International Journal of Applied Mathematics and Engineering Sciences
Vol. 2 No. 1 (June, 2017)

GENERATING UNIFORM POINTS ON THE BOUNDARY OF BOUNDED SPECTRAHEDRON WITH APPLICATIONS TO RANK CONSTRAINED LINEAR MATRIX INEQUALITY PROBLEM

Shafiu Jibrin

ABSTRACT: This paper presents an algorithm for generating (asymptotically) uniform points on the boundary of a bounded spectrahedron (a closed convex set defined by a set of linear matrix inequality constraints). This is an extension of the *running shake-and-bake* algorithm from linear constraints to linear matrix inequality constraints. The algorithm can be used to visualize boundary shapes of bounded spectrahedra in $\mathbb{R}^2$ and $\mathbb{R}^3$. It can also be used to get approximate solution of any semidefinite programming problem that attains its solution on the boundary. We show an application of the algorithm to the rank constrained linear matrix inequality problem, which is an important problem in engineering and mathematics. Our numerical results indicate the effectiveness of this approach.

KEYWORDS: Semidefinite programming, linear matrix inequalities, asymptotically uniform points, shake- and-bake method, simulation, rank constraints, visualization.

AMS Subject Classification: 15A39, 65C05, 68U05, 90C22, 62P30

1. INTRODUCTION

A semidefinite programming problem is an extension of a linear programming problem where the linear constraints are replaced by linear matrix inequality (LMI) constraints. Semidefinite programming also generalizes quadratically constrained quadratic programming QCQP [17]. Semidefinite programming problems arise in a variety of applications e.g., in engineering, statistics and combinatorial optimization ([1], [6]). Semidefinite Programming has been the subject of intense research since the early 1990's in both theory and applications. A semidefinite programming program (SDP) has the form:

$$SDP: \quad min\ c^T x$$

$$s.t.\ A^{(j)}(x) := A_0^{(j)} + \sum_{n=1}^{n} x_i A_i^{(j)} \succeq 0,\ j = 1, 2, \ldots, q \tag{1.1}$$

where $c, x \in \mathbb{R}^n, A_i^{(j)}$ are $m_j \times m_j$ symmetric matrices. The constraint $A^{(j)}(x) \succeq 0$ is called a *linear matrix inequality* (LMI) and $B \succeq 0$ means that B is positive semidefinite.

We assume that $m_1 \leq m_2 \leq \ldots \leq m_q$.

Consider the feasible region $\mathcal{R}$ defined by the set of linear matrix inequalities (1.1). The region $\mathcal{R}$ is called a *spectrahedron* and is convex. This is a polyhedron when all the LMI constraints are linear. We assume that $\mathcal{R}$ is bounded and full-dimensional.

Hit-and-run algorithms for generating uniform points in the interior of $\mathcal{R}$ when $\mathcal{R}$ is a polyhedron, are described in ([3], [12]) in the context of detecting necessary linear constraints. These were extended to LMI constraints in [10]. It has been shown by R. L. Smith that hit-and-run algorithms do not generate uniform points on the boundary of $\mathcal{R}$ [5]. We illustrate this point in the case of the coordinate directions (CD) hit-and-run algorithm. Consider the following polyhedron in Figure 1, which is a rectangle with length 1 and width 1/2. The regions 1 and 2 partition the polyhedron into two equal parts. We apply CD to the polyhedron.

Consider the boundary segments a, b and c. Let p_{ij} be the probability of hitting boundary segment j along a random coordinate direction from region i of the polyhedron in an iteration of CD. Then, $p_{1a} = p_{2b} = p_{2c} = p_{1c}$ and $p_{2a} = p_{1b} = 0$. So, $p_a = p_{1a} + p_{2a} = p_{1c}$

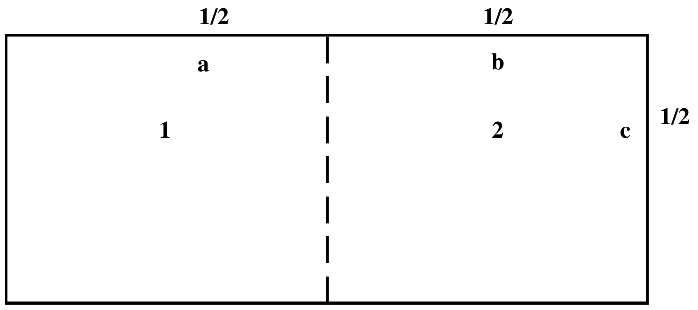

Figure 1. The Boundary Segments *a*, *b*, *c* do not have the Same hit probabilities

and $p_c = p_{1c} + p_{2c} = p_{1c} + p_{1c} = 2p_{1c}$. Hence, $p_a \neq p_c$. This means that the points generated by CD on the boundary of the polyhedron are not asymptotically uniform.

