# Succinct Data Structures for Approximating Convex Functions with Applications*

Prosenjit Bose[1], Luc Devroye[2], and Pat Morin[1]

[1] School of Computer Science, Carleton University, Ottawa, Canada, K1S 5B6, {jit,morin}@cs.carleton.ca
[2] School of Computer Science, McGill University, Montréal, Canada, H3A 2K6, luc@cs.mcgill.ca

**Abstract.** We study data structures for providing $\epsilon$-approximations of convex functions whose slopes are bounded. Since the queries are efficient in these structures requiring only $O(\log(1/\varepsilon)+\log\log n)$ time, we explore different applications of such data structures to efficiently solve problems in clustering and facility location. Our data structures are succinct using only $O((1/\varepsilon)\log^2(n))$ bits of storage. We show that this is optimal by providing a matching lower bound showing that any data structure providing such an $\epsilon$-approximation requires at least $\Omega((1/\varepsilon)\log^2(n))$ bits of storage.

## 1 Introduction

We consider the problem of approximating convex functions of one variable whose slopes are bounded. We say that a non-negative number $y$ is an $\varepsilon$-approximation to a non-negative number $x$ if $(1-\varepsilon)x \le y \le x$ [1]. We say that a function $g$ is an $\varepsilon$-approximation to a function $f$ if $g(x)$ is an $\varepsilon$-approximation to $f(x)$ for all $x$ in the domain of $f$.

Let $f : \mathbb{R} \to \mathbb{R}^+$ be a convex function that is non-negative everywhere. In this paper we show that, if the absolute value of the slope of $f$ is bounded above by $n$, then there exists a piecewise-linear function $g$ that $\varepsilon$ approximates $f$ at all points $x$ except where the slope of $f$ is small (less than 1) and that consists of $O(\log_E n)$ pieces, where $E = 1/(1-\varepsilon)$. The function $g$ can be computed in $O(K\log_E n)$ time, where $K$ is the time it takes to evaluate expressions of the form $\sup\{x : f'(x) \le t\}$ and $f'$ is the first derivative of $f$. Once we have computed the function $g$, we can store the pieces of $g$ in an array sorted by $x$ values so that we can evaluate $g(x)$ for any query value $x$ in $O(\log\log_E n)$ time. Since we are interested in the joint complexity as a function of $\varepsilon < 1/2$ and $n \ge 10$, it is worth noting that $log_E n = \Theta((1/\varepsilon)\log n)$ and thus that $\log\log_E n = \Theta(\log(1/\varepsilon)+\log\log n)$.

As an application of these results, we consider functions defined by sums of Euclidean distances in $d$ dimensions and show that that they can be approximated using the above results. To achieve this, we use a random rotation

* This research was partly supported by NSERC.

[1] This definition is a bit more one-sided that the usual definition, which allows any $y$ such that $|x-y| \le \varepsilon x$.

J. Akiyama and M. Kano (Eds.): JCDCG 2002, LNCS 2866, pp. 97–107, 2003.

technique similar to the method of random projections [6]. We show that the sum of Euclidean distances from a point to a set of $n$ points can be closely approximated by many sums of Manhattan distances from the point to the set. This technique is very simple and of independent interest.

The remainder of the paper is organized as follows. Section 2 presents our result on approximating convex functions using few linear pieces. Section 3 discusses how these results can be interpreted in terms of data structures for approximating convex functions. Section 4 gives lower bounds on the space complexity of approximating convex function. Section 5 describes applications of this work to facility location and clustering problems. Finally, Section 6 summarizes and concludes the paper.

## 2 Approximating Convex Functions

Let $h(x) = c + |nx|$, for some $c, n \geq 0$. Then, it is clear that the function $g$ such that $g(x) = c + (1 - \varepsilon)|nx|$ is an $\varepsilon$-approximation of $h$. Furthermore, $g$ is an $\varepsilon$-approximation for any function $h_2$ such that $g(x) \leq h_2(x) \leq h(x)$ for all $x \in \mathbb{R}$. (see Fig. 1). This trivial observation is the basis of our data structure for approximating convex functions.

