


# Dynamic Approximate Multiplicatively-Weighted Nearest Neighbors

Boris Aronov  

Department of Computer Science and Engineering, Tandon School of Engineering,  
New York University, Brooklyn, NY, USA

Matthew J. Katz  

Department of Computer Science, Ben Gurion University of the Negev, Beer Sheva, Israel

---

## Abstract

We describe a dynamic data structure for approximate nearest neighbor (ANN) queries with respect to multiplicatively weighted distances with additive offsets. Queries take polylogarithmic time, while the cost of updates is amortized polylogarithmic. The data structure requires near-linear space and construction time.

The approach works not only for the Euclidean norm, but for other norms in  $\mathbb{R}^d$ , for any fixed  $d$ .

We employ our ANN data structure to construct a faster dynamic structure for approximate SINR queries, ensuring polylogarithmic query and polylogarithmic amortized update for the case of non-uniform power transmitters, thus closing a gap in previous state of the art.

To obtain the latter result, we needed a data structure for dynamic approximate halfplane range counting in the plane. Since we could not find such a data structure in the literature, we also show how to dynamize one of the known static data structures.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational geometry; Theory of computation  $\rightarrow$  Design and analysis of algorithms

**Keywords and phrases** Nearest neighbors, Approximate nearest neighbors, Weighted nearest neighbors, Nearest neighbor queries, SINR queries, Dynamic data structures

**Digital Object Identifier** 10.4230/LIPIcs.SWAT.2022.11

**Funding** Supported by Grants 2019715/CCF-20-08551 from the US-Israel Binational Science Foundation/US National Science Foundation.

*Boris Aronov:* Supported in part by NSF grant CCF 15-40656.

*Matthew J. Katz:* Supported in part by Grant 1884/16 from the Israel Science Foundation.

**Acknowledgements** We wish to thank Pankaj K. Agarwal for his help and encouragement, and David Mount for a clarification regarding the approximating polytope in [8].

## 1 Introduction

*Nearest-neighbor* (NN) search is a fundamental problem that has received much attention in a variety of research fields, such as databases, machine learning, and statistics. It is a central ingredient in clustering and pattern matching algorithms, as well as in numerous retrieval and recommendation systems. In this study, we are interested in the multi-shot version of the problem: Given a set  $P$  in some space  $S$  and a distance function  $d(\cdot, \cdot)$  on  $S$ , preprocess  $P$  for nearest-neighbor queries, where such a query is specified by an element  $q \in S$  and the goal is to return an element of  $P$  that is nearest to  $q$  under  $d$  among all elements in  $P$ . In this study, we shall restrict our attention to the case where the input set consists of points in  $\mathbb{R}^d$  and the distance is measured by a weighted version of a metric derived from a norm.

Let  $P = \{p_1, \dots, p_n\}$  be a set of  $n$  points in  $\mathbb{R}^d$ , such that each point  $p_i \in P$  is associated with a positive real weight  $w_i$  and a non-negative real weight  $a_i$ , and let  $\|\cdot\|$  denote any norm on  $\mathbb{R}^d$ . The distance from  $q \in \mathbb{R}^d$  to  $p_i \in P$  is now defined as  $d_{W,A}(q, p_i) := w_i d(q, p_i) + a_i$ , where  $d(q, p_i) := \|q - p_i\|$  and  $W$  and  $A$  are the sets of multiplicative



© Boris Aronov and Matthew J. Katz;

licensed under Creative Commons License CC-BY 4.0

18th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2022).

Editors: Artur Czumaj and Qin Xin; Article No. 11; pp. 11:1–11:14

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

and additive weights, respectively. We consider nearest-neighbor search in this setting, that is, following a preprocessing stage, service a sequence of queries of the form: given a query point  $q$ , return a point  $p_i \in P$  realizing  $\min_{p_i \in P} d_{W,A}(q, p_i)$ . Notice that when  $\|\cdot\|$  is the Euclidean norm, we obtain the *additively weighted* Euclidean nearest-neighbor search problem, if  $w_1 = \dots = w_n = 1$ , the *multiplicatively weighted* Euclidean nearest-neighbor search problem, if  $a_1 = \dots = a_n = 0$ , and the combined *additively and multiplicatively weighted* Euclidean nearest-neighbor search problem, otherwise.

More specifically, we consider *dynamic* approximate nearest-neighbor (ANN) search in this setting, where, in addition, points may be inserted into and deleted from  $P$  over time. An approximate nearest neighbor of  $q$  in  $P$  is a point  $p_j \in P$ , such that  $d_{W,A}(q, p_j) \leq (1 + \varepsilon)d_{W,A}(q, p_i)$ , where  $p_i$  is a nearest neighbor of  $q$  in  $P$  and  $\varepsilon > 0$  is a prespecified parameter. The reason for considering approximate, rather than exact, nearest-neighbor search is that already in the plane the induced (multiplicatively-weighted) Voronoi diagram may have complexity  $\Theta(n^2)$  [11], so constructing it explicitly for sufficiently large values of  $n$  is impractical. On the other hand, we want to be able to handle queries efficiently, typically in time polylogarithmic in  $n$ , thus we resort to approximate queries.

Although there are known solutions for approximate nearest-neighbor search (see below), we are not aware of any published technique suitable for efficient updates of  $P$ , where each such update involves the adjustment of the data structure following an insertion or a deletion of a point to/from  $P$ . We present a data structure for dynamic approximate nearest-neighbor search in this general setting. The data structure is of size  $O(tn \text{ polylog } n)$ , which is also the bound on its construction time; it supports approximate nearest-neighbor queries in  $O(t \text{ polylog } n)$  time and it can be updated in  $O(t \text{ polylog } n)$  amortized time, where  $t = \sqrt{1/\varepsilon}$  for  $d = 2$ ,  $t = 1/\varepsilon$  for  $d = 3$ , and in general  $t = 1/\varepsilon^{(d-1)/2}$  for  $d \geq 2$ .

The case where  $\|\cdot\|$  is the Euclidean norm and  $a_1 = \dots = a_n = 0$ , namely, dynamic approximate multiplicatively weighted Euclidean nearest-neighbor search, is of special interest, since (i) it applies to many practical scenarios, and (ii) it is more difficult than some of the other common cases, such as in the additively weighted scenario. In Section 5, we show that our data structure for this case enables us to significantly improve the best known solution to the most general version of the approximate SINR (signal-to-interference-plus-noise ratio) query problem. In this version, we are given a set of simultaneous transmitters, where each transmitter is represented by a point in the plane and has its own power level. Moreover, transmitters may appear or disappear over time. The goal is to construct a dynamic data structure, so that given a receiver (i.e., a point)  $q$ , one can (approximately) determine which transmitter (if any) is received at  $q$ , according to the SINR model (see Section 5). After a preprocessing stage of near-linear time, in which a data structure of near-linear size is constructed, we can answer a query in polylogarithmic time and insert or delete a transmitter in polylogarithmic amortized time. In contrast, in the best previous solution, the query time was roughly  $\sqrt{n}$  [4]. Our algorithm is randomized, with performance guarantees holding with high probability; see Theorem 5.

