Sublinear Average-Case Shortest Paths in Weighted Unit-Disk Graphs

Adam Karczmarz
Institute of Informatics, University of Warsaw, Poland

Jakub Pawlewicz
Institute of Informatics, University of Warsaw, Poland

Piotr Sankowski
Institute of Informatics, University of Warsaw, Poland

Abstract

We consider the problem of computing shortest paths in weighted unit-disk graphs in constant dimension d. Although the single-source and all-pairs variants of this problem are well-studied in the plane case, no non-trivial exact distance oracles for unit-disk graphs have been known to date, even for $d = 2$.

The classical result of Sedgewick and Vitter [Algorithmica '86] shows that for weighted unit-disk graphs in the plane the A^* search has average-case performance superior to that of a standard shortest path algorithm, e.g., Dijkstra's algorithm. Specifically, if the n corresponding points of a weighted unit-disk graph G are picked from a unit square uniformly at random, and the connectivity radius is $r \in (0, 1)$, A^* finds a shortest path in G in $O(n)$ expected time when $r = \Omega(\sqrt{\log n/n})$, even though G has $\Theta((nr)^2)$ edges in expectation. In other words, the work done by the algorithm is in expectation proportional to the number of vertices and not the number of edges.

In this paper, we break this natural barrier and show even stronger sublinear time results. We propose a new heuristic approach to computing point-to-point exact shortest paths in unit-disk graphs. We analyze the average-case behavior of our heuristic using the same random graph model as used by Sedgewick and Vitter and prove it superior to A^*. Specifically, we show that, if we are able to report the set of all k points of G from an arbitrary rectangular region of the plane in $O(k + t(n))$ time, then a shortest path between arbitrary two points of such a random graph on the plane can be found in $O(1/r^2 + t(n))$ expected time. In particular, the state-of-the-art range reporting data structures imply a sublinear expected bound for all $r = \Omega(\sqrt{\log n/n})$ and $O(\sqrt{n})$ expected bound for $r = \Omega(n^{-1/4})$ after only near-linear preprocessing of the point set.

Our approach naturally generalizes to higher dimensions $d \geq 3$ and yields sublinear expected bounds for all $d = O(1)$ and sufficiently large r.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases unit-disk graphs, shortest paths, distance oracles

Digital Object Identifier 10.4230/LIPIcs.SoCG.2021.46

Related Version The full version of this paper also describes a slightly different query algorithm whose running time has a better dependence on n for higher dimensions $d > 2$.
Full Version: `https://arxiv.org/abs/2103.09684`

Funding *Adam Karczmarz*: supported by ERC Consolidator Grant 772346 TUgbOAT and by the Foundation for Polish Science (FNP) via the START programme.
Jakub Pawlewicz: supported by ERC Consolidator Grant 772346 TUgbOAT.
Piotr Sankowski: supported by ERC Consolidator Grant 772346 TUgbOAT.

1 Introduction

Computing shortest paths is certainly one of the most fundamental graph problems and has numerous theoretical and practical applications. The two classical variants of the shortest paths problem are *single-source shortest paths* (SSSP) and *all-pairs shortest path* (APSP). A common generalization of these variants is the *distance oracle* problem, where we are allowed to preprocess a given network into a (possibly small) data structure that is later used to answer arbitrary *point-to-point* shortest paths queries. Clearly, SSSP and APSP algorithms can be viewed as extreme solutions to the distance oracle problem: the former can be used without any preprocessing to query a distance in near-linear time, whereas the latter precomputes the answers to all the n^2 possible queries and thus can answer queries in constant time. Hence, when constructing distance oracles we seek a tradeoff between these two extremes. Unfortunately, it is not known how to obtain a non-trivial (with both subquadratic space and sublinear query time) *exact* distance oracle for general graphs. Subquadratic space and constant query time oracle is only known for undirected weighted graphs if approximation factor of at least 3 is allowed [19].

Due to this theoretical inefficiency of distance oracles, researchers either focus on special graph classes, or study approximate approaches. On one hand, near-optimal (in terms of space and query time) exact distance oracles have been recently described for planar graphs [8]. On the other hand, for many important graph classes near-optimal $(1+\epsilon)$-approximate distance oracles are known [15, 2].

Nevertheless, in practice heuristic approaches are usually preferable – for an overview of used techniques see [3]. However, the term "heuristic" in this domain usually refers to ways of speeding up exact algorithms. There are some examples of heuristics that have been analyzed theoretically and proved to yield speedups in meaningful settings, see e.g., [1].

Perhaps the most well-known heuristic approach to speeding up a shortest path computation is a variant of Dijkstra's algorithm called the *A^* search* [13]. This algorithm incorporates a heuristic function that lower-bounds the distance to the target and uses it to decide which search paths to follow first. The algorithm is still guaranteed to find the shortest path to the target vertex, but the number of vertices explored can be much smaller compared to the standard Dijkstra's algorithm. For example, if vertices of the network correspond to points in the plane, the Euclidean distance to the target is a valid and well-working heuristic function. The natural question arises when such algorithms perform provably better than in the worst-case.

1.1 Shortest paths in weighted unit-disk graphs

The seminal result that answers this question is by Sedgewick and Vitter [18], who studied the performance of A^* search on various random geometric graph models. Perhaps the most interesting of their results concerns the *weighted unit-disk graphs*. In a weighted unit-disk graph with *connectivity radius* r, vertices correspond to points on the plane. An edge between two distinct vertices (points) u, v exists in such a graph if $||u - v||_2 \le r$ and has weight $||u - v||_2$. This class of geometric graphs has been widely studied from the algorithmic perspective since such graphs can model e.g., ad-hoc communication networks. A *random weighted unit-disk graph* G, given n and radius $r \in (0, 1)$, is obtained from a set of n random points of a unit square $[0, 1]^2$. Note that such a random G has $\Theta((nr)^2)$ edges in expectation. However, Sedgewick and Vitter [18] show that, assuming that the neighbors of each vertex in G are stored explicitly, one can compute a point-to-point shortest path in G using A^*

search in $O(n)$ expected time, i.e., independent of r and sublinear in the size of the edge set of G. In other words, they have given an exact distance oracle for random weighted unit-disk graphs that in expectation requires $O((nr)^2)$ space and $O(n)$ query time.