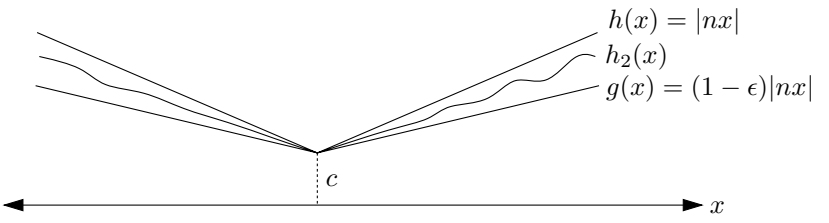


**Fig. 1.** The function $g$ is an approximation of $h$ and of $h_2$.

Let $f$ be a non-negative convex function and let $f'$ be the first derivative of $f$. Assume that $f'(x)$ is defined for all but a finite number of values of $x$ and that $|f'(x)| \leq n$ for all $x$ in the domain of $f'$. For convenience, we define the *right derivative* $f^*(x)$ as follows: If $f'(x)$ is defined, then $f^*(x) = f'(x)$. Otherwise, $f^*(x) = \lim_{\delta \to 0+} f'(x + \delta)$.

Let $a$ be the largest value at which the slope of $f$ is at most $-(1 - \varepsilon)n$, i.e.,

$$a = \max\{x : f^*(x) \leq -(1 - \varepsilon)n\} \ .$$

(Here, and throughout, we use the convention that $\max \emptyset = -\infty$ and $\min \emptyset = \infty$.) Similarly, let $b = \min\{x : f^*(x) \geq (1 - \varepsilon)n\}$. Then, from the above discussion, it is clear that the function

$$g(x) = \begin{cases} f(a) + (1 - \varepsilon)(x - a)n & \text{if } x \leq a \\ f(b) + (1 - \varepsilon)(b - x)n & \text{if } x \geq b \\ f(x) & \text{otherwise} \end{cases} \tag{1}$$

is an $\varepsilon$-approximation of $f$ (see Fig. 2).

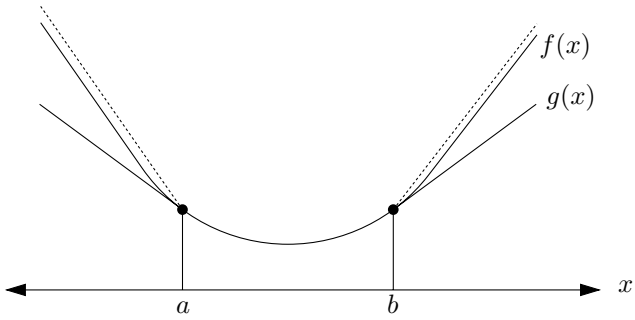


**Fig. 2.** The function $g$ is a $(1-\varepsilon)$ approximation of $f$.

Equation (1) tells us that we can approximate $f$ by using two linear pieces and then recursively approximating $f$ in the range $(a, b)$. However, in the range $(a, b)$, $f^*$ is in the range $(-(1-\varepsilon)n, (1-\varepsilon)n)$. Therefore, if we recurse $\lceil \log_E n \rceil$ times, we obtain a function $g$ with $O(\log_E n) = O((1/\varepsilon) \log n)$ linear pieces that approximates $f$ at all points except possibly where $f^*$ is less than one.

**Theorem 1** *Let $f$ and $f^*$ be defined as above. Then there exists a piecewise-linear function $g$ with $O(\log_E n)$ pieces that is an $\varepsilon$-approximation to $f$ at all values except where $|f^*(x)| \leq 1$.*

## 3 Data Structures

In this section, we consider the consequences of Theorem 1 in terms of data structures for approximating convex functions. By storing the pieces of $g$ in an array sorted by $x$ values, we obtain the following.

**Theorem 2** *Let $f$ and $f^*$ be defined as in Section 2. Then there exists a data structure of size $O((1/\varepsilon) \log n)$ that can compute an $\varepsilon$-approximation to $f(x)$ in $O(\log(1/\varepsilon) + \log \log n)$ time for any query value $x$ where $|f^*(x)| \geq 1$.*

Next, we consider a more dynamic model, in which the function $f$ is updated over time. In particular, we consider the following operations that are applied to the initial function $f(x) = 0$, for all $x \in \mathbb{R}$.

1. QUERY($x$): Return an $\varepsilon$-approximation to $f(x)$.
2. INSERT($a$): Increase the slope of $f$ by 1 in the range $(a, \infty)$, i.e., set $f(x) \leftarrow f(x) + x - a$ for all $x \in [a, \infty)$.
3. DELETE($x$): Decrease the slope of $f$ by 1 in the range $(x, \infty)$. In order to maintain convexity, the number of calls to DELETE($x$) may not exceed the number of calls to INSERT($x$) for any value of $x$.