To obtain the latter result, we also need a data structure for handling approximate halfplane range counting queries in a dynamic setting in two dimensions. Since we could not find such a data structure in the literature, we also show how to dynamize one of the known static data structures. We include the description for completeness.

### Previous work

There is extensive literature, both in computational geometry and in other fields, on nearest-neighbor (NN) and approximate nearest-neighbor (ANN) data structures; see, for example, [9,10,15,17,18,20,26], and the book [30] for a database perspective. (As we assume throughout

that the dimension  $d$  is a small constant, the work addressing nearest-neighbor queries in high dimension [3] is out of scope for this summary.) This includes structures under the Euclidean metric and other metrics, supporting both exact and approximate queries, possibly in a dynamic setting. Moreover, many of these structures can also accommodate (with minor adjustments) additive weights (i.e., a non-negative weight  $a_i$  is added to the distance between  $q$  and  $p_i$ ). However, multiplicative weights are much more challenging, and we are only aware of two teams of researchers who studied this case in low dimension, as well as more general ones, albeit not in a dynamic setting.

Next we discuss the work of these two teams. Recall first that, if we require a near-linear-size structure, then we need to resort to approximation. Har-Peled and Kumar [22] were the first to present a near-linear-size data structure with logarithmic query time for approximate multiplicatively weighted Euclidean nearest-neighbor search. Actually, their result is for a much more general problem, where the input is a set  $\mathcal{F}$  of  $d$ -variate functions, satisfying several rather weak conditions, and the goal is to construct a data structure, so that given a query point  $q \in \mathbb{R}^d$ , one can return a function in  $\mathcal{F}$  whose value at  $q$  is at most  $1 + \varepsilon$  times the minimum value attained at  $q$  by any function in  $\mathcal{F}$ . Now, if we define  $f_i(q) = w_i \|q - p_i\|$ , we get approximate multiplicatively weighted nearest-neighbor search. The bounds that they obtain in this case are  $O(\frac{n}{\varepsilon^{2(d+1)}} \log^{d+2} n + \frac{n}{\varepsilon^{d(d+1)}})$  for the structure size,  $O(\frac{n}{\varepsilon^{2(d+1)}} \log^{2d+3} n + \frac{n}{\varepsilon^{d(d+1)}})$  for the construction time, and  $O(\log \frac{n}{\varepsilon})$  for the query time.

Subsequently, Abdelkader et al. [1] presented a data structure, based on *convexification*, for approximate multiplicatively weighted Euclidean nearest-neighbor search (see also [28]). Their result is actually for *scaling distance functions*, which also include the Minkowski distance (provided the unit ball is fat and smooth) and the Mahalanobis distance. It improves the bound on the structure size to  $O(\frac{n \log \frac{1}{\varepsilon}}{\varepsilon^{d/2}})$ , while retaining the  $O(\log \frac{n}{\varepsilon})$  bound on the query time, thus almost matching (up to a  $\log \frac{1}{\varepsilon}$  factor) the best known bounds for approximate Euclidean nearest-neighbor search [9]. However, it is not clear how fast this structure can be constructed.

## Our results

- (i) In Section 2 we describe a near-linear-size data structure that supports queries for approximate Euclidean nearest neighbors with multiplicative weights in the plane. The query and the amortized update times are both polylogarithmic.
- (ii) In Section 3 we explain how to handle other norms and also a combination of additive and multiplicative weights.
- (iii) We point out, in Section 4, that a further generalization extends the results of Section 3 to higher dimensions.
- (iv) We show how the data structures described above allow approximate dynamic SINR queries with logarithmic query times and amortized logarithmic time updates; see Section 5. The data structure can accommodate non-uniform transmitter powers; to the best of our knowledge it was not known how to achieve this performance for the case of non-uniform powers.
- (v) To facilitate the latter result, we also show in Section 6 how to dynamize with minimal overhead a data structure for approximate halfplane range counting queries.

The purpose of this paper is to demonstrate the existence of several related data structures, of near-linear size and construction time, polylogarithmic query time, and polylogarithmic amortized update time. We did not make any effort to optimize the performance of these structures and often used the “classical” well known tools as the building blocks. More

precisely, it was not our objective to optimize polylogarithmic factors; this may be a worthwhile project on its own. Hence, with one or two exceptions, we leave the precise polylogarithmic factors unspecified.

## 2 Multiplicatively weighted Euclidean nearest neighbors

For ease of presentation, we start with the basic case, that is, we consider the Euclidean norm in the plane with multiplicative weights. Generalizations to other norms, higher dimensions, and more complicated weights are described in some detail subsequently.

Consider a set  $P = \{p_1, \dots, p_n\}$  of  $n$  points in the plane. The points have an associated (multiplicative) weight function  $W: P \rightarrow \mathbb{R}^+$ ; the (positive) weight of  $p_i$  will be denoted by  $w_i := W(p_i)$ . The (*multiplicatively weighted*) distance from  $q \in \mathbb{R}^2$  to  $p \in P$  is defined as  $d_W(q, p) := W(p) \cdot d(q, p)$ , where  $d(p, q) := \|p - q\|$  stands for the Euclidean distance and  $\|\cdot\|$  – for the Euclidean norm, in this section. A *multiplicatively weighted Euclidean nearest neighbor*  $N(q) = N(q; P, W)$  of  $q \in \mathbb{R}^2$  is a point  $p \in P$  realizing  $\min_{p \in P} d_W(q, p)$ . For brevity, we will refer to  $N(q)$  simply as the *nearest neighbor* of  $q$  (in  $P$ ). (When we need to compute  $N(q)$ , if more than one point realizes the minimum distance, we are allowed to pick one of them arbitrarily.) We are interested in building a data structure that supports fast nearest-neighbor queries: preprocess a given pair  $(P, W)$ , so that  $N(q) = N(q; P, W)$  can be computed quickly for any query point  $q \in \mathbb{R}^2$ ; “quickly” here and hereafter means “in time polylogarithmic in  $n = |P|$ .”