Sedgewick and Vitter's result [18] can be also interpreted as follows: for weighted unit-disk graphs $G = (V, E)$, just storing the graph explicitly allows $O(n)$-time queries for an average-case graph G. Whereas such a query time is sublinear in the graph size, the $\Theta((nr)^2)$ space used might be *superlinear* in the graph's description – observe that a weighted unit-disk graph can be described using $O(n)$ space solely with n point locations and the connectivity radius r. In recent years efficient single-source shortest paths algorithm for weighted unit-disk graphs have been proposed [4, 14, 20], culminating in the $O(n \log^2 n)$ algorithm of Wang and Xue [20]. Note that their worst-case bound is near-optimal and almost matches the bound of [18] which holds only on average. All-pairs shortest paths in weighted unit-disk graphs can be computed slightly faster that running single-source computations n times [6]. To the best of our knowledge, no exact distance oracle with non-obvious space and query bounds for this graph class is known. On the contrary, a very efficient $(1+\epsilon)$-approximate distance oracle with near-optimal space, preprocessing, and query bounds was given by Chan and Skrepetos [7].

The notion of a weighted unit-disk graph naturally generalizes to three- and higher dimensions: an edge between two vertices appears if the d-dimensional balls of radius r at these points intersect. We are not aware of any non-trivial results on computing shortest paths in such graphs for $d \geq 3$.

1.2 Our results

Observe that all of the above algorithms in order to answer distance queries require work essentially proportional to the number of vertices and not the number of edges. In this paper, we break this natural barrier and show an even stronger sublinear time results.

We propose a natural heuristic approach to computing exact shortest paths in weighted unit-disk graphs. Following Sedgewick and Vitter, we analyze its average-case query time by studying its performance on a random n-vertex graph with connectivity radius r in the unit square $[0,1]^2$, where $r = \Omega\left(\sqrt{\log(n)/n}\right)$.[1] In this setting, we prove that after near-linear preprocessing, the query procedure of our average-case distance oracle has $O(1/r^2 + \sqrt{n})$ expected running time. Formally, we prove:

▶ **Theorem 1.** *Let $r \in (0,1)$ be such that $r = \Omega\left(\sqrt{\log(n)/n}\right)$. Let G be a weighted unit-disk graph with connectivity radius r on a set P of n points picked uniformly at random from the unit square $[0,1]^2$. Let $\mathcal{D}$ be a data structure that, after preprocessing P in $O(p(n))$ time, supports reporting all k points in P lying in an arbitrary (not necessarily orthogonal) rectangular subregion in $O(k + t(n))$ time. Then, there exists an exact distance oracle on G with $O(p(n))$ preprocessing time and $O(1/r^2 + t(n))$ expected query time.*

The state-of-the-art range searching data structures [5] imply that $t(n) = O(\sqrt{n})$ using $O(n)$ space and $O(n \log n)$ preprocessing. Consequently, for $r = \Omega(1/n^{1/4})$ the expected query time is $O(\sqrt{n})$ and it remains truly sublinear for all $r = \Omega(\sqrt{\log(n)/n})$ – improving the running time of Sedgewick and Vitter in the full range of parameters r they consider.

[1] This simplifying assumption has also been made by Sedgewick and Vitter [18] and excludes only very sparse graphs with $m = O(n \log n)$ from our consideration. Moreover, it is known that if $r = o(\sqrt{\log(n)/n})$, then the random unit-disk graph is disconnected with high probability [12].

The general idea behind our heuristic algorithm for computing a shortest $s - t$ path is fairly intuitive: we run the single-source shortest paths algorithm limited to increasingly "fat" rectangular subregions of G surrounding the segment $s - t$. The subregions of interest are computed using a range reporting data structure which constitutes the only preprocessed information of our oracle. Since dynamic variants of such range searching data structures are known [16] (with query and space bound matched up to polylogarithmic factors, and polylogarithmic update bounds), our heuristic distance oracle can be trivially dynamized as well (see Remark 12).

Another advantage of our algorithm is that it easily generalizes to higher dimensions. Using new ideas we prove that for random weighted unit-disk graphs[2] in $[0,1]^d$, the expected query time is $O(\min(1/r^{2d-1}, n) + t_d(n))$, assuming one can report the points from an arbitrary (not necessarily orthogonally aligned) d-dimensional hyperrectangle in $O(t_d(n) + k)$ time. It is known [5] that $t_d(n) = O(n^{1-1/d})$ so this expected time is sublinear in n unless $r = \Omega\left(n^{-\frac{1}{2d-1}}\right)$. It is worth noting that for $d = 2$, the expected query time has a "better" dependence, i.e., $O(1/r^d)$, on r than for $d \geq 3$ where the dependence is $O(1/r^{2d-1})$. This is justified by the fact that whereas single-source shortest paths in weighted unit-disk graphs for $d = 2$ can be computed nearly-optimally [20], no non-trivial algorithm like this is known for $d \geq 3$ and we have to resort to running the standard Dijkstra's algorithm.

Undoubtedly, the technical difficulty of our result lies in the probabilistic analysis. We use similar approach to the one used by Sedgewick and Vitter [18] to bound the probability that the sought path exist in ellipsoidal grid-like regions called *channels*. However, in order to avoid looking at all the edges incident to a vertex we need to use a new heuristic that allows us to consider only edges induced within an rectangular region.

Interestingly, we also identify a shortcoming in their original analysis for the two-dimensional case and give a more delicate argument inspired by the techniques from so-called *oriented percolation theory* (see e.g., [10]). The original result of Sedgewick and Vitter [18] wrongly limited the sets of directed paths going through the channel grid. Thus the resulting probability that a path exists was overestimated. The more detailed description of the shortcoming of the original proof can be found in the full version of this paper.

We note that for $d = 2$ the graph model considered here has been widely studied in the context of wireless networks [12]. For example, Gupta and Kumar [11] studied the connectivity of such networks, and have shown a critical r above which the graph is connected with high probability. This result was generalized by Penrose [17] to k-connectivity. Our result gives the first known sublinear shortest path routing oracle for such networks. In a sense, our results call for further work on exact distance oracles for weighted unit disk graphs. In particular it might suggest that near-linear space and sublinear query time exact distance oracles in the worst-case exist, as proving such result over random graphs can be a seen as a proof-of-concept for such a possibility.

2 Preliminaries

A *weighted unit-disk graph* $G = (V, E)$ with *connectivity radius* r is an undirected geometric graph whose vertices are identified with some n points in $\mathbb{R}^d$, where $d \geq 2$ is a constant. The edge set of G contains an edge $\{u, v\}$ for all $u = (u_1, \dots, u_d)$, $v = (v_1, \dots, v_d) \in V$ such that $||u - v||_2 = \sqrt{\sum_{i=1}^{d}(u_i - v_i)^2} \leq r$. For brevity, in the following we omit the subscript and write $||x - y||$ instead of $||x - y||_2$.

[2] Since a disk is a subset of a plane, in higher dimensions $d > 2$, it would be perhaps more appropriate to call such graphs *weighted unit-ball graphs*. However, anyway, we stick to the well-established term *weighted unit-disk graph* since our main result concerns the plane case $d = 2$.