Note that a sequence of INSERT and DELETE operations can only produce a monotonically increasing function $f$ whose slopes are all integers. This is done

to simplify the exposition of the data structure. If an application requires that $f$ be allowed to decrease and increase then two data structures can be used and their results summed.

The function $f$ has some number $m$ of breakpoints, where the slope of $f$ changes. We store these breakpoints in a balanced search tree $T$, sorted by $x$-coordinate. With each breakpoint $x$, we also maintain the value $\Delta(x)$ by which the slope of $f$ increases at $x$. In addition, we link the nodes of $T$ in a doubly-linked list, so that the immediate successor and predecessor of a node can be found in constant time. It is clear that $T$ can be maintained in $O(\log n)$ time per operation using any balanced search tree data structure.

In addition to the search tree $T$, we also maintain an array $A$ of size $O((1/\varepsilon)\log n)$ that contains the piecewise linear approximation of $f$. The $i$th element in this array contains the value $x_i$ such that $x_i = \min\{x : f^*(x) \geq E^i\}$, a pointer to the node in $T$ that contains $x_i$, and the values of $f(x_i)$ and $f^*(x_i)$, i.e., the value of $f$ at $x_i$ and slope of $f$ at $x_i$. To update this array during an INSERT or DELETE operation, we first update the values of $f(x_i)$ and $f^*(x_i)$ for each $i$. Since there are only $O((1/\varepsilon)\log n)$ array entries, this can be done in $O((1/\varepsilon)\log n)$ time.

Next, we go through the array again and check which values of $x_i$ need to be changed (recall that $x_i = \min\{x : f^*(x) \geq E^i\}$). Note that, since INSERT or DELETE can only change the value of $f^*(x)$ by 1, if the value of $x_i$ changes then it changes only to its successor or predecessor in $T$. Since the nodes of $T$ are linked in a doubly-linked list, and we store the values of $f(x_i)$ and $f^*(x_i)$ we can detect this and update the value of $x_i$, $f(x_i)$ and $f^*(x_i)$ in constant time. Therefore, over all array entries, this takes $O((1/\varepsilon)\log n)$ time.

To evaluate an approximation to $f(x)$, we do a binary search on $A$ to find the index $i$ such that $[x_i, x_{i+1})$ contains $x$ and then output $f(x_i) + (x - x_i)f^*(x_i)$. By the choice of $x_i$, this is a $\varepsilon$-approximation to $f(x)$. We have just proven the following:

**Theorem 3** *There exists a data structure of size $O(n)$ that supports the operations* INSERT, DELETE *in $O((1/\varepsilon)\log n)$ time and* QUERY *in $O(\log(1/\varepsilon) + \log\log n)$ time, where $n$ is the maximum slope of the function $f$ being maintained.*

## 4 A Lower Bound on Storage

In this section we prove an $\Omega((1/\varepsilon)\log^2 n)$ lower bound on the number of bits required by any data structure that provides an $\varepsilon$-approximation for convex functions. The idea behind our proof is to make $m = \Theta((1/\varepsilon)\log n)$ choices from a set of $n$ elements. We then encode these choices in the form of a convex function $f$ whose slopes are in $[0, n]$. We then show that given a function $g$ that is an $\varepsilon$-approximation to $f$ we can recover the $m = \Theta((1/\varepsilon)\log n)$ choices. Therefore, any data structure that can store an $\varepsilon$-approximation to convex functions whose slopes lie in $[0, n]$ must be able to encode $\binom{n}{m}$ different possibilities and must therefore store $\Omega((1/\varepsilon)\log^2 n)$ bits in the worst case.

Let $x_1, \ldots, x_n$ be an increasing sequence where $x_1 = 0$ and each $x_i$, $2 \leq i \leq n$ satisfies

$$x_{i-1} + (1-\varepsilon)\left(\frac{x_i - x_{i-1}}{1-2\varepsilon}\right) > x_{i-1} + \frac{x_i - x_{i-1}}{1-2\varepsilon} \quad , \tag{2}$$

which is always possible since $(1-\epsilon)/(1-2\varepsilon) > 1$.