A natural approach to preprocessing for fast nearest-neighbor queries is to consider the induced Voronoi diagram; unfortunately, in presence of multiplicative weights, the Voronoi diagram may have complexity  $\Theta(n^2)$  [11] and is thus not a viable option for a compact data structure. We therefore focus on looking for an *approximate (multiplicatively weighted Euclidean) nearest neighbor*  $\tilde{N}(q)$  for  $q$ , which is a point  $p \in P$  such that  $d_W(q, p) \leq (1 + \varepsilon)d_W(q, N(q))$ ; of course, by definition, for this point  $p$ ,  $d_W(q, N(q)) \leq d_W(q, p)$ .

### Terminology and notation

We use  $f(x) = \text{poly}(x)$  to indicate that there exists a constant  $a > 0$  such that  $f(x)$  is in  $O(x^a)$ ; similarly,  $g(x, y) = \text{poly}(x, y)$  means that there exist two constants  $a, b > 0$  such that  $g(x, y)$  is in  $O(x^a y^b)$ .

We say that an event holds *with high probability*, if it holds with probability  $1 - 1/n^c$  for a suitably large  $c > 0$ .

Finally, we say that a quantity  $k'$   $(1 + \varepsilon)$ -*approximates* a quantity  $k$ , if  $(1 - \varepsilon)k \leq k' \leq (1 + \varepsilon)k$ .

### Problem statement

DYNAMIC APPROXIMATE MULTIPLICATIVELY WEIGHTED EUCLIDEAN NEAREST NEIGHBORS IN THE PLANE

*Input:* an  $n$ -point set  $P$  in the plane with positive weights  $W$  as above and  $\varepsilon > 0$

*Output:* a data structure

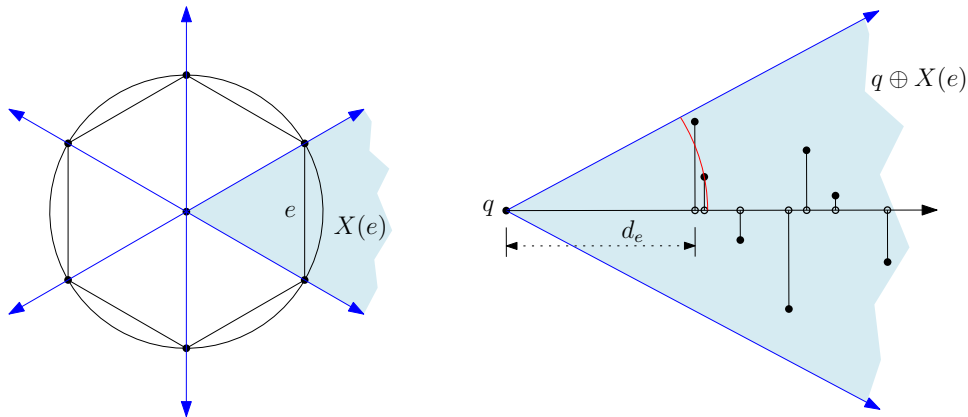
- of size  $O(n \text{poly}(\log n, 1/\varepsilon))$ ,
- constructed in time  $O(n \text{poly}(\log n, 1/\varepsilon))$ ,
- that supports approximate nearest-neighbor queries in time  $\text{poly}(\log n, 1/\varepsilon)$ , and
- insertions and deletions for  $P$  in amortized time  $\text{poly}(\log n, 1/\varepsilon)$ .

**Outline of our approach**

We use the standard approximation that replaces the unit disk  $U$  of the Euclidean norm by a regular  $k$ -gon  $U_k$  inscribed in it, for a suitable choice of  $k = \Theta(\varepsilon^{-1/2})$ . More precisely, we choose  $k$ , so that for any point  $z \in U$ , its closest point of  $U_k$  on the segment  $oz$  connecting  $z$  to the origin  $o$  is at Euclidean distance at most  $\varepsilon$  from  $z$ :  $\forall z \in U: d(z, oz \cap U_k) \leq \varepsilon$ .

The norm  $\|\cdot\|_k$  defined by  $U_k$  as the unit disk is a  $(1 + \varepsilon)$ -approximation of the Euclidean norm. We will denote the distance induced by this norm by  $d_k(\cdot, \cdot)$  and its multiplicatively weighted version by  $d_{k,W}(\cdot, \cdot)$ . We thus have  $d(q, P) \leq d_k(q, P) \leq (1 + \varepsilon)d(q, P)$ , where  $\text{dist}(q, P) := \min_{p \in P} \text{dist}(q, p)$  and  $\text{dist}$  stands for either  $d_k$  or  $d$ .

With each edge  $e$  of  $U_k$  we associate a wedge  $X(e)$  centered at the origin and delimited by the rays from the origin through the endpoints of  $e$ ; see Figure 1 (left).



**Figure 1** Left: The regular  $k$ -gon  $U_k$ , for  $k = 6$ , and the wedges  $X(e)$ . Right: Determining the distance  $d_e$  for query point  $q$ .

Given a point  $q$ ,  $d_k(q, P)$  can be computed as follows: for each edge  $e$  of  $U_k$ , let  $P_e := P \cap (q \oplus X(e))$ , or, in words, consider the subset  $P_e$  of points of  $P$  (if any) that lie in the wedge  $X(e)$  translated to  $q$ ; see Figure 1 (right). Now project the points of  $P_e$  onto the central ray of  $q \oplus X(e)$  and compute  $d_e$  as the distance from  $q$  to the closest projected point (set  $d_e = +\infty$  if  $P_e = \emptyset$ ). Then  $d_k(q, P) = \min_e d_e$ . (Up to this point, our approach is quite similar to that of Kapoor and Smid [26], who studied the basic (i.e., unweighted Euclidean) version of dynamic approximate nearest neighbors.)

We will make use of the fact that computing  $d_{k,W}(q, P)$  is a *decomposable* problem, since  $d_{k,W}(q, P_1 \cup P_2) = \min\{d_{k,W}(q, P_1), d_{k,W}(q, P_2)\}$ .

**Approximately nearest in a wedge**

We first examine a fundamental subproblem. Fix an edge  $e$  of  $U_k$  and denote its associated wedge  $X(e)$  by  $X$ . Let  $R \subseteq P$  be a subset of the given points and consider only those queries  $q$  for which  $R \subseteq q \oplus X$ . We will describe a data structure for determining, given  $q$ , its nearest neighbor  $N(q; R, W)$  in  $R$ , which we will continue to denote  $N(q)$ , slightly abusing the notation.

After a suitable rotation, we may suppose that the central ray of  $X$  coincides with the positive  $x$ -axis; recall that the apex of  $X$  is the origin; refer to Figure 1.