For $u, v \in V$, by $\text{dist}_G(u, v)$ we denote the length of a shortest $u \to v$ path in G.

We consider *exact distance oracles* for weighted unit-disk graphs G, i.e., data structures that preprocess G (ideally into a near-linear space data structure using near-linear time) and then accept point-to-point distance queries, i.e., given query vertices $u, v \in V$, compute $\text{dist}_G(u, v)$. The algorithms we propose can be straightforwardly extended to also report actual shortest paths within the same asymptotic time bound. Hence, we focus only on computing distances.

In order to perform a meaningful average-case analysis of a distance oracle's query algorithm on weighted unit-disk graphs for a given r, we need to limit the space of possible graphs. To this end, following Sedgewick and Vitter [18], for $r \in (0, 1)$ we limit our attention to graphs with all n points in $[0, 1]^d$. In order to compute the average running time of a shortest path query, we would like to compute it over all possible such graphs. Equivalently, we study the expected running time of a query algorithm on a *random graph* G, where each of n points is picked uniformly at random from $[0, 1]^d$. Note that in such a case, each vertex w has $\Theta(nr^d)$ neighbors in expectation: the probability that another vertex z is connected with w with an edge equals the probability that z is picked in the d-dimensional ball of radius r around v which clearly has volume $\Theta(r^d)$.

We also assume $r \geq \left(\frac{\beta \log n}{n}\right)^{1/d}$ for a sufficiently large constant $\beta > 1$. Then, the random graph G has $\Omega(n \log n)$ edges in expectation. For $d = 2$, the bound $r = \Omega\left(\left(\frac{\log n}{n}\right)^{1/d}\right)$ has also been assumed by Sedgewick and Vitter [18], as it greatly simplifies calculations. Moreover, for $r = o\left(\left(\frac{\log n}{n}\right)^{1/d}\right)$, with high probability G is not connected [11].

3 The distance oracle

3.1 Preprocessing

Let the coordinates of the n points of an input weighted unit-disk graph G be given. In the preprocessing phase, in $O(n \log n)$ time we build a simplex range searching data structure on V [5]. This data structure requires only linear space and allows $O(n^{1-1/d} + k)$ worst-case time queries reporting all of the k input points in an arbitrary hyperrectangle (with sides not necessarily parallel to the axes) of $\mathbb{R}^d$.

3.2 Query algorithm

Suppose the query is to compute $\text{dist}_G(s, t)$ for $s, t \in V$. Let

$$w = ||t - s||.$$

Clearly, we have $\text{dist}_G(s, t) \geq w$. Moreover, in the following we assume $w > r$, since otherwise we trivially have $\text{dist}_G(s, t) = w$.

Let us first move and rotate the coordinate system so that the origin is now in s and the direction of the first axis is the same as $\overrightarrow{st}$, thus we have $s = (0, 0, \ldots, 0)$ and $t = (w, 0, \ldots, 0)$ in the new coordinate system.

▶ **Observation 2.** *Let $W \geq w$ denote an upper bound on $\text{dist}(s, t)$. If a s–t shortest path in G contains a vertex $x \in V$ then*

$$||x - s|| + ||x - t|| \leq W. \qquad (1)$$

Inequality (1) describes a set of points contained in a d-dimensional ellipsoid. The first axis of that ellipsoid has length $W/2$, whereas all other $d-1$ axes have length R, where R satisfies $(w/2)^2 + R^2 = (W/2)^2$. Hence:

$$R = \frac{1}{2}\sqrt{W^2 - w^2}.$$

Note that the ellipsoid is contained in a d-dimensional *bounding box*

$$\left[-\frac{W-w}{2}, \frac{W+w}{2}\right] \times [-R, R] \times \ldots \times [-R, R] \tag{2}$$

with first side length equal to W and the other $d-1$ side lengths equal to $2R$.

We will later pick an unbounded increasing function $W_{\text{ub}} : \mathbb{Z}_+ \to \mathbb{R}_+$ with values depending on n, d, r, with the goal of defining increasingly large bounding boxes, as follows.

▶ **Definition 3.** *For a given integer $i \geq 1$, by $\text{BE}(i)$ we denote the set of points satisfying inequality (1) for $W = W_{\text{ub}}(i)$. Similarly, by $\text{BB}(i)$ we denote the bounding box as in formula (2) for $W = W_{\text{ub}}(i)$.*

Our entire algorithm will be to run a single-source shortest paths algorithm on the graphs

$$G(i) = (V_i, E_i) = G \cap \text{BB}(i),$$

subsequently for $i = 1, 2, \ldots, i_{\max}$ (where $i_{\max}$ is to be set later) until an $s \to t$ path of length no more than $W_{\text{ub}}(i)$ is found. If we are successful with that for some i, the found path is returned as the shortest $s \to t$ path. Otherwise, we simply run Dijkstra's algorithm from s on the entire G and either return the found shortest $s \to t$ path, or return ∞ if no path is found.

▶ **Lemma 4.** *The above algorithm is correct.*

Proof. The algorithm clearly stops. Moreover, the final Dijkstra step ensures that an $s \to t$ path is found if and only if a $s \to t$ path in G exists.

To prove correctness suppose that $\text{dist}_G(s,t) < \infty$. Let i^* be the first i for which $\text{dist}_{G(i^*)}(s,t) \leq W_{\text{ub}}(i^*)$, if such i^* exists. Since $G(i^*) \subseteq G$, $\text{dist}_G(s,t) \leq \text{dist}_{G(i^*)}(s,t)$ and hence $\text{dist}_G(s,t) \leq W_{\text{ub}}(i^*)$. So, by Observation 2, a path of length $\text{dist}_G(s,t)$ has all its vertices in $\text{BE}(i^*) \subseteq \text{BB}(i^*)$. This proves $\text{dist}_G(s,t) \geq \text{dist}_{G(i^*)}(s,t)$, so in fact $\text{dist}_G(s,t) = \text{dist}_{G(i^*)}(s,t)$.