In the special case of linear constraints, the so-called shake-and-bake algorithms for generating asymptotically uniform points on the boundary of $\mathcal{R}$ are given in [4]. The most efficient of them is the *running SB* algorithm. To describe running SB, suppose $\mathcal{R}$ is the polyhedron

$$\mathcal{P} = \{x \in \mathbb{R}^n : a_j^T x \leq b_j,\ j = 1, 2, \ldots, q\}$$

where each $\| a_j \| = 1$. Find a point x_0 on the boundary of $\mathcal{P}$ and determine the boundary point x_1 as follows:

1. Choose a random point u on the (relative) interior of the intersection of the $(n - 1)$-dimensional unit hypersphere centered at the origin and the hyperplane $a_{x_0}^T x = 0$
2. Define s to be the inward unit vector whose projection on $a_{x_0}^T x = 0$ is equal to u
3. Find the boundary hit point y_0 of $\mathcal{P}$ along the ray $\{x_0 + \sigma s : \sigma > 0\}$
4. Set $x_1 = y_0$ with probability $p = 1$

By repeating the above procedure, we generate a sequence of points $\{x_k\}$ on the boundary of $\mathcal{P}$. We state the following result.

LEMMA 1.1. *[4] In the running SB algorithm the points $\{x_k\}$ converges (asymptotically) to uniform points on the boundary of $\mathcal{P}$*

McDonald in 1989 [13] showed that shake-and-bake algorithms can be generalized, in principle, to almost all bounded convex regions. A key element is the ability to compute supporting halfspaces at boundary points. It is also essential that intersection points of a random ray with the boundary can be computed efficiently.

In this paper, we extend running SB to the spectrahedron $\mathcal{R}$. We call the extension *running spectrahedron shake-and-bake algorithm* (running SSB). We give examples to show that running SSB can be used to visualize boundary shapes of bounded spectrahedra in $\mathbb{R}^2$ and $\mathbb{R}^3$. The algorithm can also be used to find approximate solutions of difficult-to-solve semidefinite programming problems which attain their solutions on the boundary. As an example, we study an important application of the algorithm to the rank constrained linear matrix inequality problem, which is an NP-hard problem. Our numerical results indicate the usefulness of this approach. Rank constrained linear matrix inequality problem arises in robust control, output feedback, signal processing, computational geometry and statistics ([8], [9], [15]).

All numerical experiments were done using a Dell OptiPlex GX300 computer with codes written in MATLAB Version 7.1.

2. DETERMINING SUPPORTING HALFSPACES AND LMI BOUNDARY INTERSECTION POINTS

In this section, we show how to determine the supporting halfspace at a boundary point of an LMI constraint. We also show how to find the boundary intersection points of an LMI constraint from both interior and infeasible points.

For simplification, we consider a single LMI here

$$A(x) := A_0 + \sum_{i=1}^{n} x_i A_i \succeq 0$$

where $x \in \mathbb{R}^n$ and A_i are $m \times m$ symmetric matrices. We assume throughout this section that $A(x) \succeq 0$ is strictly feasible. Let w be a point on the boundary of $A(x) \succeq 0$ and that the boundary is *smooth* at w (see Figure 2).

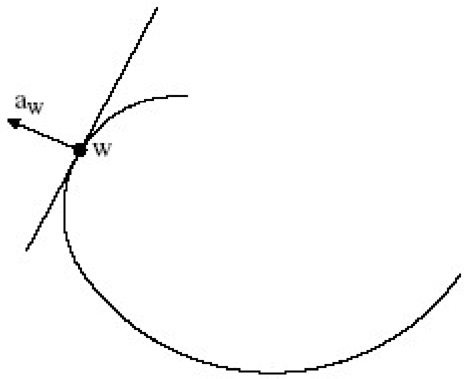

Figure 2. a_w is a normal vector at the boundary point w.

We determine the supporting halfspace at w. Let rank $(A(w)) = r$ and consider an orthogonal decomposition of $A(w)$

$$A(w) = [Q_1Q_2]\text{diag}(0, \dots, 0, \lambda_1, \lambda_2, \dots, \lambda_r)[Q_1Q_2]^T .$$

where $\lambda_1, \dots, \lambda_r$ are the positive eigenvalues of $A(w)$.

LEMMA 2.1. *([2], [11]) An (outward) normal vector a_w at w on the boundary of $A(x) \succeq 0$ is given by:*

$$(a_w)_i = -Q_1^T A_i Q_1 (i = 1, 2, \dots, n).$$

Hence, the supporting halfspace at w is given by

$$a_w^T x \le b_w$$

where $b_w = a_w^T w$. The time complexity for finding the supporting halfspace is $O(m^3 + nm^2 + n^2)$. Next, we show how to determine the boundary intersection points(s) of $A(x) \succeq 0$ from interior and infeasible points. Let x_0 be a point in $\mathcal{R}^n$ and choose a search direction s.

Consider the case when x_0 is an interior point of the LMI (see Figure 3). The boundary intersection point of the LMI along the ray $\{x_0 + \sigma s : \sigma > 0\}$ is defined by $x_0 + \sigma_1 s$, where σ_1 is the solution of the following problem:

$$\begin{array}{ll} \max & \sigma \\ s.t. & A(x_0 + \sigma s) \succeq 0 \end{array}$$