By definition, if  $q = (x_q, y_q)$ , the function  $d_{k,W}(q, p)$ , for  $p \in q \oplus X$ , is simply  $W(p) \cdot |x_q - x_p| = W(p)(x_p - x_q)$ , since  $p \in q \oplus X$  implies  $x_p \geq x_q$ . Therefore  $N(q)$  is precisely the point  $p \in R$  whose corresponding function achieves  $\min_{p \in R} W(p)(x_p - x_q)$ . One way to view this

## 11:6 Dynamic Approximate Multiplicatively-Weighted Nearest Neighbors

operation is to consider the graphs of functions  $f_p(x) = W(p)(x_p - x)$ , for  $p \in R$ , define the *lower envelope*  $L: \mathbb{R} \rightarrow \mathbb{R}$  by  $x \mapsto \min_{p \in R} f_p(x)$ , and then perform *vertical ray-shooting* in (the graph of)  $L$ , namely given  $x = x_q$ , identify (the graph of) the function that achieves the value  $L(x_q)$  at this  $x$ .

Being a lower envelope of a set of lines, the graph of  $L$  is the boundary of the intersection of a set of lower halfplanes in the plane, and so a monotone concave chain. In a static setting, it can be precomputed and stored in, say, an array, to facilitate vertical ray shooting via binary search on the  $x$ -coordinate of the query point  $q$ . In a dynamic setting, the intersection of lower halfplanes is dual to the upper convex hull of the dual points of the lines bounding the halfplanes. It can be stored, say, in the data structure of Overmars and van Leeuwen [29], that supports  $O(\log^2 n)$  time updates and  $O(\log n)$  time queries, where  $n = |R|$ .<sup>1</sup> A vertical ray shooting query corresponds in the dual to finding the extreme point of the hull in a given direction, one of the standard queries supported by the data structure.

### The general case

We build  $k$  data structures, one for each wedge  $X = X(e)$ . For each  $X$ , let  $u$  and  $v$  be the directions of its bounding rays and  $m$  be the direction of its central ray. We build a three-level range-search tree structure on the points of  $P$ , where the first two levels sort points of  $P$  in the directions orthogonal to  $u$  and to  $v$ , respectively. The effect of this is that a query with a point  $q$  will return the points of  $P \cap (q \oplus X)$  as a disjoint union of  $O(\log^2 n)$  canonical subsets on the second level of the structure; here we use the decomposability of the queries we are interested in. On the bottommost level, for each canonical subset  $R$ , we build the nearest-in-a-wedge data structure described above, with the distinguished direction being  $m$ , the central direction of  $X$  (rather than that of the positive  $x$ -axis).

Among the  $O(k \log^2 n)$  points returned from  $O(\log^2 n)$  canonical subsets in each of the  $k$  structures, we pick the one that is closest to  $q$  in the distinguished direction as the approximate nearest neighbor  $\tilde{N}(q)$ ; this produces the correct answer, since our query is decomposable.

Now to address the efficiency of the updates. Given  $m$  points, the corresponding nearest-in-a-wedge structure for them can be built from scratch in  $O(m \log m)$  time. This structure is fully dynamic by construction and hence the only concern is maintaining the balance in the upper levels of the overall range tree. If upper levels of the range-search structure are implemented as  $\text{BB}[\alpha]$  trees, amortized rebalancing costs are polylogarithmic, as implied by Theorem 5 in section III.5.1 of Mehlhorn's monograph [27, pages 198–199].

We summarize in the following theorem:

► **Theorem 1.** *For any  $n$ -point set  $P$  in the plane and  $\varepsilon > 0$ , we can construct a dynamic data structure for  $(1 + \varepsilon)$ -approximate multiplicatively weighted Euclidean nearest-neighbor queries in  $P$*

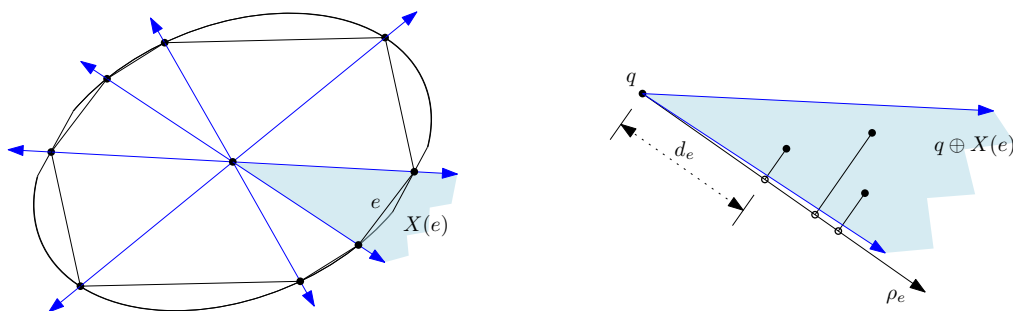
- *of size  $O(n \text{ poly}(\log n, 1/\varepsilon))$ ,*
- *in time  $O(n \text{ poly}(\log n, 1/\varepsilon))$ ,*
- *supporting  $O(\text{poly}(\log n, 1/\varepsilon))$ -time queries and*
- *$O(\text{poly}(\log n, 1/\varepsilon))$  amortized time updates.*

<sup>1</sup> Recall that, in this work, a conservative choice of the data structure that guarantees performance polylogarithmic in  $n$  and polynomial in  $1/\varepsilon$  but does not necessarily attempt to achieve optimal performance is sufficient.

► Remark. As an illustration, we calculate the actual performance characteristics using admittedly suboptimal “classical” building blocks: With the Overmars-van Leeuwen data structure for dynamic convex hulls with  $O(\log n)$  time queries and  $O(\log^2 n)$  time updates, and with  $k = O(\varepsilon^{-1/2})$  top-level structures, we obtain space and time  $O((n/\sqrt{\varepsilon}) \log^2 n)$ , query time of  $O(\varepsilon^{-1/2} \log^3 n)$  and amortized update time of  $O(\varepsilon^{-1/2} \log^4 n)$ .

### 3 Other norms

In this section we outline how the results of Section 2 generalize to an arbitrary norm on  $\mathbb{R}^2$ . Indeed, for any norm  $\|\cdot\|$ , consider its unit disk  $D := \{p \in \mathbb{R}^2 : \|p\| \leq 1\}$ .  $D$  is a compact centrally symmetric convex set with non-empty interior.<sup>2</sup> We proceed as follows: by a theorem of John [24] there exists a pair of concentric ellipses, one contained in  $D$  and one containing it, which are scaled copies of each other, with a scaling factor of at most 2. Consider an affine transformation that turns the inner ellipse into the Euclidean unit disk and the outer one into a Euclidean disk of radius at most 2. Apply this transformation to the unit ball  $D$ , to all of  $\mathbb{R}^2$ , and to  $P$  for the subsequent processing. It is easy to check that this transformation leaves the answer to the (both exact and approximate) nearest-neighbor problem unchanged. With a slight abuse of notation, we will continue to refer to objects after the transformation by the same symbols.



