a direction y, the points $\underline{x}^{rob} = x + \underline{\lambda}^{rob}y$ and $\overline{x}^{rob} = x + \overline{\lambda}^{rob}y$ belong to the boundary of the robustly feasible domain. This makes immediate the implementation of the HR-algorithm over D_k^{rob} and evaluating the center of gravity.

V. Implementation Issues and the Results of Simulations

In the experiments, various purifications of the approach described above were implemented, leading to essential acceleration of convergence above the guaranteed rate (5) given by Lemma 2. Attention has been paid to various modifications of the HR-algorithm, averaging techniques for computing the center of gravity from the HR-points, selecting the initial point, stopping rules, etc. We describe some of the most crucial issues.

Projective step: In the original formulation of the method, the estimate $\hat{x}$ of the center of gravity of D_{k+1} is adopted as the next iteration x^{k+1}. Instead, we can perform a "large" step from the current x^k in the direction $(\hat{x} - x^k)$ up to the boundary of D_{k+1}, i.e., project the point x^k on the boundary of D_{k+1}. Assuming that x^k and $\hat{x}$ reasonably well approximate the respective centers of gravity, this leads to an acceleration of the method far beyond the guaranteed rate (5). The desired intersection point $x^b \in \partial D_{k+1}$ is found using Lemma 1, and the point $\alpha x^b + (1-\alpha)x^k$, $0 \le \alpha < 1$, is taken as the next iteration x^{k+1}. The multiplier α serves to shift the point x^b slightly inward the set to guarantee a stable performance of the HR-algorithm at the next iteration.

Dilation: Typically, as the method approaches the optimum, the sets D_k become "skinny" in the direction c. As a result, the HR-algorithm exhibits poor performance, i.e., it sticks inside certain subdomains of D_k. To avoid such effects, it is suggested to dilate the set D_k using certain linear transformation (e.g., see [7]). Namely, having generated N_{hr} HR-points z^i in the set D_{k-1}, we compose the covariance matrix

$$W = \frac{1}{N_{hr}-1}\sum_{i=1}^{N_{hr}}\big(z^i - \hat{z}\big)\big(z^i - \hat{z}\big)^{\mathsf{T}},\ \hat{z} = \frac{1}{N_{hr}}\sum_{i=1}^{N_{hr}} z^i,$$

which reconstructs the shape of D_{k-1} from the available information. The direction vector for the HR-algorithm is then taken in the form $\eta = W^{1/2}\xi$, where $\xi \in \mathbb{R}^n$ is uniformly distributed on the surface of the unit hypersphere. In other words, the directions η are generated uniformly on the ellipsoid which approximates the skinny shape of D_k (which is supposed to be "similar" to that od D_{k-1}); this prevents HR-algorithm of sticking and yields a more accurate estimate of the center of gravity even for highly shrunk domains. As a result, the method attains considerably higher accuracy.

Boundary-biased Hit-and-Run (BBHR): In the original HR-algorithm, the next point z^{i+1} is generated uniformly randomly on the chord $[\overline{z}^i,\ \underline{z}^i]$, i.e., $z^{i+1} = \beta\overline{z}^i + (1-\beta)\underline{z}^i$, where $\beta = \texttt{rand}([0,\ 1])$. In the experiments, we used the biased version of the HR-algorithm, in which the point z^{i+1} is chosen non-randomly, closer the boundary of D_k in the desired direction:

$$z^{i+1} \ = \ \beta\overline{z}^i + (1-\beta)\underline{z}^i, \qquad \beta < 1.$$

The goal is twofold. First, the subsequent averaging gives a point $\hat{x}_{\text{BBHR}}$ with much smaller variance; this makes the method numerically more stable and accurate while using smaller amount of HR-points. Second, the proper choice of β yields a smaller function value at the current step and overall acceleration of the method.