If i^* does not exists, we run Dijkstra's algorithm on the entire graph G, so clearly a shortest $s \to t$ path is returned. ◀

Let $\text{T}^V_{\text{gen}}(i)$ and $\text{T}^E_{\text{gen}}(i)$ be the times required to find sets V_i and E_i, respectively. Since V_i is defined as a subset of V inside a d-dimensional bounding box $\text{BB}(i)$, it can be clearly computed using a single query to the preprocessed range searching data structure. Hence,

$$\text{T}^V_{\text{gen}}(i) = O(n^{1-1/d} + |V_i|).$$

Denote by $T_d(i)$ the worst-case running time of step i. The cost $T_d(i)$ might differ depending on the algorithm that we use to find a shortest path in $G(i)$. Note that $G(i)$ is a weighted unit-disk graph, so if $d = 2$, and we employ the recent nearly-linear (in the number of vertices), albeit difficult to implement, algorithm of Wang and Xue [20], so we have:[3]

$$T_2(i) = O\left(|V_i| \log^2(|V_i| + 2) + \text{T}^V_{\text{gen}}(i)\right) = O\left(|V_i| \log^2(|V_i| + 2) + \sqrt{n}\right). \tag{3}$$

[3] We use $\log(|V_i| + 2)$ instead of just $\log |V_i|$ to make sure this term is at least a positive constant.

On the other hand, if $d > 2$, we need to use the simple-minded Dijkstra's algorithm to find a shortest path in $G(i)$, so we have

$$T_d(i) = O\left(|V_i|\log(|V_i|+2) + |E_i| + \mathrm{T}^E_{\mathrm{gen}}(i)\right). \tag{4}$$

Let $\bar{P}(i)$ be the probability that we fail to find a path of length at most $W_{\mathrm{ub}}(i)$ in the graph G_i. The expected running time of the algorithm is then

$$O\left(\sum_{i=1}^{i_{\max}} \bar{P}(i-1)\cdot\mathbb{E}[T_d(i)] + \bar{P}(i_{\max})\cdot n^2\right). \tag{5}$$

We will prove that by choosing

$$i_{\max} = \Theta(nr^d), \tag{6}$$

and

$$W_{\mathrm{ub}}(i) = \Theta\left(w\cdot\sqrt{1+\left(\frac{i}{nr^d}\right)^{\frac{2}{d-1}}}\right) = O(w), \tag{7}$$

as described precisely in Section 5, we can obtain the following key bound. The proof of this bound is covered in Sections 4 and 5.

▶ **Lemma 5.** *For $i = 1,\dots,i_{max}$, $\bar{P}(i) \le e^{-i}$.*

We now derive bounds on the expected sizes of sets V_i and E_i.

▶ **Lemma 6.** *For $i = 1,\dots,i_{max}$, $\mathbb{E}[|V_i|] = \Theta\left((w/r)^d i\right)$.*

Proof. Clearly, $\mathbb{E}[|V_i|]$ equals the volume of $\mathrm{BB}(i)$ times n. For $W = W_{\mathrm{ub}}(i)$ we have

$$R = \frac{1}{2}\sqrt{W^2 - w^2} = \Theta\left(w\cdot\left(\frac{i}{nr^d}\right)^{\frac{1}{d-1}}\right). \tag{8}$$

Since $\mathrm{BB}(i)$ has size $W \times 2R \times \ldots \times 2R$, its volume is

$$W\cdot(2R)^{d-1} = \Theta(w)\cdot\Theta(R^{d-1}) = \Theta(w)\cdot\Theta\left(\frac{w^{d-1}i}{nr^d}\right) = \Theta\left(\frac{1}{n}\cdot\left(\frac{w}{r}\right)^d\cdot i\right).$$ ◀

In order to analyse the running time we will need the following technical lemma whose proof can be found in the full version.

▶ **Lemma 7.** *Let X be a random variable from a binomial distribution with n variables and mean $\mathbb{E}[X] = \mu = \Omega(1)$. Then for any constant integer $\alpha \ge 1$:*

$$\mathbb{E}[X\cdot\log^\alpha(X+2)] = O(\mathbb{E}[X]\cdot\log^\alpha(\mathbb{E}[X]+2)) = O(\mu\cdot\log^\alpha(\mu+2))).$$

▶ **Corollary 8.** *For any integer $\alpha \ge 1$ we have*

$$\mathbb{E}[|V_i|\log^\alpha(|V_i|+2)] = O(\mathbb{E}[|V_i|]\cdot\log^\alpha(\mathbb{E}[|V_i|]+2)).$$

Proof. We can apply Lemma 7 since $\mathbb{E}[|V_i|] = \Omega(1)$ by Lemma 6. ◀

▶ **Lemma 9.** *Let*

$$f^E(i) = \min\{nr^d, (w/r)^{d-1}i\} = \begin{cases} (w/r)^{d-1}i & \text{for } r \geq \sqrt[2d-1]{w^{d-1}i/n} \\ nr^d & \text{otherwise.} \end{cases}$$

Then for $i = 1, \dots, i_{max}$, $\mathbb{E}[|E_i|] = \mathbb{E}[|V_i|] \cdot O(f^E(i))$.

Proof. Take a vertex $v \in V_i$. All neighbours of v in $G(i)$ belong to the intersection of the d-dimensional ball of radius r centered at v, and the bounding box $\mathrm{BB}(i)$. This intersection, on one hand, is contained in a box of size $2r \times 2R \times \cdots \times 2R$, where $R = \frac{1}{2}\sqrt{(W_{\mathrm{ub}}(i))^2 - w^2}$ (see (8)). On the other hand, it is trivially inside a ball of radius r. In the former case the volume of the box with v's neighbours is

$$O(rR^{d-1}) = O\left(\frac{1}{n}(w/r)^{d-1} \cdot i\right)$$

In the latter case the volume is $O(r^d)$. Therefore, the expected number of neighbours of v is

$$O\left(n \cdot \min\left\{\frac{1}{n}(w/r)^{d-1} \cdot i, r^d\right\}\right) = O(\min\{(w/r)^{d-1} \cdot i, nr^d\}). \tag{9}$$

By linearity of expectation we get the desired bound on $\mathbb{E}[|E_i|]$. ◀

The following lemma describes how to efficiently generate the edges E_i when we use Dijkstra's algorithm (for $d \geq 3$).

▶ **Lemma 10.** *Let f^E be as in Lemma 9. Given V_i, the edge set E_i can be computed in $T^E_{gen}(i) = O(\mathbb{E}[|V_i|] \cdot f^E(i))$ expected time.*

Proof. We divide $[0,1]^d$ into cubes of size $r \times r \times \cdots \times r$. With each non-empty cube we will keep a list of vertices from V_i that belongs to that cube. We build these lists by iterating over all $v \in V_i$ and assign v to the appropriate cube's list. Technically speaking, the lists are stored in a hash table with expected $O(1)$ insertion and access time (see e.g., [9]): note that the cubes can be mapped to integers $[1, (\lceil 1/r \rceil)^d]$ and we have $(\lceil 1/r \rceil)^d = O(n)$ by $r = \Omega\left((\log(n)/n)^{1/d}\right)$. To find the edges, for each v we iterate over all vertices w belonging to the same cube as v or a neighbouring cube and check whether $||v - w|| \leq r$. There are at most 3^d such cubes and each neighbor of v necessarily lies in these neighboring cubes.