■ **Figure 2** Left: Approximating  $D$  by a centrally-symmetric convex  $k$ -gon  $U_k$ , and the wedges  $X(e)$ . Right: Determining the distance  $d_e$  for query point  $q$ .

After the transformation, the unit ball  $D$  is “fat” in the sense that it is sandwiched between two concentric disks with a bounded ratio of radii. It therefore can be approximated by a centrally-symmetric convex polygon  $U_k \subseteq D$  with  $k = \Theta(\varepsilon^{-1/2})$  sides, just as the Euclidean disk in Section 2 (except that now  $U_k$  is not necessarily a regular polygon) [14], see Figure 2 (left).  $U_k$  approximates  $D$  in the sense that, for any direction  $\rho$ , the ratio between the distances of the farthest points of  $D$  and  $U_k$  along the ray from the origin in direction  $\rho$  is at most  $1 + \varepsilon$ . This implies that replacing  $D$  by  $U_k$  as the unit disk distorts the distance by a factor of at most  $1 + \varepsilon$ , as desired.

We now proceed as before: We associate each of  $U_k$ ’s edges  $e$  with the wedge  $X(e)$  formed by the rays from the origin passing through  $e$ ’s endpoints. We use a shifted version  $q \oplus X(e)$  of  $X(e)$  to compute the “approximately closest” point of  $P_e = P \cap (q \oplus X(e))$  from  $q$ , where the distance from  $q$  is measured along a ray  $\rho_e$  emanating from  $q$  and orthogonal to  $e$  ( $\rho_e$  need not be the central ray of  $X(e)$  any longer), see Figure 2 (right). That is, we project the points of  $P_e$  onto  $\rho_e$ , and measure the distance  $d_e$  from  $q$  to the first projected point along  $\rho_e$ . The remainder of the argument and the data structure remain unchanged.

<sup>2</sup>  $D$  with empty interior would allow non-zero vectors of zero norm, violating the standard norm definition.

We summarize in the following theorem.

► **Theorem 2.** *For any norm in  $\mathbb{R}^2$ , there exists a dynamic data structure supporting  $(1 + \varepsilon)$ -approximate multiplicatively weighted nearest-neighbor queries with the same performance as the data structure in Theorem 1.*

Notice that at no extra cost, we can support more general approximate nearest-neighbor queries where both multiplicative and positive additive weights are present. More precisely, let  $A: P \rightarrow \mathbb{R}_0^+$  be non-negative additive weights on  $P$  and modify the distance from  $q$  to  $p \in P$  to mean  $A(p) + W(p) \cdot \|q - p\|$ , where  $\|\cdot\|$  is any norm as above. (Setting  $A(p) \equiv 0$  recovers the multiplicative-weights-only version of the problem, and setting  $W(p) \equiv 1$  recovers the familiar additive-weights-only version.) Indeed the functions  $f_p(x)$  as defined above become  $W(p)(x_p - x) + A(p)$ , i.e., remain linear, and the argument goes through verbatim.

► **Theorem 3.** *For any norm in  $\mathbb{R}^2$ , we can construct a dynamic data structure supporting  $(1 + \varepsilon)$ -approximate additively and multiplicatively weighted nearest-neighbor queries with the same performance guarantees as the data structure in Theorem 1.*

► **Remark.** In fact, our result is slightly more general. It applies to any asymmetric norm, whose unit disk is a not necessarily centrally-symmetric convex set with non-empty interior, provided that this disk is sandwiched between two Euclidean disks centered at the origin, with a constant ratio  $c$  of the outer and inner radii. In this situation the unit disk is still approximable by a  $D(c)\sqrt{\varepsilon}$ -gon where  $D(c)$  is a constant that depends on  $c$ . The argument goes through essentially without modification, taking care that the distance from  $p$  to  $q$  need not be equal to the distance from  $q$  to  $p$ .

## 4 Higher dimensions

We now outline the fairly standard procedure to extend our results from Sections 2 and 3 from the plane to any fixed dimension  $d \geq 2$ ; we assume  $d$  is a small constant. We proceed as in Section 3. Consider a norm  $\|\cdot\|$  on  $\mathbb{R}^d$  and let  $D$  be its unit ball – a compact centrally symmetric convex set with non-empty interior. We consider the Löwner-John ellipsoids for  $D$  that approximate it up to a factor of  $d$ , in the following sense: The two ellipsoids are concentric, one is contained in  $D$  while the other contains it, and the outer ellipsoid is a scaled copy of the inner one, with a scaling factor of at most  $d$  [24]. We now apply an affine transformation to the entire space turning the inner ellipsoid into the Euclidean unit ball and proceed with the transformed problem; transforming the space, the input point set  $P$ , and the unit ball  $D$  does not affect distance measurements according to  $\|\cdot\|$ . Slightly abusing the notation, we use the same symbols referring to the objects after the transformation. After the transformation,  $D$  is “fat” in the sense that it is sandwiched between two concentric balls with radius ratio of at most  $d$ .

Bronshsteyn and Ivanov [14] proved that  $D$  in this situation (see also [21] for a compact self-contained proof) can be  $(1 + \varepsilon)$ -approximated in the Hausdorff metric by a polytope with  $O(1/\varepsilon^{(d-1)/2})$  vertices. Dudley [19] showed that there exists such an approximating polytope with  $O(1/\varepsilon^{(d-1)/2})$  facets. Much more recently Arya et al. [8] proved that one can construct such an approximating polytope whose total number of faces of all dimensions is  $k := O(1/\varepsilon^{(d-1)/2})$ ;<sup>3</sup> moreover, one can assume that the resulting polytope  $U_k$  is simplicial,

<sup>3</sup> This is the best possible answer, as  $\Omega(1/\varepsilon^{(d-1)/2})$  faces are sometimes required. In fact, this is the case for the Euclidean norm.



that is each of its faces is a simplex. As in the plane,  $U_k$  approximates  $D$  in the sense that the norm  $\|\cdot\|_k$  defined by  $U_k$  as the unit ball has the property that  $\|p\| \leq \|p\|_k \leq (1 + \varepsilon)\|p\|$ . Since  $U_k$  is a simplicial polytope with at most  $k$  facets, its boundary is already triangulated by at most  $k$  simplices.