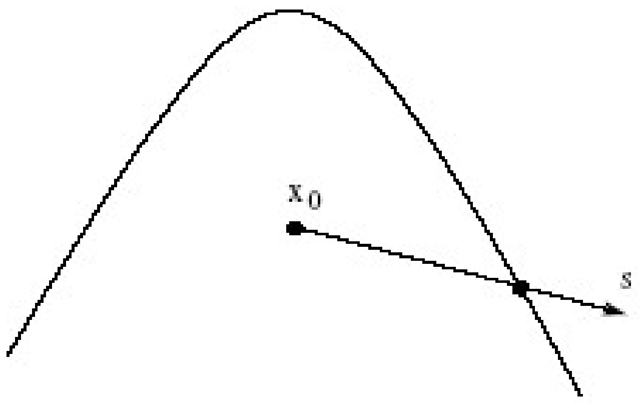

Figure 3. In the figure, *x*0 is an interior point of $\mathcal{R}$

Note that $A(x_0) \succ 0$ (positive definite). Define the symmetric matrix

$$B(x_0, s) := -A(x_0)^{-1/2}\left(\sum_{i=1}^{n} s_i A_i\right)A(x_0)^{-1/2}$$

LEMMA 2.2. *[10] The boundary intersection point of A(x) $\succeq$ 0 along the ray (x_0 + σs : σ > 0} is given by $x_0 + \sigma_1 s$, where*

$$\sigma_1 = 1/\lambda^{+}_{\max}(B(x_0, s))$$

If $A(x_0) \succ 0$ and $\sum_{i=1}^{n} s_i A_i$ is of rank one or two, another method for finding the boundary intersection point is given in [7].

Now, consider the case when x_0 is an infeasible point of the LMI (see Figure 4).

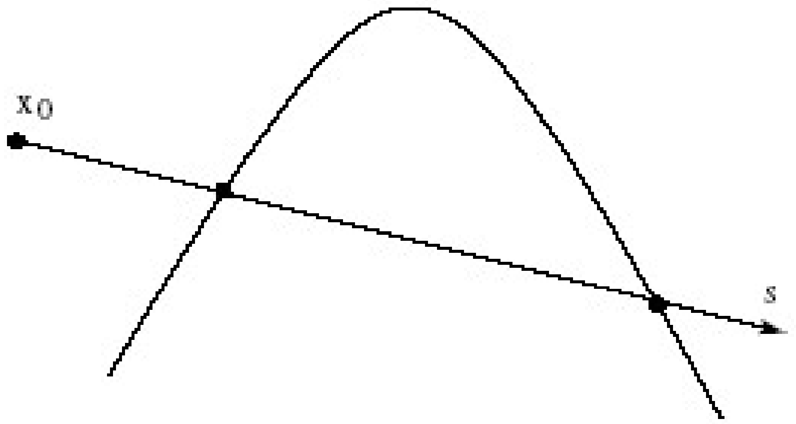

Figure 4. In the figure, x_0 is an infeasible point of $\mathcal{R}$

The boundary intersection points of $A(x) \succeq 0$ along the ray $\{x_0 + \sigma s : \sigma > 0\}$ are defined by $x_0 + \sigma_1 s$ and $x_0 + \sigma_2 s$, where σ_1 and σ_2 are the optimal solutions of the following problems max σ

s.t. $A(x_0 + \sigma s) \succeq 0$

min σ

s.t. $A(x_0 + \sigma s) \succeq 0$

We have the following theorem:

THEOREM 2.1. *Suppose that* $A(x_0)$ *is invertible. The boundary intersection points of* $A(x) \succeq 0$ *along the ray* $\{x_0 + \sigma s\colon \sigma > 0\}$ *are given*

$$\sigma_1 = max\ \{\sigma\ |1/\sigma \text{ is an eigenvalue of } -A(x_0)^{-1}(\sum_{i=1}^{n} s_i A_i),\ \ A(x_0 + \sigma s) \succeq 0\}$$

$$\sigma_2 = min\{\sigma\ |1/\sigma \text{ is an eigenvalue of } -A(x_0)^{-1}(\sum_{i=1}^{n} s_i A_i),\ \ A(x_0 + \sigma s) \succeq 0\}$$

Proof: The determinant of $A(x)$ is zero at boundary intersection point since it is a boundary point. So, if $x_0 + \sigma s$ is intersection point, then

$$|A(x_0 + \sigma s)| = 0 \Leftrightarrow |A(x_0) + \sigma(\sum_{i=1}^{n} s_i A_i)| = 0$$

$$\Leftrightarrow |1/\sigma I + A(x_0)^{-1}(\sum_{i=1}^{n} s_i A_i)| = 0$$

$$\Leftrightarrow 1/\sigma \text{ is an eigenvalue of } -A(x_0)^{-1}(\sum_{i=1}^{n} s_i A_i)$$

The rest of the proof follows from the previous definition of LMI boundary intersection points.

We see that Lemma 2.2 and Theorem 2.1 reduce the problem of finding boundary intersection points to an eigenvalue problem. The time complexity in each case is $O(m^3 + nm^2)$.

3. DETERMINING SPECTRAHEDRON BOUNDARY HIT POINTS

In this section, we show how to determine the boundary hit points of $\mathcal{R}$. We will use the results of the previous section. Recall that $\mathcal{R}$ is the spectrahedron

$$\mathcal{R} = \{x \in \mathbb{R}^n \mid A^{(j)}(x) \succeq 0,\ j = 1, 2, \ldots, q\}.$$

Furthermore, recall that $\mathcal{R}$ is assumed to be bounded and full-dimensional. It follows that for each j, $A^{(j)}(x) \succeq 0$ is strictly feasible. The boundary $\partial\mathcal{R}$ of $\mathcal{R}$ is smooth except possibly on a subset of zero $(n - 1)$-dimensional Lebesgue measure.

