Dynamic Connectivity in Disk Graphs

Haim Kaplan
School of Computer Science,
Tel Aviv University, Israel

Alexander Kauer
Institut für Informatik,
Freie Universtiät Berlin, Germany

Katharina Klost
Institut für Informatik,
Freie Universität Berlin, Germany

Kristin Knorr
Institut für Informatik,
Freie Universität Berlin, Germany

Wolfgang Mulzer
Institut für Informatik,
Freie Universität Berlin, Germany

Liam Roditty
Department of Computer Science,
Bar Ilan University, Ramat Gan, Israel

Paul Seiferth
Institut für Informatik,
Freie Universtiät Berlin, Germany

Abstract

Let $S \subseteq \mathbb{R}^2$ be a set of n planar *sites*, such that each $s \in S$ has an *associated radius* $r_s > 0$. Let $\mathcal{D}(S)$ be the *disk intersection graph* for S. It has vertex set S and an edge between two distinct sites $s, t \in S$ if and only if the disks with centers s, t and radii r_s, r_t intersect. Our goal is to design data structures that maintain the connectivity structure of $\mathcal{D}(S)$ as sites are inserted and/or deleted.

First, we consider *unit disk graphs*, i.e., $r_s = 1$, for all $s \in S$. We describe a data structure that has $O(\log^2 n)$ amortized update and $O(\log n / \log\log n)$ amortized query time. Second, we look at disk graphs *with bounded radius ratio* Ψ, i.e., for all $s \in S$, we have $1 \le r_s \le \Psi$, for a $\Psi \ge 1$ known in advance. In the fully dynamic case, we achieve amortized update time $O(\Psi\lambda_6(\log n)\log^7 n)$ and query time $O(\log n/\log\log n)$, where $\lambda_s(n)$ is the maximum length of a Davenport-Schinzel sequence of order s on n symbols. In the incremental case, where only insertions are allowed, we get logarithmic dependency on Ψ, with $O(\alpha(n))$ query time and $O(\log\Psi\lambda_6(\log n)\log^7 n)$ update time. For the decremental setting, where only deletions are allowed, we first develop an efficient *disk revealing* structure: given two sets R and B of disks, we can delete disks from R, and upon each deletion, we receive a list of all disks in B that no longer intersect the union of R. Using this, we get decremental data structures with amortized query time $O(\log n/\log\log n)$ that support m deletions in $O((n\log^5 n + m\log^7 n)\lambda_6(\log n) + n\log\Psi\log^4 n)$ overall time for bounded radius ratio Ψ and $O((n\log^6 n + m\log^8 n)\lambda_6(\log n))$ for arbitrary radii.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Data structures design and analysis; Mathematics of computing → Paths and connectivity problems

Keywords and phrases Disk Graphs, Connectivity, Lower Envelopes

Digital Object Identifier 10.4230/LIPIcs.SoCG.2022.49

Related Version *Full Version*: `https://arxiv.org/abs/2106.14935`

Funding Supported in part by grant 1367/2016 from the German-Israeli Science Foundation (GIF).
Haim Kaplan: Partially supported by ISF grant 1595/19 and by the Blavatnik research foundation.
Alexander Kauer: Supported in part by grant 1367/2016 from the German-Israeli Science Foundation (GIF), by the German Research Foundation within the collaborative DACH project *Arrangements and Drawings* as DFG Project MU 3501/3-1, and by ERC StG 757609.
Kristin Knorr: Supported by the German Science Foundation within the research training group "Facets of Complexity" (GRK 2434).
Wolfgang Mulzer: Supported in part by ERC StG 757609.

