An Improved Algorithm for Approximating the Radii of Point Sets *

Yinyu Ye †   Jiawei Zhang ‡

April 3, 2003

Abstract

We consider the problem of computing the outer-radii of point sets. In this problem, we are given integers n, d, k where $k \leq d$, and a set P of n points in R^d. The goal is to compute the *outer k-radius* of P, denoted by $R_k(P)$, which is the minimum, over all $(d-k)$-dimensional flats F, of $\max_{p \in P} d(p, F)$, where $d(p, F)$ is the Euclidean distance between the point p and flat F. Computing the radii of point sets is a fundamental problem in computational convexity with significantly many applications. The problem admits a polynomial time algorithm when the dimension d is constant [9]. Here we are interested in the general case when the dimension d is not fixed and can be as large as n, where the problem becomes NP-hard even for $k = 1$.

It has been known that $R_k(P)$ can be approximated in polynomial time by a factor of $(1 + \varepsilon)$, for any $\varepsilon > 0$, when $d - k$ is a fixed constant [15, 2]. A factor of $O(\sqrt{\log n})$ approximation for $R_1(P)$, the width of the point set P, is implied from the results of Nemirovski [19] and Nesterov [18]. The first approximation algorithm for general k has been proposed by Varadarajan, Venkatesh and Zhang [20]. Their algorithm is based on semidefinite programming relaxation and the Johnson-Lindenstrauss lemma, and it has a performance guarantee of $O(\sqrt{\log n \cdot \log d})$.

In this paper, we show that $R_k(P)$ can be approximated by a factor of $O(\sqrt{\log n})$ for any $1 \leq k \leq d$ and thereby improve the ratio of [20] by a factor of $O(\sqrt{\log d})$ that could be as large as $O(\sqrt{\log n})$. This ratio also matches the previously best known ratio for approximating the special case $R_1(P)$, the width of point set P. Our algorithm is based on semidefinite programming relaxation with a new mixed deterministic and randomized rounding procedure.

1 Introduction

Computing the outer k-radius of a point set is a fundamental problem in computational convexity with applications in global optimization, data mining, statistics, and clustering and has received considerable attention in the computational geometry literature [12, 13, 15]. In this problem,

*Research supported in part by NSF grant DMI-0231600.

†Management Science and Engineering and, by courtesy, Electrical Engineering, Stanford University, Stanford, CA 94305. E-mail: `yinyu-ye@stanford.edu`.

‡Management Science and Engineering, Stanford University, Stanford, CA 94305. E-mail: `jiazhang@stanford.edu`.

we are given integers n, d, k where $k \leq d$, and a set P of n points in R^d. The goal is to compute the *outer k-radius* of P, denoted by $R_k(P)$, which is the minimum, over all $(d-k)$-dimensional flats F, of $\max_{p \in P} d(p, F)$, where $d(p, F)$ is the Euclidean distance between the point p and flat F. A $(d-k)$-flat is simply an affine subspace of R^n of dimension k. (See Section 2 for a more precise definition of $R_k(P)$). Roughly speaking, the outer k-radius $R_k(P)$ measures how well the point set P can be approximated by an affine subspace of dimension $d-k$. A few special cases of $R_k(P)$ which received particular attentions includes: $R_1(P)$, the *width* of P; $R_d(P)$, the radius of the minimum enclosing ball of P; and $R_{d-1}(P)$, the radius of the minimum enclosing cylinder of P.

When the dimension d is a fixed constant, $R_k(P)$ can be computed exactly in polynomial time [9]. It is also known that $R_k(P)$ can be approximated by a factor of $(1+\varepsilon)$, for any $\varepsilon > 0$, in $O(n + (\frac{1}{\varepsilon})^{O(dk)})$ time [1, 14]. In this paper, we are interested in the general scenario when the dimension d is not fixed and can be as large as n,

When the dimension d is a part of the input, the complexity of computing $R_k(P)$ depends on whether $d-k$ is constant or not. It is well-known that the problem is polynomial time solvable when $d-k=0$, i.e., the minimum enclosing ball of a set of points can be computed in polynomial time (Gritzmann and Klee [12]). To the best of our knowledge, whether the problem is NP-hard or not is still open when $d-k=1$. However, Badoiu et al. [2] show that $R_{d-1}(P)$ can be approximated in polynomial time by a factor of $(1+\varepsilon)$, for any $\varepsilon > 0$. Har-Peled and Varadarajan [15, 16] generalize this result and show that $R_k(P)$ can be approximated by a factor of $(1+\varepsilon)$ for any $\varepsilon > 0$ when $d-k$ is constant.