Each cube contains $O(n \min\{rR^{d-1}, r^d\})$ vertices in expectation, where we again set $R = \frac{1}{2}\sqrt{(W_{\mathrm{ub}}(i))^2 - w^2}$ (see (8)). Recall from (9) in Lemma 9 that this quantity is $O(f^{\mathrm{E}}(i))$. This is because if $2R < r$ then the cube's intersection with $\mathrm{BB}(i)$ has size at most $r \times (2R) \times \cdots \times (2R)$ and only in that part of the cube the vertices from V_i can appear. Therefore, the expected total work for each vertex will be $O(3^d \cdot f^{\mathrm{E}}(i)) = O(f^{\mathrm{E}}(i))$. Thus, by linearity of expectation, the expected running time is indeed $O(\mathbb{E}[|V_i|] \cdot f^E(i))$. ◀

We are now ready to prove the following theorem bounding the expected running time of the query algorithm.

▶ **Theorem 11.** *The expected running time of the query algorithm on an n-vertex random weighted unit-disk graph in $[0,1]^d$ with connectivity radius r is*
(a) $O\left((w/r)^2 \log^2(1 + w/r) + \sqrt{n}\right)$ *for* $d = 2$,
(b) $O\big((w/r)^{2d-1} + n^{1-1/d}\big)$ *for* $d \geq 3$ *and* $r \geq \sqrt[2d-1]{w^{d-1}/n}$,
(c) $O(nw^d + n^{1-1/d})$ *otherwise.*

Proof. In all cases we will bound the expected query time as given in sum (5):

$$O\left(\sum_{i=1}^{i_{\max}} \bar{P}(i-1)\cdot\mathbb{E}[T_d(i)] + \bar{P}(i_{\max})\cdot n^2\right).$$

First of all, note that by Equation 6, Lemma 5 and the assumption $r \geq (\beta\log(n)/n)^{1/d}$ where $\beta > 1$ is a large enough constant, for some constant $\gamma > 0$ we have:

$$i_{\max} \geq \gamma\cdot nr^d \geq \gamma\cdot\beta\log n$$

So, picking $\beta = 2/\gamma$ gives us

$$\bar{P}(i_{\max})\cdot n^2 = O\left(e^{-i_{\max}}\cdot n^2\right) = O\left(e^{-2\log n}\cdot n^2\right) = O(1).$$

Hence, we can focus on the below sum. By Lemma 5, we have:

$$O\left(\sum_{i=1}^{i_{\max}} \bar{P}(i-1)\cdot\mathbb{E}[T_d(i)]\right) = O\left(\sum_{i=1}^{\infty}\mathbb{E}[T_d(i)]e^{-(i-1)}\right) = O\left(\sum_{i=1}^{\infty}\mathbb{E}[T_d(i)]e^{-i}\right).$$

In the following, we will use the asymptotic formula $\sum_{i=1}^{\infty} f(i)e^{-i} = O(1)$ that holds for any function $f(i) = \text{poly}(i)$. Recall that $w > r$.

Let us first prove item (a). By (3) and Lemma 6, we have:

$$\begin{aligned}
&O\left(\sum_{i=1}^{\infty}\mathbb{E}[T_2(i)]e^{-i}\right)\\
&= O\left(\sum_{i=1}^{\infty}(w/r)^2\cdot i\cdot\log^2\left((w/r)^2 i+2\right)\cdot e^{-i} + \sum_{i=1}^{\infty}\sqrt{n}e^{-i}\right)\\
&= O\left((w/r)^2\log^2(w/r+1)\sum_{i=1}^{\infty} i\log^2(i)\cdot e^{-i} + \sqrt{n}\sum_{i=1}^{\infty}e^{-i}\right)\\
&\qquad\qquad = O\left((w/r)^2\log^2(1+w/r)+\sqrt{n}\right).
\end{aligned}$$

Above we silently used Corollary 8 for $X = |V_i|$ and $\alpha = 2$. Now let us prove items (b) and (c). Let us first argue that the term $\mathbb{E}[|V_i|\log|V_i|]$ is, by Corollary 8, asymptotically dominated by the bound $\mathbb{E}[|V_i|]\cdot O(f^{\mathrm{E}}(i))$ on $\mathbb{E}[|E_i|]$ from Lemma 9. This follows by Lemmas 6 and 9 – if r is sufficiently large. Thus by plugging that bound into (4) we get

$$\begin{aligned}
&O\left(\sum_{i=1}^{\infty}\mathbb{E}[T_d(i)]e^{-i}\right)\\
&= \sum_{i=1}^{\infty}(w/r)^d i\cdot\min\{nr^d,(w/r)^{d-1}i\}e^{-i} + \sum_{i=1}^{\infty}n^{1-1/d}e^{-i}\\
&= O\left(\min\left\{nw^d\sum_{i\geq 1}ie^{-i},(w/r)^{2d-1}\sum_{i=1}^{\infty}i^2e^{-i}\right\} + n^{1-1/d}\sum_{i=1}^{\infty}e^{-i}\right)\\
&\qquad\qquad = O\left(\min\left\{nw^d,(w/r)^{2d-1}\right\}+n^{1-1/d}\right).
\end{aligned}$$ ◀

▶ Remark 12. The described distance oracle can be very easily made dynamic with only polylogarithmic overhead. That is, we can support insertions and deletions of vertices of the weighted unit-disk graph G, in amortized $O(\text{polylog}\, n)$ time. To this end we simply replace the simplex range query data structure of Chan [5] that we build in the preprocessing with that of Matousek [16] which allows for polylogarithmic amortized updates to the point set and has only polylogarithmically slower preprocessing and query times.

4 Channels

The remaining part of the paper is devoted to proving the very convenient bound on $\bar{P}(i)$ from Lemma 5.

We start by introducing a notion of a *channel*, which is a parameterized grid-like object whose goal is to "discretize" the space of possible shortest $s \to t$ paths in $\mathrm{BB}(i)$. The next step is to upper-bound the probability that we fail to find reasonably short $s \to t$ path in the channel. Afterwards, we are ready to give explicit formulas for $i_{\max}$ and $W_{\mathrm{ub}}(i)$ so that the asymptotic bounds (6) and (7), as well as the bound $\bar{P}(i) \le e^{-i}$ hold.

Roughly speaking, a channel is a subset of vertices V restricted to some subspace. We generalize the channels defined in [18, page 41] to d-dimensional space and arbitrary start/end vertices s and t.