We now repeat the reasoning of Section 3 for each facet  $\Delta$  of  $U_k$  (which is a simplex):  $\Delta$  is associated with a (simplicial) cone of directions  $X = X(\Delta)$ , and for any point of  $q \in \mathbb{R}^d$ , the distance  $d_k(q, P(q, \Delta))$  from  $q$  to the closest point of  $P(q, \Delta) := P \cap (q \oplus X)$  can be computed exactly by using a suitable range-searching data structure, as  $P(q, \Delta)$  is the intersection of  $P$  with the simplicial cone  $q \oplus X$  with fixed and known directions of its  $d$  bounding hyperplanes, and the distance  $\|p - q\|_k$  for  $p \in P(q, \Delta)$  is the distance from  $q$  to the projection of  $p$  onto the ray  $\rho = \rho(\Delta)$  emanating from  $q$  and orthogonal to the hyperplane containing  $\Delta$ . Employing a suitable dynamic range structure, *à la* Section 3, completes the description; the only difference is that there are  $d$  upper levels in the structure to handle the narrowing to the cone  $X$ . Note that the bottommost level still handles dynamic vertical ray shooting in *two* dimensions.

The query structure has to be built for each of the at most  $k$  cones.

► **Theorem 4.** *For any norm in  $\mathbb{R}^d$ ,  $d \geq 2$ , any  $n$ -point set  $P$ , and  $\varepsilon > 0$ , there exists a dynamic data structure for approximate additively-and-multiplicatively weighted nearest-neighbor queries, with performance guarantees as follows:*

- *data structure has size  $O(kn \text{ polylog } n)$  and it can be constructed in this time,*
  - *updates in  $O(k \text{ polylog } n)$  amortized time, and*
  - *queries in  $O(k \text{ polylog } n)$  time,*
- where  $k = \Theta(1/\varepsilon^{(d-1)/2})$ . In particular, when  $d = 2$ ,  $k = \Theta(1/\sqrt{\varepsilon})$ , and when  $d = 3$ ,  $k = \Theta(1/\varepsilon)$ .

## 5 Dynamic SINR queries: An application

We now explain how to use the ANN data structure described in Section 2 to speed up dynamic approximate SINR (signal-to-interference-plus-noise ratio) queries for non-uniform power transmitters. We first formulate the problem, give some background and history, and then outline the application.

### Problem setup and formulation

Let  $S = \{s_1, \dots, s_n\}$  be a set of  $n$  transmitters (distinct points in the plane), and let  $p_i \geq 0$  denote the transmission power of  $s_i$ , for  $i = 1, \dots, n$ . Let  $q$  be a receiver (a point in the plane). According to the SINR model,  $q$  receives  $s_i$  if and only if

$$\frac{\frac{p_i}{d(q, s_i)^\alpha}}{\sum_{j \neq i} \frac{p_j}{d(q, s_j)^\alpha} + N} =: \text{sinr}(q, s_i) \geq \beta,$$

where  $\alpha \geq 1$  and  $\beta > 1$  are given constants,  $N$  is a given constant representing the background noise, and  $d(a, b)$  is the Euclidean distance between points  $a$  and  $b$ . The quantity  $\frac{p_i}{d(q, s_i)^\alpha}$  is the strength of the signal of the  $i$ th transmitter (located at  $s_i$  with power  $p_i$ ) as measured at the receiver  $q$ , where  $\alpha$  is the *path-loss* parameter. In words, the above inequality states that  $q$  receives  $s_i$  if and only if  $s_i$ 's signal (at  $q$ ) is at least  $\beta$  times stronger than the combined signal of all other transmitter and the noise; see, for example, [31].

Observe that, since  $\beta > 1$ ,  $q$  may receive at most one transmitter – the one for which the value  $\frac{1}{p_i^{1/\alpha}} d(q, s_i)$  is minimum. Thus, in the *uniform power* situation, where  $p_1 = p_2 = \dots = p_n$ , one needs to test reception at  $q$  for the (Euclidean) nearest neighbor of  $q$  in  $S$ , while in

## 11:10 Dynamic Approximate Multiplicatively-Weighted Nearest Neighbors

the *non-uniform power* situation, where transmitter powers vary, one needs to test reception for the multiplicatively weighted nearest neighbor of  $q$ . An *SINR query* is therefore: Given a receiver  $q$ , find the sole transmitter  $s \in S$  that may be received by  $q$  and determine whether  $\text{sinr}(q, s) \geq \beta$ .

Since it seems unlikely that one can answer an SINR query *exactly* in significantly sublinear time with the help of a near-linear-size data structure, as the degree of the polynomials involved is high, the relevant research has focused on preprocessing to facilitate efficient *approximate* SINR queries.

Given  $\varepsilon > 0$ , an *approximate SINR query* is: Given a receiver  $q$ , find the sole transmitter  $s$  that may be received by  $q$  and return a value  $\tilde{\text{sinr}}(q, s)$ , such that  $(1 - \varepsilon)\text{sinr}(q, s) \leq \tilde{\text{sinr}}(q, s) \leq (1 + \varepsilon)\text{sinr}(q, s)$ . Thus, unless  $(1 - \varepsilon)\beta \leq \tilde{\text{sinr}}(q, s) < (1 + \varepsilon)\beta$ , the value  $\tilde{\text{sinr}}(q, s)$  enables us to determine definitely whether or not  $s$  is received by  $q$ .

Given  $S$ ,  $\alpha, \beta$ , and  $N$ , as above, and  $\varepsilon > 0$ , there are four natural problems to consider, depending on whether the transmission powers are uniform or not, and whether the setting is static or dynamic (i.e., transmitters may be added to or deleted from  $S$ ). In each of these problems, the goal is to devise efficient algorithms for handling approximate SINR queries, after some preprocessing of total near-linear time, where in the dynamic setting one also needs to handle updates (i.e., insertions and deletions of transmitters) efficiently.

### Some history

For all but the most general problem, satisfactory solutions already exist, where by satisfactory we mean near-linear time preprocessing, polylogarithmic-time approximate queries and amortized polylogarithmic-time updates (in the dynamic setting) [4]; the dependence on  $1/\varepsilon$  is polynomial (see also [6, 12, 25]).

In this section, we obtain a satisfactory solution also for the most general problem. That is, we show that even if the transmission powers are non-uniform, it is possible to construct in near-linear time a dynamic data structure that supports polylogarithmic-time approximate SINR queries and amortized polylogarithmic-time updates.