More hardness results are known when $d-k$ becomes large or k becomes small. Bodlaender et al. [4] show that the problem is NP-hard when $k=1$. This is true even for the case $n=2d$ ([12]). Gritzmann and Klee [12] also show that it is NP-hard to compute $R_k(P)$ if $k \leq c \cdot d$, for any fixed $0 < c < 1$. These negative results are further improved by Brieden et al. [5] and Brieden [8], latter of which has shown that it is NP-hard to approximate $R_1(P)$, the width of a point set, to within *any* constant factor. Furthermore, Varadarajan, Venkatesh and Zhang [20] show that there exists some constant $\delta > 0$ such that for any $0 < \varepsilon < 1$, there is no quasi-polynomial time algorithm that approximates $R_k(P)$ within $(\log n)^{\delta}$ for $k \leq d - d^{\varepsilon}$ unless NP $\subseteq$ DTIME $[2^{(\log m)^{O(1)}}]$.

On the positive side, the algorithms of Nesterov [19] and Nemirovski et al. [18] imply that $R_1(P)$, the width of point set P, can be approximated within a factor of $O(\sqrt{\log n})$. Another algorithm for approximating the width of a point set is given by Brieden et al. [6, 7] and their algorithm has a performance guarantee $\sqrt{d/\log d}$ that is measured in the dimension d. The first approximation algorithm for general k have been proposed by Varadarajan et al [20]. Their algorithm is based on semidefinite programming relaxation and the Johnson-Lindenstrauss lemma, and has a performance factor of $O(\sqrt{\log n \cdot \log d})$.

Above mentioned results on computing $R_k(P)$ would give us a projection that the problem is harder when $d-k$ becomes larger or k becomes smaller. However, the result of Varadarajan et al [20] does not confirm this trend, since we have already known that $R_1(P)$ can be approximated by a factor of $O(\sqrt{\log n})$ while, for general k, the ratio proved in [20] is $O(\sqrt{\log n \cdot \log d})$, which is larger than $O(\sqrt{\log n})$. Therefore, they have conjectured that the factor of $O(\sqrt{\log n})$ applies to general k as well.

The main result of the present paper is to show that $R_k(P)$ can indeed be approximated by the factor of $O(\sqrt{\log n})$ for all $1 \leq k \leq d$, and therefore proves their conjecture. Note that the new approximation ratio is reduced by a factor of $O(\sqrt{\log d})$, which could be as large as $O(\sqrt{\log n})$. Our algorithm is based on semidefinite programming relaxation with a mixed deterministic and randomized rounding procedure, in contrast to all other purely randomized rounding procedures used for semidefinite programming approximation.

2 Preliminaries and SDP relaxation

Generally speaking, the problem of computing $R_k(P)$ can be formulated as a quadratic minimization problem. Semidfinite programming (SDP) problems, where the unknowns are represented by positive semidefinite matrices, have recently been developed for approximating such problems; see, for example, Goemans and Williamson [10]. In the case of $k = 1$, computing $R_1(P)$ corresponds to a SDP problem plus an additional requirement that the rank of the unknown matrix equals 1. Removing the rank requirement, the SDP problem becomes a relaxation of the original problem and polynomially solvable for any given accuracy. Once obtaining an optimal solution, say X, of the SDP relaxation, one would like to generate a rank-1 matrix, say $\hat{X} = yy^T$, from X, where y is a column vector and serves as an solution to the original problem. Such rank reduction is called "rounding", and many procedures are proposed and almost all of them are randomized, see, for example, [3].

One particular procedure has been proposed by Nemirovski et al. [18] which can be used for approximating $R_1(P)$. Their procedure is a simple randomized rounding that can described as follows: an optimal solution X of the SDP relaxation, whose rank could be as large as d, can be represented as (e.g., by eigenvector decomposition)

$$X = \lambda_1 v_1 v_1^T + \lambda_2 v_2 v_2^T + \cdots \lambda_d v_d v_d^T.$$

Then one can generate a single vector y by take a random combination of the vectors $v_1, v_2, \cdots, v_d$ where the coefficients of the combination takes values of -1 or 1 uniformly and independently.