Recall that $w = ||t - s||$ and $w > r$. Let $K \ge 1$ be the smallest integer such that $l = w/(4K+1) \le r/4$. We also have

$$l = \frac{w}{4(K-1)+1} \cdot \frac{4(K-1)+1}{4K+1} > r/4 \cdot \frac{4K-3}{4K+1} \ge r/20. \tag{10}$$

We are going to work in the coordinate system introduced in Section 3.2. Let us denote the first axis by x_0 and the remaining axes by $x_1, \ldots, x_{d-1}$.

▶ **Definition 13** (Box $R(z_0, z_1, \ldots, z_{d-1})$). *Let $h > 0$ be fixed. Let us cut the space using planes $x_0 = lz$ and $x_i = (1/2 + z)h$ for all integers z and $i = 1, \ldots, d-1$.*

For $z_0, z_1, \ldots, z_{d-1} \in \mathbb{Z}$, the box *$R(z_0, z_1, \ldots, z_{d-1})$ contains all points $(x_i)_{i=0}^{d-1}$ satisfying:*
- *$lz_0 \le x_0 \le l(z_0+1)$,*
- *$(-1/2 + z_i)h \le x_i \le (1/2 + z_i)h$ for all $i = 1, \ldots, d-1$.*

Each box, defined as above, has size $l \times h \times \cdots \times h$. Note that $s \in R(0, 0, \ldots, 0)$ and $t \in R(4K, 0, \ldots, 0)$. Now suppose we want to travel from the box containing s to the box containing t using *jumps*, defined below.

▶ **Definition 14** (Jumping between boxes). *We say that we can* jump *from box $R(z_0, z_1, \ldots, z_{d-1})$ to box $R(z'_0, z'_1, \ldots, z'_{d-1})$ iff*
- *$z'_0 = z_0 + 2$,*
- *$|z'_i - z_i| = 1$ for all $i = 1, \ldots, d-1$.*

Consider a jumping trip from $R(0, 0, \ldots, 0)$ to $R(4K, 0, \ldots, 0)$.

▶ **Observation 15** (Reachable boxes). *Let $B = R(z_0, z_1, \ldots, z_{d-1})$ be an arbitrary box. Suppose a sequence of jumps (as defined above) from $R(0, 0, \ldots, 0)$ to $R(4K, 0, \ldots, 0)$ goes through the box B. Then, the following conditions hold:*
- *$z_0 = 2k$ for some integer k, $0 \le k \le 2K$,*
- *$|z_i| \le \min(k, 2K - k)$ for all $i = 1, \ldots, d-1$,*
- *$z_i \equiv k \pmod 2$.*

Now we are ready to define the *channel* parameterized by h.

▶ **Definition 16** (Channel). *A* channel *$ch(h)$ is a subset of $[0,1]^d$ defined as the union of all boxes B satisfying the conditions of Observation 15.*

In other words, a channel $\mathrm{ch}(h)$ consists of all boxes that can appear in a sequence of jumps from the box containing s to the box containing t. Boxes, jumps, and channels are depicted in Figure 1.

In the following, we say that a box B is *empty* if it does not contain any vertex of G.

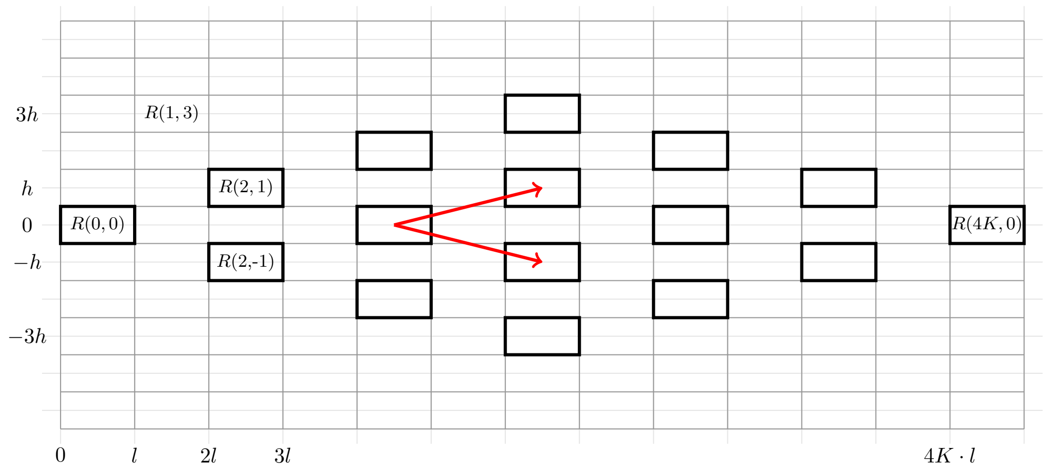

Figure 1 The rectangles represent boxes from Definition 13 for $d = 2$. The red arrows represent possible jumps from a single box. The channel ch(h) for $K = 3$ (see Definition 16) is represented by rectangles with thick black border.

4.1 Paths in a channel

Not all channels ch(h) are of our interest. We need a condition on h guaranteeing that if we can jump from a non-empty box B to another non-empty box B' then there exists an appropriate edge in the graph, namely if there is $u \in B \cap V$ and $v \in B' \cap V$ then $||u - v|| \leq r$. Then, a sequence of jumps between non-empty boxes will certify the existence of a path in G.

Observe that the distance between two opposite corners of B and B' (recall that B and B' have to satisfy Definition 14) is

$$\sqrt{(3l)^2 + (d-1)(2h)^2}.$$

We need this to be smaller than r. Taking into account that $l \leq r/4$, it is sufficient that

$$\left(\frac{3}{4}r\right)^2 + (d-1)(2h)^2 \leq r^2,$$

which gives

$$h \leq \frac{1}{8}\sqrt{\frac{7}{d-1}} \cdot r. \tag{11}$$

▶ **Definition 17** (Path in ch(h))**.** A path *in ch*(h) *with* h *satisfying* (11) *is a sequence of non-empty boxes* $B_0, \ldots, B_{2K}$ *such that* $B_0 = R(0, 0, \ldots, 0)$*,* $B_{2K} = R(4K, 0, \ldots, 0)$*, and we can jump from* B_j *to* B_{j+1} *for all* $j = 0, \ldots, 2K - 1$*.*

Now we show that a path in ch(h) certifies the existence of an $s - t$ path in G which is not too long. Specifically, we show the following bound.