Let x_k be a point on the boundary $\partial\mathcal{R}$ of $\mathcal{R}$ such that $\partial\mathcal{R}$ is smooth at x_k. Let l be the index of the LMI binding at x_k and consider the supporting halfspace $a_{x_k}^T w \le b_{x_k}$ at x_k determined by $A^{(l)}(x) \succeq 0$. Without loss of generality, we assume that a_{x_k} is a unit vector. Let s be a random direction such that $a_{x_k}^T s < 0$.

How to efficiently determine the point y_k hitting on $\partial\mathcal{R}$ from x_k along the ray $\{x_k + \sigma s : \sigma \geq 0\}$? We first state the following lemma and its corollary:

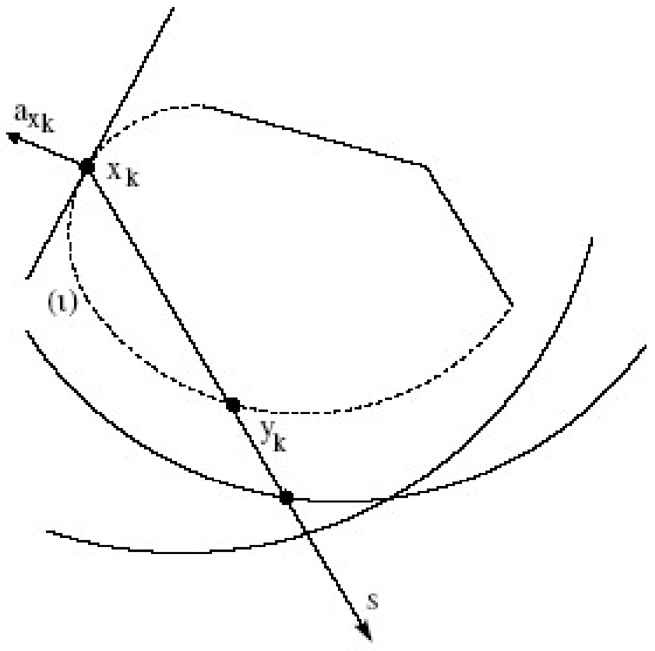

Figure 5. y_k **is the point hitting on** $\partial\mathcal{R}$ **along the ray** $\{x_k + \sigma s : \sigma \geq 0\}$

LEMMA 3.1. *[16] Any real-valued polynomial function on* $\mathbb{R}^n$*, is either identically* 0*, or non-zero* almost everywhere.

COROLLARY 3.1. *For each j,* $A^{(j)}(x)$ *is invertible for almost all* $x \in \mathbb{R}^n$*.*

Proof: By assumption, $A^{(j)}(x) \succeq 0$ is strictly feasible. So, there is an x such that the determinant $|A^{(j)}(x)| \neq 0$. Hence, by Lemma 3.1, the polynomial $|A^{(j)}(x)|$ is nonzero for almost all $x \in \mathbb{R}^n$.

Now, define the spectrahedron

$$\mathcal{R}_\ell = \{x \in \mathbb{R}^n \mid A^{(j)}(x) \succeq 0, j = 1, 2, \ldots, \ell - 1, \ell + 1, \ldots, q\}$$

We can determine the point hitting on $\partial\mathcal{R}_\ell$ from x_k along the ray $\{x_k + \sigma s : \sigma \geq 0\}$ as follows:

1. Find the boundary intersection points (if any) of $A^{(j)}(x) \succeq 0$ from x_k along the ray for $j = 1, 2, \ldots, \ell - 1, \ell + 1, \ldots, q$. Note that x_k is an interior point of each $A^{(j)}(x) \succeq 0$ for $j = 1, 2, \ldots, \ell - 1, \ell + 1, \ldots, q$. So, Lemma 2.2 can be used.
2. Find the boundary intersection point z closest to x_k in above (if any)

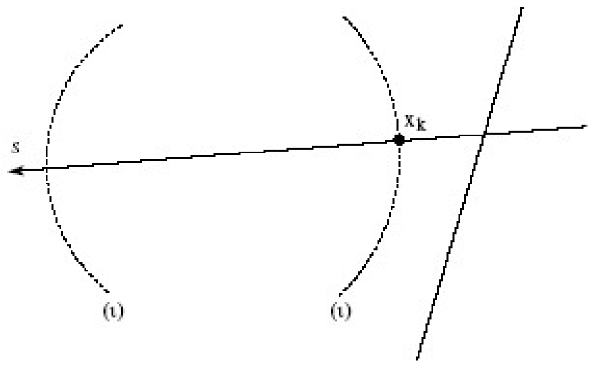

Figure 6. $\mathcal{R}_\ell$ **is unbounded along the ray** $\{x_k + \sigma s : \sigma \geq 0\}$,

There two cases to consider:

Case 1: The ray $\{x_k + \sigma s : \sigma \geq 0\}$ is unbounded in $\mathcal{R}_\ell$. (see Figure 6).

In this case, there are no boundary intersection points on $\mathcal{R}_\ell$ from x_k along the ray. The point y_k hitting on $\partial\mathcal{R}$ is given by the second boundary intersection point of the ℓ^{th} LMI from z along the ray $\{z + \sigma s : \sigma \geq 0\}$, where $z = x_k - s$. See Figure 7. This can be found using Theorem 2.1. Note that $A^{(\ell)}(z) \not\succeq 0$. That is, $A^{(\ell)}(x) \succeq 0$ is infeasible at z. Note that since z is random, by Corollary 3.1, $A^{(\ell)}(z)$ is invertible.