1 Introduction

Suppose we are given a simple, undirected graph G, and we would like to preprocess it so that we can determine efficiently if two vertices of G lie in the same connected component. If G is fixed, we can simply perform a graph search in G (e.g., BFS or DFS) to label the vertices of each connected component with a unique identifier, allowing us to answer all queries in $O(1)$ time with linear preprocessing time and space. When G changes over time, the problem becomes much harder. If the vertex set is fixed and edges can only be inserted, the problem reduces to disjoint set union. Then, there is a folklore optimal data structure.It achieves $O(1)$ time for updates and $O(\alpha(n))$ amortized time for queries, where $\alpha(n)$ is the inverse Ackermann function [5]. If the vertex set is fixed, but edges can be inserted and deleted, there is a data structure due to Holm et al. [8], with $O(\log n/\log\log n)$ amortized query time and $O(\log^2 n)$ amortized update time. For planar graphs, Eppstein et al. [6] give a structure with $O(\log n)$ amortized time for queries *and* updates.

In this paper, we add a geometric twist and study the dynamic connectivity problem on different variants of *disk intersection graphs*. Let $S \subset \mathbb{R}^2$ be a set of planar *point sites*, where each site $s \in S$ has an associated radius $r_s > 0$. The *disk intersection graph* (*disk graph*, for short) $\mathcal{D}(S)$ is the undirected graph with vertex set S that has an undirected edge between any two distinct sites s and t if and only if the Euclidean distance between s and t is at most $r_s + r_t$. Note that even though $\mathcal{D}(S)$ is fully described by the n sites and their associated radii, it might have $\Theta(n^2)$ edges. Thus, our goal is to find algorithms whose running time depends only on the number of sites and not on the number of edges. We consider three variants of disk graphs, characterized by the possible values for the radii. In the first variant, *unit disk graphs*, all radii are 1. In the second variant, *bounded radius ratio*, all radii must come from the interval $[1, \Psi]$, where Ψ is a parameter known in advance that may depend on the number of sites n. In the third variant, *general disk graphs*, the radii can be arbitrary.

We assume that S is dynamic, i.e., sites can be inserted and deleted over time. At each update, the edges incident to the modified site appear or disappear in $\mathcal{D}(S)$. An update can change up to $n-1$ edges in $\mathcal{D}(S)$, so simply storing $\mathcal{D}(S)$ in the data structure by Holm et al. could lead to potentially superlinear update times and might even be slower than recomputing the connectivity information from scratch.

For dynamic connectivity in general disk graphs, Chan et al. [4] give a data structure with amortized $O(n^{1/7+\varepsilon})$ query time and $O(n^{20/21+\varepsilon})$ update time. As far as we know, this is still the currently best fully dynamic connectivity structure for general disk graphs. However, Chan et al. present their data structure as a special case of a more general setting, so there is hope that the specific geometry of disk graphs may allow for better running times.

Indeed, several results show that for certain disk graphs, we can achieve polylogarithmic update and query times. For unit disk graphs, Chan et al. [4] observe that there is a data structure with $O(\log^6 n)$ update time and $O(\log n/\log\log n)$ query time.[1] For bounded radius ratio, Kaplan et al. [9] show that there is a data structure with expected amortized update time $O(\Psi^2 \lambda_6(\log n)\log^7 n)$ and query time $O(\log n/\log\log n)$.[2] Both results use the notion of a *proxy graph*, a sparse graph that models the connectivity of the original disk graph and that can be updated efficiently with suitable dynamic geometric data structures. The proxy graph can then be stored in the data structure by Holm et al., so the query procedure coincides with the one by Holm et al. The update operations involve a combination of updating the proxy graph with the help of the geometric data structures and of modifying the edges in the structure of Holm et al.

[1] Actually, Chan et al. [4] claim an update time of $O(\log^{10} n)$. Recent results [3] improve the bound.

[2] The original paper claims an update time of $O(\Psi^2 \lambda_6(\log n)\log^9 n)$, but recent improvements in the underlying data structure [10] lead to the better bound.

Our results. For unit disk graphs, we significantly improve over Chan et al. [4]: with a direct approach that uses a grid-based proxy graph and dynamic lower envelopes, we obtain $O(\log^2 n)$ amortized update and $O(\log n/\log\log n)$ amortized query time (Theorem 3.2).