Let $p_1, \ldots, p_m$ be any increasing sequence of $m = \lfloor \log_{E'} n \rfloor$ integers in the range $[1, n]$, where $E' = 1/(1-2\varepsilon)$. We construct the function $f$ as follows:

1. For $x \in [-\infty, 0)$, $f(x) = 0$.
2. For $x \in (x_{p_i}, x_{p_{i+1}})$, $f(x)$ has slope $1/(1-2\varepsilon)^i$.
3. For $x > p_m$, $f(x)$ has slope $n$.

The following lemma, illustrated in Fig. 3 allows us to decode the values of $p_1, \ldots, p_m$ given an $\varepsilon$-approximation to $f$.

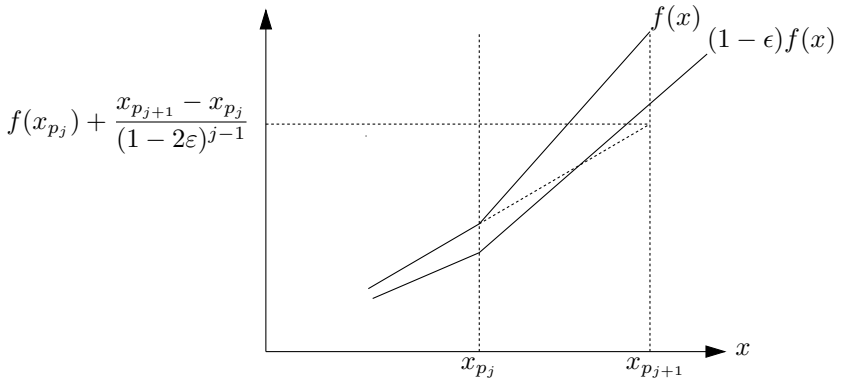


**Fig. 3.** An illustration of Lemma 1.

**Lemma 1** *For the function $f$ defined above and for any $i$ such that $i = p_j$ for some $1 \leq j \leq m$,*

$$(1-\varepsilon) f(x_{p_{j+1}}) > f(x_{p_j}) + \frac{x_{p_{j+1}} - x_{p_j}}{(1-2\varepsilon)^{j-1}} \quad .$$

*Proof.* The lemma follows (with some algebra) from (2).

Suppose that $g$ is an $\varepsilon$-approximation to $f$, i.e, for all $x \in \mathbb{R}$, $g(x)$ satisfies $(1-\varepsilon)f(x) \leq g(x) \leq f(x)$. Then Lemma 1 can be used to recover the values of $p_1, \ldots, p_m$ from $g$. Suppose, that we have already recovered $p_1, \ldots, p_j$ and that we now wish to recover $p_{j+1}$. Note that, since we have recovered $p_1, \ldots, p_j$ we can compute the exact value of $f(x_{p_j})$. We then evaluate $g(x_{p_j} + 1)$, $g(x_{p_j} + 2)$, and so on until encountering a value $k$ such that

$$g(x_{p_j} + k) > f(x_{p_j}) + \frac{x_{p_j} + k - x_{p_j}}{(1-2\varepsilon)^j}$$

Lemma 1 then guarantees that $p_{j+1} = p_j + k - 1$. In this way, we can reconstruct the entire function $f$ and recover the values of $p_1, \ldots, p_m$.

Although in the above discussion the slopes used in the construction of $f$ are not always integral it is clear that carefully rounding values appropriately would yield the same results using only integer valued slopes. Since we can encode $\binom{n}{m}$ different choices of $p_1, \ldots, p_m$ in this manner and $\log \binom{n}{m} = \Omega((1/\varepsilon) \log^2 n)$, we conclude the following:

**Theorem 4** *Any data structure that can represent an $\varepsilon$-approximation to any convex function whose slopes are integers in the range $[0, n]$ must use $\Omega((1/\varepsilon) \log^2 n)$ bits of storage in the worst case.*

**Remark 1** Some readers may complain that the function used in our lower bound construction uses linear pieces whose lengths are exponential in $n$. However, one should take into account that the endpoints of these pieces have $x$-coordinates that are integral powers of 2 and they can therefore be encoded in $O(\log n)$ bits each using, e.g., a floating point representation.

**Remark 2** Another easy consequence of Lemma 1 is that any piecewise linear function that $\varepsilon$-approximates $f$ has $\Omega((1/\varepsilon) \log n)$ pieces.