When $k \geq 2$, one needs to generate k rank-1 matrices from X, the optimal solution of the corresponding SDP relaxation, such that

$$\hat{X} = \sum_{i=1}^{k} y_i y_i^T$$

where y_is are orthogonal to each other. The method of Varadarajan et al [20] first applies the Johnson-Lindenstrauss randomized dimension reduction technique [17] to reduce the rank of solution X to $k + O(\log n \cdot \log d)$ without losing much in the quality of the solution in terms of the objective value. Then they show that among the $k + O(\log n \cdot \log d)$ vectors, which are orthogonal to each other, k vectors can be randomly chosen as the solution with an approximate ratio $O(\sqrt{\log n \cdot \log d})$.

Our algorithm is based on the same SDP relaxation developed in [20]. However, our rounding procedure is different. Our procedure works as follows. Once obtaining an optimal solution for the SDP relaxation with

$$X = \lambda_1 v_1 v_1^T + \lambda_2 v_2 v_2^T + \cdots \lambda_d v_d v_d^T,$$

we deterministically partition the vectors $v_1, v_2, \cdots, v_d$ into k groups where group j may contain n_j vectors and each group can be seen as a single semidefinite matrix with rank n_j. We can then generate one vector from each group using the randomized rounding procedure similar to that of Nemirovski et al. [18]. The k vectors generated by this rounding procedure will automatically satisfy the condition that any pair of them must be orthogonal to each other, and the quality of these vectors have an approximation ratio no more than $O(\sqrt{\log n})$.

We now present the quadratic program formulation of the k-radii problem and its semidefinite programming relaxation. It will be helpful to first introduce some notations that will be used later. The trace of a given matrix A, denoted by $\mathrm{Tr}(A)$, is the sum of the entries on the main diagonal of A. We use I to denote the identity matrix whose dimension will be clear in the context. The inner product of two vector p and q is denoted by $\langle p, q\rangle$. The 2-norm of a vector x, denoted by $\|x\|$, is defined by $\sqrt{\langle x, x\rangle}$. A positive semidefinite matrix X are represented by $X \succeq 0$. For simplicity, we assume that the P is symmetric in the sense that if $p \in P$ then $-p \in P$. Denote the set $\{-p | p \in P\}$ by $-P$. Let $Q = P \cup -P$. It is clear that $R_k(P) \leq R_k(Q) \leq 2R_k(P)$. Therefore, if we found a good approximation for $R_k(Q)$ then it must also be a good approximation for $R_k(P)$.

Thus, the square of $R_k(P)$ can be defined by the optimal value of the following quadratic minimization problem:

$$
\begin{array}{rll}
R_k(P)^2 := & \text{Minimize} & \alpha \\
& \text{Subject to} & \displaystyle\sum_{i=1}^{k} \langle p, x_i\rangle^2 \leq \alpha, \ \forall p \in P, \\
& & \|x_i\|^2 = 1, \ i = 1, ..., k, \\
& & \langle x_i, x_j\rangle = 0, \ \forall i \neq j.
\end{array}
$$

Assume that $x_1, x_2, \cdots, x_k \in R^d$ is the optimal solution of (1). Then one can easily verify that the matrix $X = x_1 x_1^T + x_2 x_2^T + \cdots x_k x_k^T$ is a feasible solution for the following semidefinite program:

$$
\begin{array}{rll}
\alpha_k^* := & \text{Minimize} & \alpha \\
& \text{Subject to} & \mathrm{Tr}(pp^T X) \ (= p^T X p) \leq \alpha, \ \forall p \in P, \\
& & \mathrm{Tr}(X) = k, \\
& & I - X \succeq 0, \ X \succeq 0.
\end{array}
$$

It follows that $\alpha_k^* \leq R_k(P)^2$. The following Lemma, which is proved in Varadarajan *et al* [20], follows from the above observations. We reproduce the proof below for completeness.

Lemma 1. *There exists an integer $r \geq k$ such that we can compute, in polynomial time, r nonnegative reals $\lambda_1, \lambda_2, \cdots, \lambda_r$ and r orthorgonal unit vectors $v_1, v_2, \cdots, v_r$ such that*

(i). $\sum_{i=1}^{r} \lambda_i = k$.

(ii). $\max_{1 \leq i \leq r} \lambda_i \leq 1$.