### Tools required

A *static* structure supporting fast approximate SINR queries was presented in [4, section 6]. With the goal of generalizing it to support dynamic updates, inspecting the proposed algorithm, we discover that dynamizing the following two structures is sufficient for achieving our objective: The first one is a dynamic approximate multiplicatively weighted nearest-neighbor structure, which is precisely the problem in Section 2.

The second issue is handling *approximate halfplane range counting* queries in a dynamic setting; the problem is stated formally and a reasonably efficient solution is sketched in Section 6.

Examining the static algorithm of [4, section 6], we note that the approximate halfplane range counting data structure is used as the bottom level of a multilevel dynamic orthogonal range-searching structure, where approximate halfplane range counting queries are applied to bottommost canonical subsets of points. Replacing the static structure by the dynamic one from Section 6 completes the dynamization of the algorithm.<sup>4</sup>

---

<sup>4</sup> In slightly more detail, if we use  $\text{BB}[\alpha]$  trees as the basis of the orthogonal range counting structure, then, by Theorem 5 in section III.5.1 and subsequent discussion in [27, pages 198–199], the amortized cost of an update remains polylogarithmic.

We thus summarize our result.

► **Theorem 5.** *Given a set  $S$  of  $n$  transmitters in the plane, with their power transmission levels, parameters  $\alpha$ ,  $\beta$ ,  $N$ , and an approximation constant  $\varepsilon > 0$ , one can preprocess  $S$  into a data structure of worst-case size  $O(n \text{poly}(\log n, 1/\varepsilon))$  that supports approximate SINR queries in time  $O(\text{poly}(\log n, 1/\varepsilon))$  and updates in amortized time  $O(\text{poly}(\log n, 1/\varepsilon))$ .*

*The construction time is  $O(n \text{poly}(\log n, 1/\varepsilon))$  with high probability, and query and update times are with high probability, assuming the number of queries is polynomial.*

► **Remark.** As already noted in [4], the same approach generalizes to any fixed dimension  $d > 2$  with increased overhead in the dependence on  $1/\varepsilon$  and higher polylogarithmic factors. The algorithm still only needs dynamic approximate counting *in the plane* and results from Section 4 provide the machinery for finding nearest neighbors.

## 6 Dynamic approximate halfplane counting

In this section we outline a solution to the following problem which is required to complete our algorithm for efficient dynamic SINR queries in Section 5. No effort has been made to optimize the performance of the data structure; this may be an interesting question in itself.

**DYNAMIC APPROXIMATE HALFPLANE RANGE COUNTING:** Given a set  $P$  of  $n$  points in the plane and a parameter  $\varepsilon$ ,  $0 < \varepsilon < 1$ , preprocess  $P$  in  $O(n \text{poly}(\log n, 1/\varepsilon))$  time so that, given a query halfplane  $h$ , a  $(1 + \varepsilon)$ -approximation of the number  $|P \cap h|$  can be returned in time  $O(\text{poly}(\log n, 1/\varepsilon))$ . Moreover, insertions and deletions can be processed in  $O(\text{poly}(\log n, 1/\varepsilon))$  amortized time.

We sketch a Monte Carlo algorithm for this problem, where the  $(1 + \varepsilon)$ -approximation is correct *with high probability* (i.e., with probability  $1 - 1/n^c$ , for a sufficiently large  $c > 0$ ), provided one makes a polynomial number of queries, assuming that the updates to the data structure do not depend on the random choices made by the algorithm.

The easiest, though perhaps not the most efficient method to achieve our goal is to use the “black-box” reduction of Aronov and Har-Peled [5], who observed that an approximate range counting structure can be obtained, at the cost of multiplicative overhead of  $\text{poly}(\log n, 1/\varepsilon)$  in construction time and space and in query time, from a data structure for *emptiness testing*. In our context, emptiness testing is, given a set  $P$  of points in the plane, preprocess it so that, for a query halfplane  $h$ , one can quickly check whether or not  $h \cap P = \emptyset$ . The nature of the reduction is constructing a number of emptiness-testing structures on randomly chosen subsets of  $P$ . In our case, emptiness testing is easily dynamized by using, say, the classical dynamic convex hull data structure of Overmars-van Leeuwen [29]. Random subsets generated by the reduction involve picking each point of  $P$  independently with a given probability and so can be updated efficiently on insertion into and deletion from  $P$ . The number of such subsets is  $\text{poly}(\log n, 1/\varepsilon)$ , so an update cannot affect too many of them, even in the worst case. (Some additional aspects of the structure depend on the value of  $n$ , but these can be handled, as is standard, by periodic rebuilds when  $n$  changes substantially.)

If one is interested in improving the performance of this data structure, there are several natural avenues of improvement:

- One can replace the dynamic convex hull structure of Overmars-van Leeuwen [29] by a faster alternative, such as [16] or [13, 23]; refer to the introduction of the latter reference for a more thorough literature survey.

## 11:12 Dynamic Approximate Multiplicatively-Weighted Nearest Neighbors

- Using the *black-box* reduction from [5] likely affects efficiency, see more elaborate non-black-box work in [7] (though that work focuses on the higher-dimensional case and does not address supporting updates).
- Afshani and Chan [2] describe a more efficient Monte Carlo black-box reduction from approximate counting to *small-count* range searching, where a query needs to check if  $|P \cap h|$  is small for a halfplane  $h$ , and if so, produce the correct answer, and otherwise output TOO LARGE. The algorithm essentially builds small-count structures for random samples of the input, of various sizes, conceptually similar to the reduction from [5]. As the counts required in the reduction are essentially of size  $O(\text{poly}(1/\varepsilon) \log n)$ , one could implement such a dynamic data structure in (near-)linear space with efficient updates using classical tools such as [29] or their subsequent improvements. One could even employ a dynamic *reporting* structure in this situation, provided it can interrupt the reporting if the query answer is too large; most reporting structures have such an ability.
- We conclude with an open problem: All the reductions described so far produce a Monte Carlo algorithm for dynamic approximate halfplane range counting. Is there a (simple and straightforward) Las Vegas algorithm with comparable performance? Or even a deterministic one?

### 7 Discussion and open problems

- Our data structures in Sections 2 through 4 can handle *farthest* rather than *nearest* neighbors, with simple and obvious modifications.
- It might be interesting to determine the best performance that can be attained in Theorems 1 through 4 with the current state-of-the-art data structures.
- The static data structures of Har-Peled and Kumar [22] and of Abdelkader et al. [1] cover a large class of (weighted) distance measures. In particular, both structures allow assigning different norms to different sites, which our approach cannot accommodate.