For bounded radius ratio, we give a data structure that improves the update time. Specifically, we achieve expected amortized update time $O(\Psi\lambda_6(\log n)\log^7 n)$ and amortized query time $O(\log n/\log\log n)$, where $\lambda_s(n)$ is the maximum length of a Davenport-Schinzel sequence of order s on n symbols [11]. Compared to the previous data structure of Kaplan et al., this improves the factor in the update time from Ψ^2 to Ψ.

We also provide partial results that push the dependency on Ψ from linear to logarithmic. For this, we consider the *semi-dynamic* setting, in which only insertions (*incremental*) or only deletions (*decremental*) are allowed. In the incremental setting, we use a dynamic additively weighted Voronoi diagram to obtain a data structure with $O(\alpha(n))$ amortized query time and $O(\log\Psi\lambda_6(\log n)\log^7 n)$ expected amortized update time. Due to space reasons, this result is deferred to the full version. In the decremental setting, a main challenge is to identify those edges in $\mathcal{D}(S)$ that were incident to a freshly removed site and that change the connectivity in $\mathcal{D}(S)$. To address this, we first develop a data structure for a related dynamic geometric problem which might be of independent interest: suppose we have two sets R and B of disks in the plane, such that the disks in B can only be deleted, while the disks in R can be both inserted and deleted. We would like to maintain R and B in a data structure such that whenever we delete a disk b from B, we receive a list of all the disks in the current set R that intersect the disk b but no other disk from the remaining set $B\setminus\{b\}$. We say that these are the disks in R that are *revealed* by the deletion of b. We call this data structure a *disk revealing structure* (RDS). Due to space reasons, the details of the RDS are relegated to the full version. Its properties are summarized in the following theorem:

▶ **Theorem 1.1.** *Let R and B be disjoint sets of disks in $\mathbb{R}^2$ with $|R|+|B|=n$. We can preprocess $R\cup B$ into a structure that supports deletions from $R\cup B$, while detecting all newly revealed disks of R after each deletion from b. Preprocessing needs $O\big(|B|\log^5 n\lambda_6(\log n)+|R|\log^3 n\big)$ expected time and $O(n\log n)$ expected space. Deleting k disks from B and any number of disks from R needs $O\big(|R|\log^4 n+k\log^7 n\lambda_6(\log n)\big)$ expected time, where $\lambda_s(n)$ is the maximum length of a Davenport-Schinzel sequence of order s.*

The RDS plays a crucial part in developing decremental connectivity structures for disk graphs of bounded radius ratio and for general disk graphs. For both cases, we obtain data structures with $O(\log n/\log\log n)$ amortized query time. The total expected time for processing k deletions is $O((n\log^5 n+k\log^7 n)\lambda_6(\log n)+n\log\Psi\log^4 n)$ for bounded radius ratio (Theorem 5.6) and $O((n\log^6 n+k\log^8 n)\lambda_6(\log n))$ for the general case (full version).

2 Preliminaries

Data structure for edge updates. We rely on the following existing data structure that supports connectivity queries and edge updates on general graphs.

▶ **Theorem 2.1** (Holm et al. [8, Theorem 3]). *Let G be a graph with n vertices and initially no edges. There is a deterministic fully dynamic data structure so that edge updates in G take amortized time $O(\log^2 n)$ and connectivity queries take worst-case time $O(\log n/\log\log n)$.*

Theorem 2.1 assumes that n is fixed, but we can easily support vertex insertions and deletions within the same amortized time bounds, with standard rebuilding. Thorup gave a variant of Theorem 2.1 that uses $O(m)$ space, where m is the current number of edges [13].