(iii). $\sum_{i=1}^{r} \lambda_i \langle p, v_i\rangle^2 \leq R_k(P)^2$, *for any $p \in P$.*

Proof. We solve the semidefinte program (1), and let X^* be an optimal solution of (1). We claim that the rank of X^*, say r, is at least k. This follows from the fact that $\mathrm{Tr}(X^*) = k$ and $I - X^* \succeq 0$. In other words, $\mathrm{Tr}(X^*) = k$ implies that the sum of the eigenvalues of X^* is equal to k, and $I - X^* \succeq 0$ implies that the all eigenvalues are less than or equal to 1. Therefore, X^* has at least k non-zero eigenvalues, which implies that the rank of X^* is at least k. Let $\lambda_1, \lambda_2, \cdots, \lambda_r$ be the r nonnegative eigenvalues and $v_1, v_2, \cdots, v_r$ be the corresponding eigenvectors. Then we have $\sum_{i=1}^r \lambda_i = k$ and $\max_{1 \le i \le r} \lambda_i \le 1$. Furthermore, for any $p \in P$,

$$\sum_{i=1}^r \lambda_i \langle p, v_i \rangle^2 = \sum_{i=1}^r \lambda_i \mathrm{Tr}(pp^T v_i v_i^T) = \mathrm{Tr}(pp^T \sum_{i=1}^r \lambda_i v_i v_i^T) = \mathrm{Tr}(pp^T X^*) \le \alpha_k^* \le R_k(P)^2.$$

□

3 Deterministic First Rounding

In this section, we prove a lemma concerning how to deterministically group the eigenvalues and their eigenvectors. This lemma is simple but play a key role for proving our main result.

Lemma 2. *The index set $\{1, 2, \cdots, r\}$ can be partitioned into k sets $I_1, I_2, \cdots, I_k$ such that*

(i). $\cup_{i=1}^k I_i = \{1, 2, \cdots, r\}$, *and for any $i \neq j$, $I_i \cap I_j = \emptyset$.*

(ii). *For any $i : 1 \le i \le k$, $\sum_{j \in I_i} \lambda_j \ge \frac{1}{2}$.*

Proof. Recall that $\sum_{j=1}^r \lambda_j = 1$ and $\lambda_j \le 1$ for all j. Without loss of generality, we can assume that $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_r$. Our partitioning algorithm is the same as the Longest-Processing-Time heuristic algorithm for parallel machine scheduling problem. The algorithm works as follows:

STEP 1. For $i = 1, 2, \cdots, k$, set $I_i = \emptyset$ and let $L_i = 0$. Let $I = \{1, 2, \cdots, r\}$.
STPE 2. While $I \neq \emptyset$
choose j form I with the smallest index;
choose set i with the samllest value L_i;
Let $I_i := I_i \cup \{j\}$, $L_i := L_i + \lambda_j$ and $I := I - \{j\}$.

It is clear that when the algorithm stops, the sets $I_1, I_2, \cdots, I_k$ satisfy condition (i). Now we prove condition (2) by contradiction. Assume that there exists some t such that

$$\sum_{j \in I_t} \lambda_j < \frac{1}{2}.$$

We now claim that, for all i,

$$\sum_{j \in I_i} \lambda_j \le 1.$$

Otherwise, suppose $\sum_{j \in I_{t'}} \lambda_j > 1$ for some t'. Note that $\lambda_j \le 1$ for every j and thus there are at least two eigenvalues are assigned to $I_{t'}$. Denote the last eigenvalues by $\lambda_{s'}$. It follows that

$$\sum_{j \in I_{t'}} \lambda_j - \lambda_{s'} = \sum_{j \in I_{t'} \setminus \{s'\}} \lambda_j \le \sum_{j \in I_t} \lambda_j$$

since, otherwise, we would have not assigned $\lambda_{s'}$ to $I_{t'}$ in the algorithm. However, since $\sum_{j\in I_t}\lambda_j < \frac{1}{2}$, we must have

$$\sum_{j\in I_{t'}}\lambda_j - \lambda_{s'} = \sum_{j\in I_{t'}\setminus\{s'\}}\lambda_j < \frac{1}{2}.$$