Other HR-related issues: One of the important practical questions relates to the proper choice of the amount N_{hr} of HR-points required to produce an accurate approximation to the uniform distribution on D_k and adequately restore the center of gravity. In spite of the existing pessimistic estimates, in practice we could lean on relatively small sample sizes (clearly, this number grows as the dimension of the problem increases). For example, $N_{hr} = 20$ points were sufficient in low-dimensional SDP problems with $\dim x = 2$, while $\dim x = 50$ required $1,500$ to $2,000$ points. In the implementation we suggest to increase the number N_{hr} dynamically if the geometric convergence rate (5) is not attained at several consecutive iterations.

Another point to mention is that averaging might be performed not over the whole set of N_{hr} points but rather over the "best" half, i.e., those having least values of the objective function. Also, since $D_{k+1} \subset D_k$, economy sampling schemes can be applied which use those of the HR-points generated at the previous iteration, which fall into D_{k+1} (so-called *reuse* techniques).

In all experiments we compared the performance of our method with the results obtained by the `solvesdp` routine in the MATLAB-based SeDuMi Toolbox [10]. This numerically well-balanced procedure implements variants of the interior-point method for solving SDP problems [4] and may be considered as a "standard" to compare with. Below, the output of `solvesdp` for the optimal value of the objective function is denoted by f^* and x^* stands for the associated minimizer. The outputs of our method are denoted by $\tilde{f}$ and $\tilde{x}$. Finally, we denote by $\tilde{\varkappa} = (f^k - \tilde{f})/(f^{k-1} - \tilde{f})$ the actual (observed) convergence rate.

Example 1: To illustrate the basic ideas, we consider a simple SDP problem with $n = 2$, where $c = (0,\ 1)^{\mathsf{T}}$ and the constraints have the form $A(x) = A_0 + x_1A_1 + x_2A_2$ with $A_0 = -I$ and randomly generated

$$A_1 = \begin{pmatrix} 0.6936 & -0.1482 & 0.2310 \\ -0.1482 & 0.0301 & 0.0460 \\ 0.2310 & 0.0460 & -0.0833 \end{pmatrix};$$

$$A_2 = \begin{pmatrix} 0.6749 & -0.0826 & 0.0761 \\ -0.0826 & -0.1297 & 0.0236 \\ 0.0761 & 0.0236 & 0.1653 \end{pmatrix}.$$

The 2D set D_{feas} is nonempty; its boundary is given in Fig. 2 below along with the $N_{hr} = 200$ points generated by the conventional HR-algorithm (left) and its boundary-biased modification with $\beta = 0.5$ (right). It is this modification which was usually used in the experiments. As a practical recipe, it is suggested to use the value $\beta = 0.9$ at the initial steps and then gradually decrease it to $\beta = 0.5$.

For this problem, the `solvesdp` routine returns $f^* = -7.11089093\mathbf{4}08564$ as the minimal value of the objective function.

The behavior of several versions of our method is illustrated in Fig. 3, where the values of $\log_{10}(f^k - \tilde{f})$ are given (the minimum among $\tilde{f}$ and f^* is adopted as $\tilde{f}$).

Curve 1 represents the performance of the simplest version where neither projection nor dilation was used, and the standard HR-algorithm was applied. The obtained function value $\tilde{f} = -7.110\mathbf{8}8654501861$ is accurate to the 5th decimal digit, and the actual convergence rate $\tilde{\varkappa} \approx 0.57$

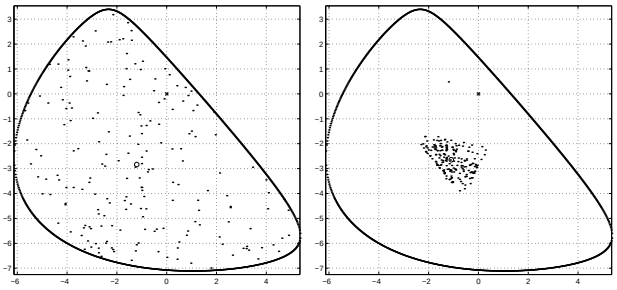

Fig. 2. HR-points.