▶ **Lemma 18** (Channel induced path length)**.** *Suppose there is a path in ch*(h)*. Then, there exists an* $s - t$ *path in* G *of length no more than*

$$w\sqrt{1 + 40^2(d-1)(h/r)^2}. \tag{12}$$

Proof. Let $u_j = (u_0^j, \ldots, u_{d-1}^j)$ be a vertex of G in $B_j \cap V$. Additionally, set $u_0 = s$ and $u_{2K} = t$. Recall that u_j exists since each box in a path in $\mathrm{ch}(h)$ is non-empty. Consider subsequent vertices u_j and u_{j+1}. Note that

$$||u_{j+1} - u_j|| = \sqrt{\sum_{i=0}^{d-1} (u_i^{j+1} - u_i^j)^2} = (u_0^{j+1} - u_0^j)\sqrt{1 + \sum_{i=1}^{d-1} \left(\frac{u_i^{j+1} - u_i^j}{u_0^{j+1} - u_0^j}\right)^2}.$$

Recall that we have $u_0^{j+1} - u_0^j \geq l$ and $u_i^{j+1} - u_i^j \leq 2h$ for $i \geq 1$. Hence,

$$||u_{j+1} - u_j|| \leq (u_0^{j+1} - u_0^j)\sqrt{1 + (d-1)\frac{(2h)^2}{l^2}}.$$

Since $u_0 \to u_1 \to \ldots u_{2K}$ is a path in G, the length of a shortest $s-t$ path in G can be bounded by:

$$\begin{aligned}\sum_{j=0}^{2K-1} ||u_{j+1} - u_j|| &\leq \sqrt{1 + (d-1)\frac{(2h)^2}{l^2}} \cdot \sum_{j=0}^{2K-1} (u_0^{j+1} - u_0^j) \\ &= \sqrt{1 + (d-1)\left(\frac{2h}{l}\right)^2} \cdot w.\end{aligned}$$

The claimed bound is obtained by $l \geq r/20$. ◀

4.2 Probability

Denote by q the probability that a single box is empty. We have:

$$q = (1 - lh^{d-1})^n \leq \exp(-nlh^{d-1}). \tag{13}$$

Denote by $\hat{P}(h)$ the probability that no path exists in $\mathrm{ch}(h)$. We are going to prove the following lemma.

▶ **Lemma 19.** *There exists constants $q_0 \in (0,1)$ and $c > 0$ such that if $q < q_0$ then we have*

$$\hat{P}(h) \leq (cq)^{2^{d-3}}. \tag{14}$$

Proof. The proof will proceed by induction on d. We will thus use the notation $\hat{P}_d(h)$ and $\mathrm{ch}_d(h)$ to underline which dimension d we are currently referring to.

The crux of the proof is to prove the induction base $d = 2$, i.e., the bound

$$\hat{P}_2(h) \leq \sqrt{cq}$$

that holds for all $q < q_0$ for some constants c, q_0. Due to space constraints, the proof of this bound can be found in the full version. The general idea behind that proof is to reformulate it in terms of directed reachability in $n \times n$ grids: we want to bound the probability that no path between the corners of the grid exist when each vertex of the grid can fail with probability q. Then next step is to adapt the so-called *contour argument* from oriented percolation theory (see e.g., [10]) to work with finite grids.

For larger d it is enough to prove that the bound

$$\hat{P}_d(h) \leq \left(\hat{P}_{d-1}(h)\right)^2.$$

holds. Let $s \in \{-1, 1\}$. Consider a subchannel $\mathrm{ch}_d^s(h)$ of the channel $\mathrm{ch}_d(h)$ that is composed of the reachable boxes $B = R(z_0, z_1, \ldots, z_{d-1})$ fulfilling the following conditions:

- $z_0 = 2k$ for some integer k, $0 \le k \le 2K$,
- $|z_i| \le \min(k, 2K - k)$ for all $i = 1, \dots, d-2$,
- $z_{d-1} = s \cdot \min(k, 2K - k)$,
- $z_i \equiv k \pmod 2$.

Observe that the above conditions say that B is a reachable box in $\mathrm{ch}_d(h)$ with additional constraint $z_{d-1} = s \cdot \min(k, 2K-k)$, which can also be written as $z_{d-1} = s \cdot \min(z_0, 4K - z_0)/2$.

Now one can see that $\mathrm{ch}_d^s(h)$ has exactly the same structure as $\mathrm{ch}_{d-1}(h)$: we can jump between boxes $R(z_0, \dots, z_{d-2})$ and $R(z_0', \dots, z_{d-2}')$ in channel $\mathrm{ch}_{d-1}(h)$ if and only if we can jump between boxes

$$R(z_0, \dots, z_{d-2}, s \cdot \min(z_0, 4K - z_0)/2)$$

and

$$R(z_0', \dots, z_{d-2}', s \cdot \min(z_0', 4K - z_0')/2)$$

in channel $\mathrm{ch}_d^s(h)$. Therefore the probability that no path exists in $\mathrm{ch}_d^s(h)$ is bounded by $\hat{P}_{d-1}(h)$.

Observe that $\mathrm{ch}_d^{-1}(h)$ and $\mathrm{ch}_d^1(h)$ share only the corner boxes $R(0, 0, \dots, 0)$ and $R(4K, 0, \dots, 0)$. Thus if no path exists in $\mathrm{ch}_d(h)$, there must be no paths in $\mathrm{ch}_d^{-1}(h)$ and $\mathrm{ch}_d^1(h)$ independently. This clearly happens with probability at most $\left(\hat{P}_{d-1}(h)\right)^2$. ◀

5 Choosing the size of i-th bounding box

In this section we show how we derive the bound of Lemma 5 from Lemma 19. We will also be able to explicitly define the value $i_{\max}$ and the function $W_{\mathrm{ub}}(i)$ so that the asymptotic bounds (6) and (7) hold.

Suppose that for a fixed i we pick such h_i that $W_{\mathrm{ub}}(i) = w\sqrt{1 + 40^2(d-1)(h_i/r)^2}$. Then, by Lemma 18, a path in $\mathrm{ch}(h_i)$ certifies the existence of a $s \to t$ path in G of length at most $W_{\mathrm{ub}}(i)$. Such a path is clearly contained in $\mathrm{BE}(i)$, and thus also in $\mathrm{BB}(i)$. As a result, we conclude

$$\bar{P}(i) \le \hat{P}(h_i).$$

Given this, and since we want the probability $\bar{P}(i)$ to decay exponentially with i, we would like to choose h_i in a such way that $\hat{P}(h_i) \le e^{-i}$, which will imply $\bar{P}(i) \le e^{-i}$.