Thus,

$$\lambda_{s'} > \sum_{j\in I_{t'}}\lambda_j - \frac{1}{2} > \frac{1}{2}.$$

This is impossible since $\lambda_{s'}$ is the last eigenvalues assigned to $I_{t'}$, which implies $\lambda_{s'} \le \lambda_j$ for every $j \in I_{t'}$, and we have already proved that there must exist an l such that $s' \neq l \in I_{t'}$ and

$$\lambda_l \le \sum_{j\in I_{t'}\setminus\{s'\}}\lambda_j < \frac{1}{2}.$$

Therefore, $\sum_{j\in I_i}\lambda_j \le 1$ for all i, and in particular $\sum_{j\in I_s}\lambda_j < \frac{1}{2}$. It follows that

$$\sum_{i=1}^{k}\sum_{j\in I_j}\lambda_j < k.$$

However, we know that, by condition (i),

$$\sum_{i=1}^{k}\sum_{j\in I_j}\lambda_j = \sum_{j=1}^{r}\lambda_j = k.$$

This results a contradiction. Therefore, such t does not exists and we have proved condition (ii). □

Notice that the running time of the partitioning algorithm is bounded by $O(r\cdot k)$.

4 Randomized Second Rounding

Assume now that we have found $I_1, I_2, \cdots, I_k$. Then our next randomized rounding procedure works as follows.

STEP 1. Generate a r dimensional random vector ϕ such that each entry of ϕ takes value, independently, -1 or 1 with probability $\frac{1}{2}$ each way.

STEP 2. For $i = 1, 2, \cdots, k$, let

$$x_i = \frac{\sum_{j\in I_i}\phi_j\sqrt{\lambda_j}\cdot v_j}{\sqrt{\sum_{j\in I_i}\lambda_j}}.$$

The following Lemmas show that $x_1, x_2, \cdots, x_k$ form a feasible solution for the original problem. In other words, they are k orthogonal unit vectors.

Lemma 3. *For $i = 1, 2, \cdots, k$, $\|x_i\| = 1$ with probability 1.*

Proof. Recall that $\langle v_l, v_j \rangle = 0$ for any $l \neq j$ and $\|v_j\| = 1$. By definition,

$$
\begin{aligned}
& \|x_i\|^2 = \langle x_i, x_i \rangle \\
= & \left\langle \frac{\sum_{j \in I_i} \phi_j \sqrt{\lambda_j} v_j}{\sqrt{\sum_{j \in I_i} \lambda_j}}, \frac{\sum_{j \in I_i} \phi_j \sqrt{\lambda_j} v_j}{\sqrt{\sum_{j \in I_i} \lambda_j}} \right\rangle \\
= & \frac{1}{\sum_{j \in I_i} \lambda_j} \left\langle \sum_{j \in I_i} \phi_j \sqrt{\lambda_j} v_j, \sum_{j \in I_i} \phi_j \sqrt{\lambda_j} v_j \right\rangle \\
= & \frac{1}{\sum_{j \in I_i} \lambda_j} \sum_{j \in I_i} \langle \phi_j \sqrt{\lambda_j} v_j, \phi_j \sqrt{\lambda_j} v_j \rangle \\
= & \frac{1}{\sum_{j \in I_i} \lambda_j} \sum_{j \in I_i} \|\phi_j \sqrt{\lambda_j} v_j\|^2 \\
= & \frac{1}{\sum_{j \in I_i} \lambda_j} \sum_{j \in I_i} (\phi_j)^2 \lambda_j \|v_j\|^2 \\
= & \frac{1}{\sum_{j \in I_i} \lambda_j} \sum_{j \in I_i} \lambda_j \\
= & 1
\end{aligned}
$$

□

Lemma 4. *If $s \neq t$ then $\langle x_s, x_t \rangle = 0$ with probability 1.*

Proof. The proof is similar as that of Lemma 3.

$$
\begin{aligned}
& \langle x_s, x_t \rangle \\
= & \left\langle \frac{\sum_{j \in I_s} \phi_j \sqrt{\lambda_j} v_j}{\sqrt{\sum_{j \in I_s} \lambda_j}}, \frac{\sum_{j \in I_t} \phi_j \sqrt{\lambda_j} v_j}{\sqrt{\sum_{j \in I_t} \lambda_j}} \right\rangle \\
= & \frac{1}{\sqrt{\sum_{j \in I_s} \lambda_j \cdot \sum_{j \in I_t} \lambda_j}} \left\langle \sum_{j \in I_s} \phi_j \sqrt{\lambda_j} v_j, \sum_{j \in I_t} \phi_j \sqrt{\lambda_j} v_j \right\rangle \\
= & 0.
\end{aligned}
$$

The last equality holds since for any $j \in I_s$ and $l \in I_t$, $\langle v_j, v_l \rangle = 0$. □

Now we establish a bound on the performance of our algorithm. First, let us introduce Bernstein's Theorem (see, e.g., [18]).