## 5 Applications

Next, we consider applications of our approximation technique for convex functions to the problem of approximating sums of distances in $d$ dimensions. Let $S$ be a set of $n$ points in $d$ dimensions. The *Fermat-Weber weight* of a point $q \in \mathbb{R}^d$ is

$$\mathrm{FW}(p) = \sum_{p \in S} \|pq\| \ ,$$

where $\|pq\|$ denotes the distance between points $p$ and $q$. Of course, different definitions of distance (e.g., Euclidean distance, Manhattan distance) yield different Fermat-Weber weights.

### 5.1 The 1-Dimensional Case

One setting in which distance is certainly well defined is in one dimension. In this case,

$$\|pq\| = |p - q| \ ,$$

so the Fermat-Weber weight of $x$ is given by

$$\mathrm{FW}(x) = f(x) = \sum_{y \in S} |x - y| \ .$$

Note that the function $f$ is convex (it is the sum of $n$ convex functions) and has slopes bounded below by $-n$ and above by $n$, so it can be approximated using

the techniques Section 3. Furthermore, adding or removing a point $p$ to/from $S$ decreases the slope of $f$ by 1 in the range $(-\infty, p)$ and increases the slope of $f$ by 1 in the range $(p, \infty)$, so the dynamic data structure of the previous section can be used to maintain an $\varepsilon$-approximation of $f$ in $O(\log_E n) = O((1/\varepsilon)\log n)$ time per update.

Given the set $S$, constructing the $\varepsilon$-approximation for $f$ can be done in $O(n/\varepsilon)$ time by a fairly straightforward algorithm. Using a linear-time selection algorithm, one finds the elements of $S$ with ranks $\lfloor \varepsilon n/2 \rfloor$ and $\lceil (1-\varepsilon/2)n \rceil$. These are the values $a$ and $b$ in (1). Once this is done, the remaining problem has size $(1-\varepsilon)n$ and is solved recursively. Although some care is required to compute the values $f(a)$ and $f(b)$ at each stage, the deatails are not difficult and are left to the interested reader.

**Remark 3** A general result of Agarwal and Har-Peled [1] implies that the Fermat-Weber weight of points in one dimension can actually be $\varepsilon$-approximated by a piecewise-linear function with $O(1/\varepsilon)$ pieces. However, it is not clear how easily this approach can be made dynamic to handle insertion and deletions of points.

### 5.2 The Manhattan Case

The Manhattan distance between two points $p$ and $q$ in $\mathbb{R}^d$ is

$$\|pq\|_1 = \sum_{i=1}^{d} |p_i - q_i| \ ,$$

where $p_i$ denotes the $i$th coordinate of point $p$. We simply observe that Manhattan distance is the sum of $d$ one-dimensional distances, so the Fermat-Weber weight under the Manhattan distance can be approximated using $d$ one-dimensional data structures.

### 5.3 The Euclidean Case

The Euclidean distance between two points $p$ and $q$ in $\mathbb{R}^d$ is

$$\|pq\|_2 = \left( \sum_{i=1}^{d} (p_i - q_i)^2 \right)^{1/2} \ .$$

A general technique used to approximate Euclidean distance is to use a polyhedral distance function, in which the unit sphere is replaced with a polyhedron that closely resembles a sphere. For example, the Manhattan distance function is a polyhedral distance function in which the unit sphere is replaced with a unit hypercube. Although this technique works well when $d$ is small, such metrics generally require a polyhedron with a number of vertices that is exponential in $d$, making them poorly suited for high dimensional applications.

Another technique, that works well when $d$ is very large (greater than $\log n$), and for most distance functions, is that of random projections [6]. Here, a random $O(\log n)$-flat is chosen and the points of $S$ are projected orthogonally onto this flat. With high probability, all interpoint distances are faithfully preserved after the projection, so the problem is reduced to one in which the dimension of the point set is $O(\log n)$. The difficulty with this approach, when using Euclidean distances, is that sums of Euclidean distances are difficult to deal with even when $d = 2$ [2], thus the reduction in dimension does not help significantly.

Here we present a third approach that combines both of these techniques and adds two new twists: (1) we obtain a polyhedral metric as the sum of several Manhattan metrics and (2) our polyhedron is random. The first twist allows us to apply approximate data structures for one-dimensional convex functions while the second allows us to achieve approximation guarantees using an a number of vertices that increases only linearly with $d$.