Suppose $\exp(-nlh^{d-1}) < q_0$, where q_0 is the constant of Lemma 19. By combining inequality (13) and the bound of Lemma 19, we have

$$\hat{P}(h) \le \exp\left(2^{d-3}(\log c - nlh^{d-1})\right).$$

In order to guarantee $\hat{P}(h_i) \le e^{-i}$, it is thus enough to have

$$\begin{aligned} 2^{d-3}(\log c - nlh_i^{d-1}) &\le -i \\ \log c + \frac{i}{2^{d-3}} &\le nlh_i^{d-1}, \end{aligned} \tag{15}$$

and

$$\log \frac{2}{q_0} \le nlh_i^{d-1}.$$

Let c' be such a positive constant that for $i \geq 1$ we have

$$\max\left(\log\frac{2}{q_0}, \log c + \frac{i}{2^{d-3}}\right) \leq c' \cdot i. \tag{16}$$

Now let h_0 be such that $h_0^{d-1} = \frac{c'}{nl}$, and let

$$h_i = h_0 \cdot i^{\frac{1}{d-1}}. \tag{17}$$

Then we have

$$\max\left(\log\frac{2}{q_0}, \log c + \frac{i}{2^{d-3}}\right) \leq c' \cdot i = c' \cdot \left(\frac{h_i}{h_0}\right)^{d-1} = c' \cdot h_i^{d-1} \cdot \frac{nl}{c'} = nlh_i^{d-1}.$$

So indeed, if h_i is defined as in (17), we have $\hat{P}(h_i) \leq e^{-i}$. So the explicit formula for $W_{\mathrm{ub}}(i)$ is:

$$W_{\mathrm{ub}}(i) = w\sqrt{1 + 40^2(d-1)\left(\frac{c'i}{nlr^{d-1}}\right)^{\frac{2}{d-1}}},$$

where c' is a constant defined in (16) and $l = \Theta(r)$ is as defined in (10). It is now verified that $W_{\mathrm{ub}}(i)$ indeed satisfies the asymptotic formula (7) from Section 3.

The above proof derivation of $\bar{P}(i) \leq e^{-i}$ is only correct if h_i is not too large. Namely, recall that the bound (11) requires that

$$h_i \leq \frac{1}{8}\sqrt{\frac{7}{d-1}} \cdot r. \tag{18}$$

Since h_i is an increasing function of i, this imposes a constraint on maximum possible $i = i_{\max}$ allowed. Hence, we need to have

$$\left(\frac{c' \cdot i_{\max}}{nl}\right)^{\frac{1}{d-1}} \leq \frac{1}{8}\sqrt{\frac{7}{d-1}} \cdot r.$$

$$i_{\max} = \left\lfloor \frac{1}{c'} \cdot \left(\frac{1}{8}\sqrt{\frac{7}{d-1}} \cdot r\right)^{d-1} \cdot nl \right\rfloor = \Theta(nr^d).$$

Observe that the above definition of $i_{\max}$ agrees with the bound (6) from Section 3.

References

1 Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. Highway dimension and provably efficient shortest path algorithms. *J. ACM*, 63(5), 2016. doi:10.1145/2985473.

2 Ittai Abraham and Cyril Gavoille. Object location using path separators. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing*, PODC '06, page 188–197, New York, NY, USA, 2006. Association for Computing Machinery. doi:10.1145/1146381.1146411.

3 Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F. Werneck. *Route Planning in Transportation Networks*, pages 19–80. Springer International Publishing, Cham, 2016. doi:10.1007/978-3-319-49487-6_2.

4 Sergio Cabello and Miha Jejcic. Shortest paths in intersection graphs of unit disks. *Comput. Geom.*, 48(4):360–367, 2015. doi:10.1016/j.comgeo.2014.12.003.

5 Timothy M. Chan. Optimal partition trees. *Discret. Comput. Geom.*, 47(4):661–690, 2012. doi:10.1007/s00454-012-9410-z.

6 Timothy M. Chan and Dimitrios Skrepetos. All-pairs shortest paths in unit-disk graphs in slightly subquadratic time. In Seok-Hee Hong, editor, *27th International Symposium on Algorithms and Computation, ISAAC 2016, December 12-14, 2016, Sydney, Australia*, volume 64 of *LIPIcs*, pages 24:1–24:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ISAAC.2016.24.

7 Timothy M. Chan and Dimitrios Skrepetos. Approximate shortest paths and distance oracles in weighted unit-disk graphs. *J. Comput. Geom.*, 10(2):3–20, 2019. doi:10.20382/jocg.v10i2a2.

8 Panagiotis Charalampopoulos, Paweł Gawrychowski, Shay Mozes, and Oren Weimann. Almost optimal distance oracles for planar graphs. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 138–151, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3313276.3316316.

9 Martin Dietzfelbinger and Friedhelm Meyer auf der Heide. A new universal class of hash functions and dynamic hashing in real time. In Mike Paterson, editor, *Automata, Languages and Programming, 17th International Colloquium, ICALP90, Warwick University, England, UK, July 16-20, 1990, Proceedings*, volume 443 of *Lecture Notes in Computer Science*, pages 6–19. Springer, 1990. doi:10.1007/BFb0032018.

10 Richard Durrett. Oriented percolation in two dimensions. *The Annals of Probability*, 12(4):999–1040, 1984. URL: http://www.jstor.org/stable/2243349.

11 P. Gupta and P. R. Kumar. Critical power for asymptotic connectivity. In *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No.98CH36171)*, volume 1, pages 1106–1110 vol.1, 1998. doi:10.1109/CDC.1998.760846.

12 P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, 2000. doi:10.1109/18.825799.

13 Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.*, 4(2):100–107, 1968. doi:10.1109/TSSC.1968.300136.

14 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar voronoi diagrams for general distance functions and their algorithmic applications. *Discret. Comput. Geom.*, 64(3):838–904, 2020. doi:10.1007/s00454-020-00243-7.

15 Ken-Ichi Kawarabayashi, Philip N. Klein, and Christian Sommer. Linear-space approximate distance oracles for planar, bounded-genus and minor-free graphs. In *Proceedings of the 38th International Colloquim Conference on Automata, Languages and Programming - Volume Part I*, ICALP'11, page 135–146, Berlin, Heidelberg, 2011. Springer-Verlag.

16 Jirí Matousek. Efficient partition trees. *Discret. Comput. Geom.*, 8:315–334, 1992. doi:10.1007/BF02293051.

17 Mathew D. Penrose. On k-connectivity for a geometric random graph. *Random Structures & Algorithms*, 15(2):145–164, 1999.

18 Robert Sedgewick and Jeffrey Scott Vitter. Shortest paths in euclidean graphs. *Algorithmica*, 1(1):31–48, 1986. doi:10.1007/BF01840435.

19 Mikkel Thorup and Uri Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, 2005. doi:10.1145/1044731.1044732.

20 Haitao Wang and Jie Xue. Near-optimal algorithms for shortest paths in weighted unit-disk graphs. *Discret. Comput. Geom.*, 64(4):1141–1166, 2020. doi:10.1007/s00454-020-00219-7.