Lemma 5. *Let ϕ be a random vector whose entries are independent and either* 1 *or* -1 *with probability* $.5$ *each way. Then, for any vector e and $\beta > 0$,*

$$prob\{\langle \phi, e\rangle^2 > \beta \|e\|^2\} < 2exp(-\frac{\beta}{2}).$$

Let $C_{ip} = \sum_{j\in I_i} \lambda_j \cdot \langle p, v_j\rangle^2$. Then we have

Lemma 6. *For each $i = 1, 2, \cdots, k$ and each $p \in P$, we have*

$$prob\{\langle p, x_i\rangle^2 > 12\ln(n) \cdot C_{ip}\} < \frac{2}{n^3}.$$

Proof. Given i and p, define a $|I_i|$ dimensional vector e such that its entries are $\sqrt{\lambda_j}\langle p, v_j\rangle$, $j \in I_i$, respectively. Furthermore, we define the vector $\phi|_{I_i}$ whose entries are those of ϕ with indices in I_i. First notice that

$$\|e\|^2 = \sum_{j\in I_i} (\sqrt{\lambda_j}\langle p, v_j\rangle)^2 = \sum_{j\in I_i} \lambda_j \cdot \langle p, v_j\rangle^2 = C_{ip}.$$

On the other hand,

$$\begin{aligned}
& \langle p, x_i\rangle^2 \\
= & \left\langle p, \frac{\sum_{j\in I_i} \sqrt{\lambda_j} v_j \phi_j}{\sqrt{\sum_{j\in I_i} \lambda_j}} \right\rangle^2 \\
= & \frac{1}{\sum_{j\in I_i} \lambda_j} \left\langle p, \sum_{j\in I_i} \sqrt{\lambda_j} v_j \phi_j \right\rangle^2 \\
\leq & \ 2\left\langle p, \sum_{j\in I_i} \sqrt{\lambda_j} v_j \phi_j \right\rangle^2 \qquad (\text{since} \sum_{j\in I_i} \lambda_j \geq \frac{1}{2}) \\
= & \ 2\left(\sum_{j\in I_i} \sqrt{\lambda_j} \phi_j \langle p, v_j\rangle \right)^2 \\
= & \ 2\langle \phi|_{I_i}, e\rangle^2
\end{aligned}$$

Thus

$$\text{prob}\{\langle p, x_i\rangle^2 > 12\ln(n) C_{ip}\} \leq \text{prob}\{\langle \phi|_{I_i}, e\rangle^2 > 6\ln(n)\|e\|^2\}.$$

Therefore, the conclusion of the lemma follows by using Lemma 5 and by letting $\beta = 6\ln(n)$. □

Theorem 1. *We can computed in polynomial time, a $(d-k)$-flat such that, with probability at least $1 - \frac{2}{n}$, the distance between any point $p \in P$ and F is at most $\sqrt{12\ln(n)} \cdot R_k(P)$.*

Proof. For given $i = 1, 2, \cdots, k$ and $p \in P$, consider the event

$$B_{ip} = \{\phi | \langle p, x_i \rangle^2 > 12 \ln(n) \cdot C_{ip}\}$$

and $B = \cup_{i,p} B_{ip}$. The probability that the event B happens is bounded by

$$\sum_{i,p} \text{prob}\{\langle p, x_i \rangle^2 > 12 \ln(n) \cdot C_{ip}\} < \frac{2kn}{n^3} \leq \frac{2}{n}.$$

If B does not happen, then for any i and p,

$$\langle p, x_i \rangle^2 \leq 12 \ln(n) \cdot C_{ip}.$$

Therefore, for each $p \in P$,

$$\sum_{i=1}^{k} \langle p, x_i \rangle^2 \leq 12 \ln(n) \sum_{i=1}^{k} \cdot C_{ip} \leq 12 \ln(n) \cdot R_k(P)^2.$$

The last inequality follows from Lemma 1. This completes the proof by taking F as the flat which is orthogonal to the vectors $x_1, x_2, \cdots, x_k$. $\square$

5 Final Remark

Finding efficient rounding methods for semidefinite programming relaxation plays a key role in constructing better approximation algorithms for various hard optimization problems. All of them developed to date are randomized in nature. Therefore, the mixed deterministic and randomized rounding procedure developed in this paper may have its own independent value. We expect its further applications in approximating various computational geometry and space embedding problems.