Case 2: The ray $\{x_k + \sigma s : \sigma \geq 0\}$ is bounded in $\mathcal{R}_\ell$.

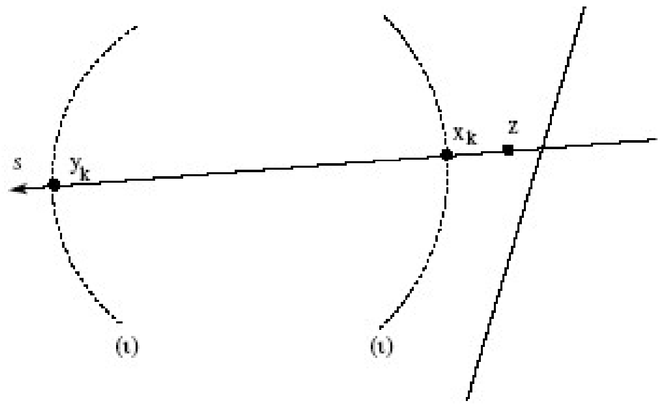

Figure 7. y_k **is the second boundary intersection point of the** ℓ^{th} **LMI from** z **along the ray** $\{z + \sigma s : \sigma \geq 0\}$

First, we determine the point z of $\partial\mathcal{R}_\ell$ from x_k along the ray $\{x_k + \sigma\, s : \sigma \geq 0\}$. Then we consider two subcases:

Case 2.1: $A^{(\ell)}(z) \not\succeq 0$, that is, z is an infeasible point of $A^{(\ell)}(x) \not\succeq 0$ (see Figure 8).

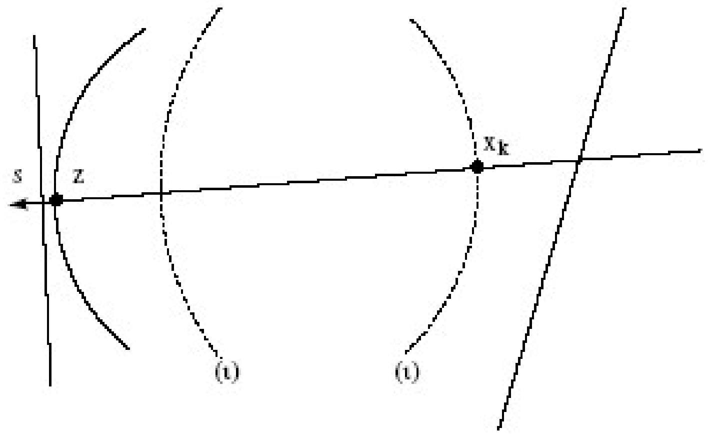

Figure 8. z **is an infeasible point of** $A^{(\ell)}(x) \succeq 0$

The point y_k hitting on $\partial\mathcal{R}$ is given by the first boundary intersection point of the ℓ^{th} LMI from z along the ray $\{z + \sigma(-s) : \sigma \geq 0\}$. See Figure 9. This can be found using Theorem 2.1. Note that z is random, then by Corollary 3.1, $A^{(\ell)}(z)$ is invertible.

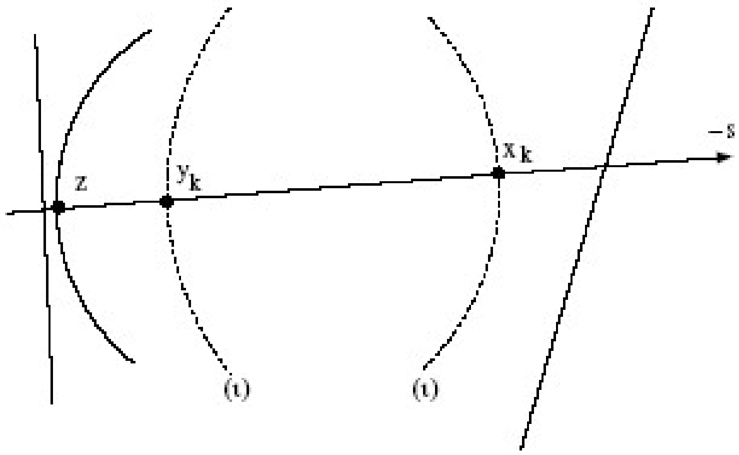

Figure 9. y_k **is the first boundary intersection point of the** ℓ^{th} **LMI from** z

Case 2.2: $A^{(\ell)}(z) \succeq 0$ (see Figure 10).

Here, the point y_k hitting on $\partial\mathcal{R}$ is the point z.