Let $f(p)$ denote the Fermat-Weber weight of $p$ under the Euclidean distance function. Choose $k$ independent random orientations of the coordinate axes. Let $f_i(p)$ denote the Fermat-Weber weight of $p$ under the Manhattan distance function after rotating the axes according to the $i$th random orientation. Then $f_i(p)$ may take on any value in the range $[f(p), \sqrt{d}f(p)]$. In particular, $f_i(p)$ has an expected value

$$\mathbf{E}[f_i(p)] = c_d f(p) \ ,$$

where $c_d$ is a constant, dependent only on $d$, whose value is derived in Appendix A. Consider the function

$$g(p) = \frac{1}{kc_d} \times \sum_{i=1}^{k} f_i(p)$$

that approximates the Fermat-Weber weight under Euclidean distance.

**Lemma 2** $\Pr\{|g(p) - f(p)| \geq \varepsilon f(p)\} = \exp(-\Omega(\varepsilon^2 k))$

*Proof.* The value of $g(p)$ is a random variable whose expected value is $f(p)$ and it is the sum of $k$ independent random variables, all of which are in the range $[f(p), \sqrt{d}f(p)]$. Applying Hoeffding's inequality [4] immediately yields the desired result.

In summary, $g(p)$ is an $\varepsilon$-approximation of $f(p)$ with probability $1-e^{-\Omega(\varepsilon^2 k)}$. Furthermore, $g(p)$ is the sum of $k$ Fermat-Weber weights of points under the Manhattan distance function. Each of these Manhattan distance functions is itself a sum of $d$ Fermat-Weber weights in 1 dimension. These 1-dimensional Fermat-Weber weights can be approximated using the results of Section 3 or the results of Agarwal and Har-Peled [1].

### 5.4 Clustering and Facility Location

Bose *et al* [3] describe data structures for approximating sums of distances. They show how to build a data structure in $O(n \log n)$ time that can $(1 - \varepsilon)$-approximate the Fermat-Weber weight of any point in $O(\log n)$ time. However, the constants in their algorithms depend exponentially on the dimension $d$.

The same authors also give applications of this data structure to a number of facility-location and clustering problems, including evaluation of the Medoid and AverageDistance clustering measures, the Fermat-Weber problem, the constrained Fermat-Weber problem, and the constrained obnoxious facility-location problem. All of these applications also work with the data structure of Section 3, many with improved running times. A summary of these results is given in Table 1.

**Table 1.** Applications of the data structure for evaluating the Fermat-Weber weights of points under the Euclidean distance function.

| Problem | Exact solution | Previous $\varepsilon$-approx. | Ref. | New $\varepsilon$-approx. |
|---|---|---|---|---|
| Average distance | $O(n^2)$ | $O(n)^a$ $O(n \log n)$ | [5,3] | $O(n \log \log n)$ |
| Medoid (1-Median) | $O(n^2)$ | $O(n)^a$ $O(n \log n)$ | [5,3] | $O(n \log \log n)$ |
| Discrete Fermat-Weber | $O(n^2)$ | $O(n)^a$ $O(n \log n)$ | [5,3] | $O(n \log \log n)$ |
| Fermat-Weber | – | $O(n)^b$ $O(n \log n)$ | [5,3] | $O(n)$ |
| Constrained Fermat-Weber | $O(n^2)$ | $O(n)^b$ $O(n \log n)$ | [5,3] | $O(n)$ |
| Constrained OFL | $O(n^2)$ | $O(n)^a$ $O(n \log n)$ | [5,7,3] | $O(n \log \log n)$ |

[a] Refers to a randomized algorithm that outputs a $(1-\epsilon)$-approximation with constant probability.

[b] Refers to a randomized algorithm that outputs a $(1-\epsilon)$-approximation with high probability, i.e., with probability $1 - n^{-c}$, for some $c > 0$.

## 6 Summary and Conclusions

We have given static and dynamic data structures for approximating convex functions of one variable whose slopes are bounded. These data structures have applications to problems involving sums of distances in $d$ dimensions under both the Manhattan and Euclidean distance functions. In developing these applications we have arrived at a technique of independent interest, namely that of approximating Euclidean distance as the sum of several Manhattan distances under several different orientations of the coordinate system.