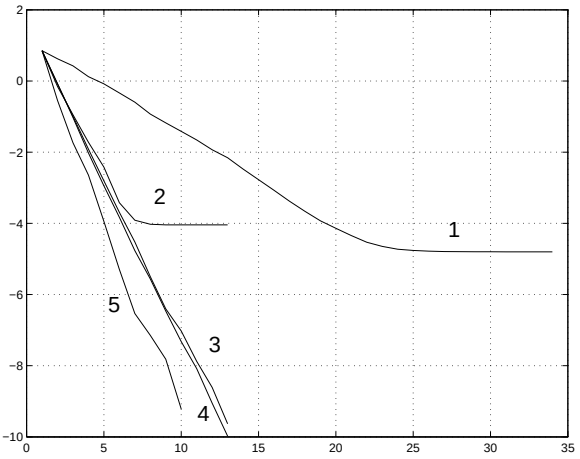

Fig. 3. Performance of various versions of the method.

(over the linear part of the curve) is slightly better than the guaranteed theoretical rate $\varkappa = 0.66$.

Curve 2 represents the version with projecting, which leads to essential acceleration ($\tilde{\varkappa} \approx 0.16$), while the accuracy remains the same.

Application of dilation (curve 3) leads to a considerably higher accuracy, which exceeds the one obtained by `solvesdp`, namely, $\tilde{f} = -7.1108909\mathbf{3}601219$. The convergence rate is approximately the same.

In all the experiments, the standard version of the HR-algorithm with $N_{hr} = 50$ points was used to simulate the uniform distribution. Increasing N_{hr} did not lead to acceleration or increase in accuracy, while decreasing the number of points destabilizes the performance of the method.

Curve 4 represents the modification of the method where the boundary-biased Hit-and-Run with $\beta = 0.5$ was used. This curve is seen to almost coincide with curve 3; however, only $N_{hr} = 20$ points were used. The function value obtained was $\tilde{f} = -7.1108909\mathbf{3}611372$.

Finally, using $\beta = 0.8$ in the boundary-biased HR accelerates the method ($\tilde{\varkappa} \approx 0.1$) at the initial iterations, see curve 5.

Example 2: In the second set of experiments the method was tested on SDP problems having large dimensions of the design vector, $n = \dim x = 300$ and $m = \dim A = 10$. Typically, the method reproduces 7 to 8 exact decimal digits for the function value after 15 iterations (averaging in the HR-algorithm was performed over $N_{hr} = 2,000$ points). The `solvesdp` routine typically exhibits slightly lower accuracy (6 to 7 digits); moreover, sometimes it yields an infeasible point x^*, i.e. $\lambda_{\max}\big(A(x^*)\big) \approx 10^{-7} > 0$.

Example 3: In the third set of experiments, we took $n = 10$ and $m = 100$ as the matrix dimension. Using only $N_{hr} = 200$ HR-points usually leads to 9 exact digits after 15 to 20 steps, and the actual convergence rate is high, $\tilde{\varkappa} \approx 0.3$. In this example with $\dim A \gg \dim x$, the feasible domain in the low-dimensional space is defined by a large number of constraints; this probably explains the fact that we can manage with a small amount of HR-points in such problems.

Example 4: We also tested the method over worst-case geometry SDP problems, where the pessimistic guaranteed rate $\varkappa\ (5)$ is attained. It can be shown that this worst case is realized with simplicial-shaped feasible domains. Accordingly, the matrices A_i were chosen to be diagonal to yield the following feasible domain:

$$D_{feas} \ = \ \{x \in \mathbb{R}^n \colon \ \|x\|_1 \le 1, \ x_n \le 0\}.$$

For $\dim x = 5$, the simplest version of the method (without dilation and projection and with standard HR-algorithm) exposed the rate $\tilde{\varkappa} \approx 0.83$, which is equal to $\varkappa\ (5)$. Respectively, to attain the 10th exact decimal digit, some 110 iterations were needed. Using the accelerating modifications described above we were able to attain the same accuracy after 15 to 18 iterations. The same conclusions apply to the behavior of the method in higher-dimensional problems; for example, with $\dim x = 10$ we have $\dim A = 513$ (for such problems $N_{hr} = 200$ points have been taken to attain accuracy 10^{-10} after 20 iterations).