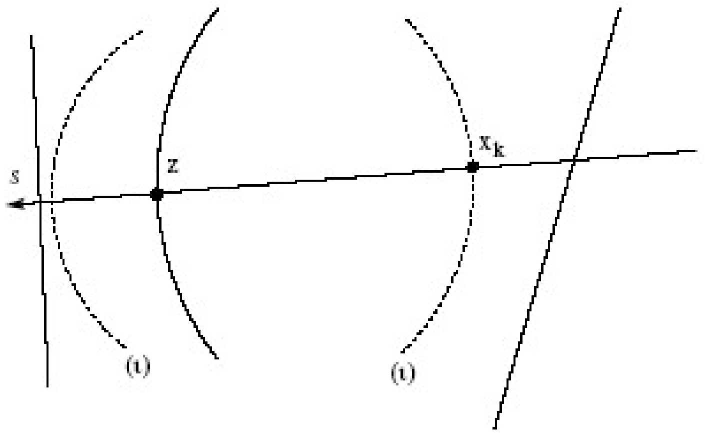

Figure 10. z **lies on the boundary** $\partial\mathcal{R}$

4. EXTENDING RUNNING SB ALGORITHM TO SPECTRAHEDRON: RUNNING SSB

In this section, we use ideas developed in the previous sections to give an extension of the running SB algorithm to linear matrix inequality constraints. We call the extension *running spectrahedron shake-and-bake algorithm* (running SSB).

Consider the spectrahedron

$$\mathcal{R} = \{x \in \mathbb{R}^n \mid A^{(j)}(x) \succeq 0, j = 1, 2, \ldots, q\}$$

Recall that $\mathcal{R}$ is assumed to be full-dimensional and bounded. The boundary $\partial\mathcal{R}$ is smooth except possibly on a subset of zero $(n-1)$-dimensional Lebesgue measure. So, the normal vector and the supporting halfspace can be found at any random point on $\partial\mathcal{R}$. The following describes running SSB. Step 1 and Step 3 use the techniques given in Section 2 and Section 3 (respectively).

Running SSB algorithm

0. Find some point $x_0 \in \partial\mathcal{R}$. Set $k = 0$.
1. Determine the supporting halfspace $a_{x_k}^T x \le b_{x_k}$ at x_k on $\mathcal{R}$ with $\| a_{x_k} \| = 1$.
2. i. Choose a random point u on the (relative) interior of the intersection of the $(n - 1)$-dimensional unit hypersphere centered at the origin and the hyperplane $a_{x_k}^T x = 0$.

   ii. Define s to be the inward unit vector whose projection on the hyperplane $a_{x_k}^T x = 0$ is equal to u.
3. Find the point y_k hitting on $\partial\mathcal{R}$ from x_k along the ray $\{x_k + \sigma s : \sigma \ge 0\}$
4. Set $x_{k+1} = x_k$ and update $k \leftarrow k + 1$. Return to Step 1 and repeat.

The time complexity of Step 1 is $O(m_\ell^3 + nm_\ell^2)$. Step 2 is $O(n)$ [4], and Step 3 is $O(m_\ell^3 + nm_\ell^2 + n^2)$.

THEOREM 4.1. *In the running SSB algorithm* $\{x_k\}$ *converges (asymptotically) to uni*form points on $\partial\mathcal{R}$.

Proof: After $n + 1$ boundary points are generated, consider the polyhedron formed by the supporting halfspaces determined by the generated points. Asymptotically: (1) by Lemma 1.1 the generated points are uniform on the limiting polyhedron (2) the polyhedron converges to the spectrahedron . A more formal proof follows from Theorem 5.3 in [13] applied to a spectrahedron.

We give two examples to show an application of running SSB in estimating the boundary shape of a spectrahedron. We ran 5000 iterations and plot the boundary points generated in each case. Figure 11A shows an example in $\mathbb{R}^2$ with $n = 2$, $m = [3, 1, 2, 2, 2]$, $q = 5$. An example in $\mathbb{R}^3$ with $n = 3$, $m = [2, 4, 5, 3]$, $q = 4$ is given in Figure 11B.

It is a challenge to describe the shape of a spectrahedron even if the bounding (necessary) LMI constraints are known. Running SSB gives us the means to visualize the shape. The more iterations are made, the better is the estimate of the shape.

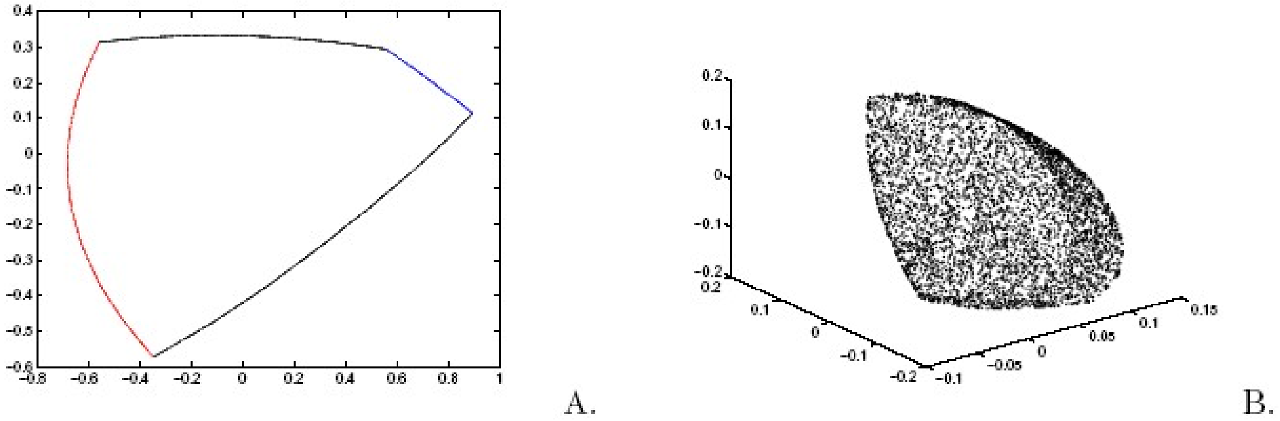

Figure 11. (A) Shape in $\mathbb{R}^2$ using running SSB. (B) Shape in $\mathbb{R}^3$ using running SSB

5. APPLICATIONS TO RANK CONSTRAINED LINEAR MATRIX INEQUALITY PROBLEM

In this section, we show an application of running SSB to finding solutions of rank constrained linear matrix inequality problem.