## References

1. P. K. Agarwal and S. Har-Peled. Maintaining the approximate extent measures of moving points. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 148–157, 2001.
2. C. Bajaj. The algebraic degree of geometric optimization problems. *Discrete & Computational Geometry*, 3:177–191, 1988.
3. P. Bose, A. Maheshwari, and P. Morin. Fast approximations for sums of distances, clustering and the Fermat-Weber problem. *Computational Geometry: Theory and Apllications*, 24:135–146, 2002.
4. W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

5. P. Indyk. Sublinear time algorithms for metric space problems. In *Proceedings of the 31st ACM Symposium on Theory of Computing (STOC'99)*, 1999.
6. P. Indyk. Algorithmic applications of low-distortion geometric embeddings. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 10–33, 2001.
7. J. Kleinberg. Two algorithms for nearest neighbour search in high dimensions. In *Proceedings of the 29th ACM Symposium on Theory of Computing (STOC'97)*, pages 599–608, 1997.
8. D. S. Mitrinovic. *Analytic Inequalities*. Springer, New York, 1970.

## A The Value of $c_d$

The value of $c_d$ is given by

$$c_d = \mathbf{E}\left[\sum_{i=1}^{d} |X_i|\right] \ ,$$

where $(X_1, \ldots, X_d)$ is a point taken from the uniform distribution on the surface of the unit ball in $\mathbb{R}^d$. We observe that $(X_1^2, \ldots, X_d^2)$ is distributed as

$$\left(\frac{N_1^2}{N^2}, \ldots, \frac{N_d^2}{N^2}\right) \ ,$$

where $N^2 = \sum_{i=1}^{d} N_i^2$ and $(N_1, \ldots, N_d)$ are i.i.d. normal$(0, 1)$. Clearly,

$$\frac{N_1^2}{N^2} = \frac{N_1^2}{N_1^2 + \sum_{i=2}^{d} N_i^2} \stackrel{\mathcal{L}}{=} \frac{G(\frac{1}{2})}{G(\frac{1}{2}) + G(\frac{d-1}{2})}$$

where $G(\frac{1}{2})$, and $G(\frac{d-1}{2})$ are independent gamma$(\frac{1}{2})$ and gamma$(\frac{d-1}{2})$ random variables, respectively. Thus, $N_1^2/N$ is distributed as a beta$(\frac{1}{2}, \frac{d-1}{2})$ random variable, $\beta(\frac{1}{2}, \frac{d-1}{2})$. We have:

$$\begin{aligned}
\mathbf{E}\left[\sum_{i=1}^{d} |X_i|\right] &= d\,\mathbf{E}\left[\sqrt{\beta\left(\frac{1}{2}, \frac{d-1}{2}\right)}\right] \\
&= d \int_0^1 \frac{x^{\frac{1}{2}-1}(1-x)^{\frac{d-1}{2}-1}}{B(\frac{1}{2}, \frac{d-1}{2})} \cdot \sqrt{x} \quad dx \\
&= d \cdot \frac{B(1, \frac{d-1}{2})}{B(\frac{1}{2}, \frac{d-1}{2})} \\
&= \frac{2}{B(\frac{1}{2}, \frac{d+1}{2})} \ ,
\end{aligned}$$

where $B(a, b)$ is the beta function.

From Mitrinovic [8, p. 286], we note:

$$\frac{2}{B(\frac{1}{2}, \frac{d+1}{2})} \geq \sqrt{\frac{d}{2} + \frac{1}{4} + \frac{1}{16d+32}} \cdot \frac{1}{\Gamma(\frac{1}{2})} \tag{3}$$

$$= 2\sqrt{\frac{d}{2} + \frac{1}{4} + \frac{1}{16d+32}} \cdot \frac{1}{\sqrt{\pi}} \tag{4}$$

$$\geq \sqrt{\frac{2d+1}{\pi}} \ . \tag{5}$$

Furthermore,

$$\begin{aligned}
\mathbf{E}\left[\sum_{i=1}^{d} |X_i|\right] &= d\,\mathbf{E}[|X_1|] \\
&\leq \frac{d+1}{\sqrt{\pi} \cdot \sqrt{\frac{d}{2} + \frac{3}{4} + \frac{1}{16d+48}}} \\
&\leq \frac{2(d+1)}{\sqrt{\pi} \cdot \sqrt{2d+3}} \\
&\leq \sqrt{\frac{2(d+1)}{\pi}} \ .
\end{aligned}$$

In summary,

$$\sqrt{\frac{2d+1}{\pi}} \leq c_d = \frac{2\Gamma(\frac{d}{2}+1)}{\sqrt{\pi} \cdot \Gamma(\frac{d+1}{2})} \leq \sqrt{\frac{2(d+1)}{\pi}} \ .$$