References

[1] G. Barequet and S. Har-Peled, "Efficiently Approximating the Minimum-Volume Bounding Box of a Point Set in Three Dimensions," *J. Algorithms*, 38:91-109, 2001.

[2] M. Badoiu, S. Har-Peled and P. Indyk, "Approximate Clustering via Core-sets," In *Proc. ACM Symp. Theory of Computing*, 2002.

[3] D. Bertsimas and Y. Ye, "Semidefinite relaxations, multivariate normal distributions, and order statistics," *Handbook of Combinatorial Optimization (Vol. 3)*, D.-Z. Du and P.M. Pardalos (Eds.) pp. 1-19, (1998 Kluwer Academic Publishers).

[4] H.L. Bodlaender, P. Gritzmann, V. Klee and J. Van Leeuwen, "The Computational Complexity of Norm Maximization", *Combinatorica*, 10: 203-225, 1990.

[5] A. Brieden, P. Gritzmann, and V. Klee, "Inapproximability of Some Geometric and Quadratic Optimization Problems," In P.M. Pardalos, editor, *Approximation and Complexity in Numerical Optimization: Continuous and Discrete Problems*, 96-115, Kluwer, 2000.

[6] A. Brieden, P. Gritzmann, R. Kannan, V. Klee, L. Lovasz and M. Simonovits, "Deterministic and Randomized Polynomial-time Approximation of Radii," To appear in *Mathematika.*

[7] A. Brieden, P. Gritzmann, R. Kannan, V. Klee, L. Lovasz and M. Simonovits, "Approximation of Diameters: Randomization Doesn't Help," In *Proc. IEEE Symp. Foundations of Comp. Sci.*, 244-251, 1998.

[8] A. Brieden, "Geometric Optimization Problems Likely Not Contained in APX," *Discrete Comput. Geom.,* 28:201-209, 2002.

[9] U. Faigle, W. Kern, and M. Streng, "Note on the Computaional Complexity of j-Radii of Polytopes in R^n," *Mathematical Programming*, 73:1-5, 1996.

[10] M. X. Goemans and D.P. Williamson, "Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems using Semi-definite Programming," *Journal of the ACM*, 42:1115-1145, 1995.

[11] P. Gritzmann and V. Klee, "Inner and Outer j-Radii of Convex Bodies in Finite-Dimensional Normed Spaces," *Discrete Comput. Geom.*, 7:255-280, 1992.

[12] P. Gritzmann and V. Klee, "Computational Complexity of Inner and Outer j-Radii of Polytopes in Finite-Dimenional Normed Spaces," *Math. Program.*, 59:162-213, 1993.

[13] P. Gritzmann and V. Klee, "On the Complexity of Some basic Problems in Computational Convexity: I. Containment Problems," *Discrete Math.*, 136:129-174, 1994.

[14] S. Har-Peled and K.R. Varadarajan, "Approximate Shape Fitting via Linearization," In *Proc. 42nd Annu. IEEE Sympos. Found. Comput. Sci.*, 66-73, 2001.

[15] S. Har-Peled and K. Varadarajan, "Projective Clustering in High Dimensions Using Core-sets," In *Proc. ACM Symp. Comput. Geom.*, 2002.

[16] S. Har-Peled and K. Varadarajan, "High-Dimensional Shap Fitting in Linear Time," In *Proc. ACM Symp. Comput. Geom.*, 2003.

[17] W.B. Johnson and J. Lindenstrauss, "Extensions of Lipshitz Mapping into Hilbert Space," *Comtemporary Mathematics*, 26:189-206, 1984.

[18] A. Nemirovski, C. Roos and T. Terlaky, "On Maximization of Quadratic Forms Over Intersection of Ellipsoids with Common Center," *Math. Program.*, 86:463-473, 1999.

[19] Yu. Nesterov, "Global Quadratic Optimization via Conic Relaxation," Working Paper, CORE, Catholic University of Louvaine, Belgium, 1998.

[20] K. Varadarajan, S. Venkatesh and J. Zhang, "On Approximating the Radii of Point Sets in High Dimensions," In *Proc. 43rd Annu. IEEE Sympos. Found. Comput. Sci.*, 2002.