The rank constrained linear matrix inequality problem (RCLMIP) has a wide range of applications across many disciplines of engineering and computational sciences ([8], [9]). In general, RCLMIP is NP-hard [17]. Algorithms available for RCLMIP are largely heuristic in nature and are not guaranteed to converge to a solution even if one exists [15]. We consider the following special case:

$$A^{(1)}(x) := A_0^{(1)} + \sum_{i=1}^{n} x_i A_i^{(1)} \succeq 0 \tag{5.1}$$

$$A^{(2)}(x) := A_0^{(2)} + \sum_{i=1}^{n} x_i A_i^{(2)} \succeq 0 \tag{5.2}$$

$$rank[A^{(2)}(x)] \leq m_2 - 1 \tag{5.3}$$

where $x \in \mathbb{R}^n$. Also $A_i^{(1)}$ are $m_1 \times m_1$ and $A_i^{(2)}$ are $m_2 \times m_2$ symmetric matrices. We assume that the solution set is either empty or has a nonzero $(n-1)$-dimensional Lebesgue measure.

Note that the solution set of problem RCLMIP (5.1)-(5.3) is precisely where the boundary of $A^{(2)}(x) \succeq 0$ intersects the boundary of the set

$$\mathcal{R} = \{x \in \mathbb{R}^n \mid A^{(j)}(x) \succeq 0, j = 1, 2\}.$$

We assume that $\mathcal{R}$ is bounded and full-dimensional. We say that $A^{(2)}(x) \succeq 0$ is *redundant* in $\mathcal{R}$ if its removal does not change the feasible region $\mathcal{R}$, otherwise it is called *necessary*. We give the following result.

THEOREM 5.1. *Assume that on the boundary of* $\mathcal{R}$, $A^{(1)}(x) \succeq 0$ *and* $A^{(2)}(x) \succeq 0$ *coincide on at most a* $(n-1)$ *dimensional subset of* $\mathcal{R}$ *of* Lebesgue measure *zero. Then RCLMIP (5.1)-(5.3) has a solution if and only if* $A^{(2)}(x) \succeq 0$ *is necessary in* $\mathcal{R}$.

Proof: Suppose RCLMIP (5.1)-(5.3) has a solution. By assumption, the boundary subset of $\mathcal{R}$ where $A^{(1)}(x) \succeq 0$ and $A^{(2)}(x) \succeq 0$ intersect has a nonzero $(n-1)$-dimensional Lebesgue measure. Hence, removal of $A^{(2)}(x) \succeq 0$ will change the feasible region $\mathcal{R}$. So, $A^{(2)}(x) \succeq 0$ is necessary in $\mathcal{R}$. Suppose $A^{(2)}(x) \succeq 0$ is necessary in $\mathcal{R}$. Then the boundary of $A^{(2)}(x) \succeq 0$ intersects the boundary of $\mathcal{R}$. Any point on the intersection is a solution to RCLMIP (5.1)-(5.3).

Theorem 5.1 shows that determining whether or not RCLMIP (5.1)-(5.3) has a solution is equivalent to the semidefinite redundancy problem described in [10] and hence an NP-complete problem. Even if a solution exist, the lack of convexity of the rank constraint (5.3) makes computing a solution of RCLMIP (5.1)-(5.3) a very hard problem ([8], [15]).

Running SSB can be used to find a solution set of RCLMIP (5.1)-(5.3) as follows: Use running SSB to generate boundary points of $\mathcal{R}$. Any generated point that satisfies the feasibility condition $rank[A^{(2)}(x)] \leq m_2 - 1$ is a solution to RCLMIP (5.1)-(5.3). Since the points generated by running SSB are asymptotically uniform, many solutions (if they exist) will be found after a finite number of iterations.

We now give numerical experiments that indicates the effectiveness of running SSB to solving RCLMIP (5.1)-(5.3). In all of the examples used, each LMI $A_0^{(j)} + \sum_{i=1}^{n} x_i A_i^{(j)} \succeq 0$ was generated as follows: $A_i^{(j)}$ is an $m_j \times m_j$ diagonal matrix with each diagonal entry chosen from $U(0, 1)$. Each $A_i^{(j)} (1 \leq i \leq n)$ is a random $m_j \times m_j$ symmetric and sparse matrix with approximately $0.8 * m_j^2$ nonzero entries; each entry is the sum of one or more normally distributed random samples. Note that the origin is an interior point for each problem.

Table I compares running SSB with *LMIRank* [14], a Newton-like method given in [15]. A maximum of 5000 iterations were used in the case of running SSB for each problem. The columns give the example number, the dimension n, the matrices sizes m_1 and m_2. They also give the number of iterations and time (in seconds) taken by running SSB and LMIRank to solve the problems.