Interestingly, for feasible domains of such a shape, dilation is useless, since it affects neither the rate of convergence, nor the accuracy, which is also confirmed by the experiments.

Example 5: Experiments have been also conducted with the robust version of the method; they showed reasonable accuracy and rate of convergence for randomly generated coefficient matrices A_i. For example, in the perturbed version of the problem considered in Example 1, the method yields 5 to 6 exact decimal digits after 10 iterations.

The main difference from the non-robust problem is that a nonlinear equation is to be solved at every step of the HR-algorithm; see Lemma 3. As a result, every step of the method requires 5 to 20 times as much cpu time as compared to the non-robust version (depending on the dimensions of the SDP problem).

This difference also has a negative effect on the accuracy of the method. Namely, to solve the nonlinear equation, the standard MATLAB routine `fsolve` was used, which required an initial point as an input parameter. As the method approaches the optimum, the segment $[\underline{\lambda}^{rob}, \ \overline{\lambda}^{rob}]$ becomes considerably smaller than the segment $[\underline{\lambda}, \ \overline{\lambda}]$, and the initial point $\underline{\lambda}$ (or $\overline{\lambda}$) for the `fsolve` routine turns out to be a very poor initial approximation to the solution $\underline{\lambda}^{rob}$ ($\overline{\lambda}^{rob}$

respectively). As a result, the estimated endpoints $\underline{\lambda}^{rob}$, $\overline{\lambda}^{rob}$ may fall out the robustly feasible domain D_{feas}^{rob}.

VI. Concluding Remarks

We proposed a new randomized approach to solving constrained optimization problems of quite a general structure. Preliminary numerical experiments have shown rather stable performance and high accuracy of the method, which often exceeds the one obtained with the known techniques. This makes us believe that a more accurate numerical implementation combined with a more sophisticated coding can lead to a powerful solver which will beat the presently existing methods.

Randomization concept underlying our approach reduces the computational burden while keeping the method exact enough in practical applications. It is therefore believed that it will be useful in high-dimensional problems as well as in situations where the data defining the structure of the problem contain uncertainty.

References

[1] O. N. Granichin and S. L. Molodtsov, *Development of Hybrid Ultrafast Computers and System Programming*. St. Petersburg: Izdatel'stvo St. Petersburg. Univ., 2006. (In Russian.)

[2] R. Tempo, G. Calafiore, and F. Dabbene, *Randomized Algorithms for Analysis and Control of Uncertain Systems*. London: Springer-Verlag, 2005.

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

[4] Yu. Nesterov and A. Nemirovskii, *Interior-point Polynomial Algorithms in Convex Programming*. Philadelphia: SIAM, 1994.

[5] R. L. Smith, "Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions," *Operations Research*, vol. 32, pp. 1296–1308, 1984.

[6] B. T. Polyak and P. S. Shcherbakov, "The D-decomposition technique for solving linear matrix inequalities," *Automat. Remote Control*, vol. 67, no. 11, pp. 159–174, 2006. (Transl. from *Avtomatika i Telemekhanika*, no. 11, pp. 159–174, 2006.)

[7] D. Bertsimas and S. Vempala, "Solving convex programs by random walks," *J. ACM*, vol. 51, no. 4, pp. 540–556, 2004.

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

[9] L. El Ghaoui, F. Oustry, and H. Lebret, "Robust solutions to uncertain semidefinite programs," *SIAM J. Optimiz.*, vol. 9, no. 1, pp. 33–52, 1998.

[10] J. F. Sturm, "Using SeDuMi, a Matlab toolbox for optimization over symmetric cones," *Optimization Methods and Software*, vol. 11–12, pp. 625–653, 1999. Available from `http://fewcal.kub.nl/sturm/software/sedumi.html`