Refer to the end of Section 6 for some discussion and open problems related to the approximate halfplane range counting data structure described there.

---

### References

- 1 Ahmed Abdelkader, Sunil Arya, Guilherme Dias da Fonseca, and David M. Mount. Approximate nearest neighbor searching with non-Euclidean and weighted distances. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 355–372. SIAM, 2019. doi:10.1137/1.9781611975482.23.
- 2 Peyman Afshani and Timothy M. Chan. On approximate range counting and depth. *Discret. Comput. Geom.*, 42(1):3–21, 2009. doi:10.1007/s00454-009-9177-z.
- 3 Alexandr Andoni and Piotr Indyk. Nearest neighbors in high-dimensional spaces. In *Handbook of Discrete and Computational Geometry*, pages 1135–1155. Chapman and Hall/CRC, 2017.
- 4 Boris Aronov, Gali Bar-On, and Matthew J. Katz. Resolving SINR queries in a dynamic setting. *SIAM J. Comput.*, 49(6):1271–1290, 2020.
- 5 Boris Aronov and Sariel Har-Peled. On approximating the depth and related problems. *SIAM J. Comput.*, 38(3):899–921, 2008.
- 6 Boris Aronov and Matthew J. Katz. Batched point location in SINR diagrams via algebraic tools. *ACM Transactions on Algorithms*, 14(4):41:1–41:29, 2018.
- 7 Boris Aronov and Micha Sharir. Approximate halfspace range counting. *SIAM J. Comput.*, 39:2704–2725, 2010.

- 8 Rahul Arya, Sunil Arya, Guilherme Dias da Fonseca, and David M. Mount. Optimal bound on the combinatorial complexity of approximating polytopes. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 786–805. SIAM, 2020. doi:10.1137/1.9781611975994.48.
- 9 Sunil Arya, Guilherme Dias da Fonseca, and David M. Mount. Optimal approximate polytope membership. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 270–288. SIAM, 2017. doi:10.1137/1.9781611974782.18.
- 10 Sunil Arya, Theodoros Malamatos, and David M. Mount. Space-time tradeoffs for approximate nearest neighbor searching. *J. ACM*, 57(1):1:1–1:54, 2009. doi:10.1145/1613676.1613677.
- 11 Franz Aurenhammer and Herbert Edelsbrunner. An optimal algorithm for constructing the weighted Voronoi diagram in the plane. *Pattern Recogn.*, 17:251–257, 1984. URL: [https://dx.doi.org/10.1016/0031-3203\(84\)90064-5](https://dx.doi.org/10.1016/0031-3203(84)90064-5).
- 12 Chen Avin, Yuval Emek, Erez Kantor, Zvi Lotker, David Peleg, and Liam Roditty. SINR diagrams: Convexity and its applications in wireless networks. *J. ACM*, 59(4):18:1–18:34, 2012. doi:10.1145/2339123.2339125.
- 13 Gerth Stølting Brodal and Riko Jacob. Dynamic planar convex hull. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 617–626. IEEE, 2002. See also [23].
- 14 Efim M. Bronshteyn and L.D. Ivanov. The approximation of convex sets by polyhedra. *Siberian Mathematical Journal*, 16(5):852–853, 1975.
- 15 Timothy M. Chan. Approximate nearest neighbor queries revisited. *Discret. Comput. Geom.*, 20(3):359–373, 1998. doi:10.1007/PL00009390.
- 16 Timothy M Chan. Dynamic planar convex hull operations in near-logarithmic amortized time. *Journal of the ACM (JACM)*, 48(1):1–12, 2001.
- 17 Timothy M. Chan. Applications of Chebyshev polynomials to low-dimensional computational geometry. *J. Comput. Geom.*, 9(2):3–20, 2018. doi:10.20382/jocg.v9i2a2.
- 18 Kenneth L. Clarkson. A randomized algorithm for closest-point queries. *SIAM J. Comput.*, 17(4):830–847, 1988. doi:10.1137/0217052.
- 19 Richard M Dudley. Metric entropy of some classes of sets with differentiable boundaries. *Journal of Approximation Theory*, 10(3):227–236, 1974.
- 20 Sariel Har-Peled. A replacement for Voronoi diagrams of near linear size. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 94–103. IEEE Computer Society, 2001. doi:10.1109/SFCS.2001.959884.
- 21 Sariel Har-Peled and Mitchell Jones. Proof of Dudley’s convex approximation, 2019. arXiv:1912.01977.
- 22 Sariel Har-Peled and Nirman Kumar. Approximating minimization diagrams and generalized proximity search. *SIAM J. Comput.*, 44(4):944–974, 2015.
- 23 Riko Jacob and Gerth Stølting Brodal. Dynamic planar convex hull, 2019. arXiv:1902.11169.
- 24 Fritz John. Extremum problems with inequalities as subsidiary conditions. *R. Courant Anniversary Volume*, pages 187–204, 1948.
- 25 Erez Kantor, Zvi Lotker, Merav Parter, and David Peleg. The topology of wireless communication. In *Proceedings 43rd ACM Symposium on Theory of Computing, STOC 2011*, pages 383–392, 2011. URL: <https://doi.acm.org/10.1145/1993636.1993688>, doi:10.1145/1993636.1993688.
- 26 Sanjiv Kapoor and Michiel H. M. Smid. New techniques for exact and approximate dynamic closest-point problems. *SIAM J. Comput.*, 25(4):775–796, 1996. doi:10.1137/S0097539793259458.
- 27 Kurt Mehlhorn. *Data Structures and Algorithms 1: Sorting and Searching*, volume 1 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1984. doi:10.1007/978-3-642-69672-5.

## 11:14 Dynamic Approximate Multiplicatively-Weighted Nearest Neighbors

- 28 David M. Mount. New directions in approximate nearest-neighbor searching. In Sudebkumar Prasant Pal and Ambat Vijayakumar, editors, *Algorithms and Discrete Applied Mathematics - 5th International Conference, CALDAM 2019, Kharagpur, India, February 14-16, 2019, Proceedings*, volume 11394 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2019. doi:10.1007/978-3-030-11509-8\_1.
- 29 Mark H. Overmars and Jan van Leeuwen. Maintenance of configurations in the plane. *J. Comput. Syst. Sci.*, 23(2):166–204, 1981. doi:10.1016/0022-0000(81)90012-X.
- 30 A. N. Papadopoulos and Y. Manolopoulos. *Nearest Neighbor Search: A Database Perspective*. Springer US, 2005.
- 31 Theodore S. Rappaport. *Wireless Communications – Principles and Practice*. Prentice Hall, 1996.