Table I. The effectiveness of running SSB

Ex	*n*	m_1	m_2	*running SSB*		*LMIRank*	
				Iter	*Time*	*Iter*	*Time*
1	2	5	4	94	0.4531	5	0.3906
2	2	5	7	5	0.2031	*	*
3	3	6	5	267	1.1094	4	0.2969
4	4	8	7	199	1.2188	*	*
5	4	9	5	152	1.1250	10	0.5156
6	5	8	6	38	0.3906	7	0.4063
7	7	10	8	377	2.9063	8	0.4531
8	7	12	6	96	0.7969	6	0.4063
9	4	10	7	*	*	*	*
10	10	10	10	15	0.3125	6	0.3906
11	10	10	5	250	2.0938	5	0.3438
12	5	6	10	35	0.4531	*	*

It is interesting to observe that while LMIRank converges faster than running SSB, it fails to converge with Example 2, 4, 9 and 12. On the other hand, running SSB found a solution in all the examples except Example 9. It is possible that Example 9 has no solution. We also note that running SSB has lower cost per iteration than LMIRank. We conclude that running SSB can be an effective tool in finding a solution to the rank constrained linear matrix inequality problem.

6. CONCLUSION

We have extended the running SB shake-and-bake algorithm for asymptotically generating uniform boundary points from polyhedron to spectrahedron. We gave examples to show that the algorithm can be used to visualize shapes of bounded spectrahedra in $\mathbb{R}^2$ and $\mathbb{R}^3$. The algorithms can also be used to get approximate solutions of semidefinite programming problems which attain their solution on the boundary. Our numerical experiments indicate the effectiveness of this approach to the rank constrained linear matrix inequality problem.

ACKNOWLEDGEMENTS

The author would like to thank Arnon Boneh for his helpful comments during the preparation of this article.

REFERENCES

[1] F. Alizadeh, "Interior Point Methods in Semidefinite Programming with Applications to Combinatorial Optimization", *SIAM Journal on Optimization*, **5**, no. 1, (1995) 13-51.

[2] F. Alizadeh, J.A. Haeberly and M. Overton, "Complementarity and Nondegeneracy in Semidefinite Programming", *Math. Programming*, **77**, no. 2, ser. B, (1997) 111-128.

[3] H.C.P. Berbee, C.G.E. Boender, A.H.G. Rinnooy Kan, C.L. Scheffer, R.L. Smith and J. Telgen, "Hit-and-run algorithms for the identification of nonredundant linear equalities", *Math. Programming*, **37**, (1987) 184-207.

[4] C.G.E. Boender, R.J. Caron, J.F. McDonald, A.H.G. Rinnooy Kan, H.E. Romeijn, R.L. Smith, J. Telgen and A.C.F. Vorst, *Operations Research*, **39**, (1991) 945-954.

[5] A. Boneh, Private Communication with S. Jibrin, 2005.

[6] S. Boyd and L. El Ghaoui, "Linear Matrix Inequalities in System and Control Theory", *SIAM*, **15**, Philadelphia, PA, 1994.

[7] R.J. Caron and N.I.M Gould, "Finding a Positive Definite Interval for a Parametric Matrix", *Linear Algebra and its Applications*, **52**, (1986) 19-29.

[8] M. Fazel, "*Matrix Rank Minimization with Applications*, *Proceedings of American Conrol Conference*", PhD Thesis, Stanford University, 2002.

[9] K.M. Grigoriadis and E.B. Beran, "Alternating Projection Algorithms for Linear Matrix Inequalities Problems with Rank Constrained", in L. El Ghaoui and S.-I. Niculescu, editors, "*Advances on Linear Matrix Inequality Methods in Control*", SIAM, 1999, 251-267.

[10] S. Jibrin and I.S. Pressman, "Monte Carlo Algorithms for the Detection of Necessary Linear Matrix Inequality Constraints", *International Journal of Nonlinear Sciences and Numerical Simulation*, **2**, (2001) 139-154.

[11] S. Jibrin and D. Stover, "Identifying Redundant Linear Constraints in System of Linear Matrix Inequalities", *Optimization Online*: http:www.optimization-online.orgDB HTML2006/061408.html, (2006).

[12] M.H. Karwan, V. Lofti, J. Telgen, and S. Zionts, "*Redundancy in Mathematical Programming*", Chapter 10, Springer-Verlag, Berlin, 1983.

[13] J.F. McDonald, "*Monte Carlo Procedures for Generating Points Uniformly Distributed Over the Surface of a Bounded Convex Region*", WMR Report 89-03, University of Windsor, 1989.

[14] R. Orsi, "LMIRank: software for rank constrained LMI problems", http://rsise.anu.edu.au/ robert/lmirank/, (2005).

[15] R. Orsi, U. Helmke, and J.B. Moore, "A Newton-Like Method for Solving Rank Constrained Linear Matrix Inequalities", Submitted to: *Automatica*, (2006).

[16] T. Traynor and R.J. Caron, "*The Zero Set of a Polynomial*", WSMR Report 05-02, University of Windsor, 2005.

[17] L. Vandenberghe and S. Boyd, "Semidefinite Programming", *SIAM Review*, **38**, (1996), 49-95.

Shafiu Jibrin
Department of Mathematics and Statistics
Northern Arizona University, Flagstaff
Arizona 86011-5717
(shafiu.jibrin@nau.edu)