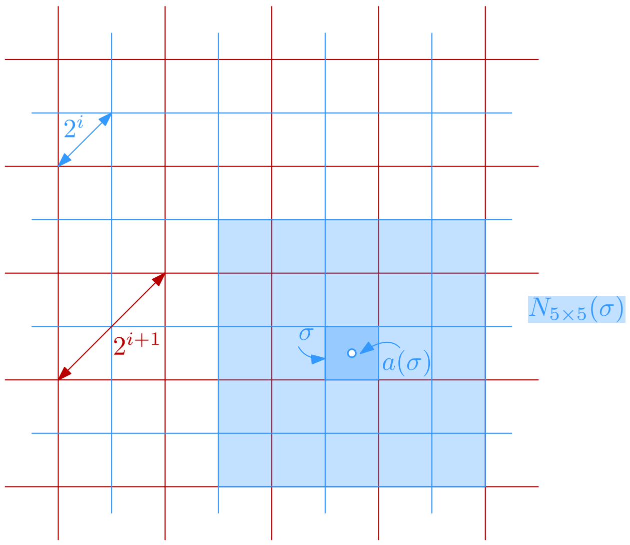


Figure 1 Two levels of the hierarchical grid.

The hierarchical grid and quadtrees. Let $\mathcal{G}_i$ be a grid with cell diameter 2^i and a corner at the origin. The *hierarchical grid* $\mathcal{G}$ is defined as $\mathcal{G} = \bigcup_{i=0}^{\infty} \mathcal{G}_i$. For any cell $\sigma \in \mathcal{G}$, we denote by $|\sigma|$ its diameter and by $a(\sigma)$ its center. We say that grid $\mathcal{G}_i$ has *level* i. We assume that we can find the coordinates of the cell of $\mathcal{G}$ containing a site on a given level in $O(1)$ time. Furthermore, for a cell $\sigma \in \mathcal{G}_i$ and odd k, we call the $k \times k$ subgrid of $\mathcal{G}_i$ centered at σ the $(k \times k)$*-neighborhood* of σ, and denote it by $N_{k\times k}(\sigma)$; see Figure 1. Let $\mathcal{C}$ be a set of cells in $\mathcal{G}$. The *quadtree* $\mathcal{T}$ for $\mathcal{C}$ is a rooted 4-nary tree whose nodes are cells from $\mathcal{G}$. The root of $\mathcal{C}$ is the smallest cell ρ in $\mathcal{G}$ that contains all of $\mathcal{C}$. If a cell σ with $|\sigma| = 2^i$, for $i \geq 1$, properly contains at least one cell of $\mathcal{C}$, then the four children of σ are the cells τ with $|\tau| = 2^{i-1}$ and $\tau \subseteq \sigma$. If a cell σ does not properly contain a cell of $\mathcal{C}$, it does not have any children. Typically, we do not distinguish between a cell σ and its associated vertex. A quadtree $\mathcal{T}$ on a given set of n cells can be constructed in $O(n \log(|\rho|))$ time, where ρ is the root of $\mathcal{T}$.

Maximal bichromatic matchings. We need a data structure that dynamically maintains a *maximal bichromatic matching* (MBM) between two sets of disks: let $R \subseteq S$ and $B \subseteq S$ be two disjoint non-empty sets of sites, and $(R \times B) \cap \mathcal{D}(S)$ the bipartite graph on R and B with all edges of $\mathcal{D}(S)$ with one vertex in R and one vertex in B. An MBM between R and B is a maximal set of vertex-disjoint edges in $(R \times B) \cap \mathcal{D}(S)$. We show how to maintain an MBM as sites are inserted or deleted in R and in B, in two ways. The first way uses a general structure by Kaplan et al. [9] and applies in all settings, see the full version for details.

▶ **Lemma 2.2.** *Let $R, B \subseteq S$ be two disjoint sets with a total of at most n sites. Then, there exists a dynamic data structure that maintains an MBM for R and B with $O(\lambda_6(\log n) \log^7 n)$ amortized expected update time, using $O(n \log n)$ expected space.*

The second way applies only to unit disks that are separated by a vertical or horizontal lines. It relies on dynamic lower envelopes for pseudolines [1], see the full version for details.

▶ **Lemma 2.3.** *Suppose that $r_s = 1$, for all sites $s \in S$. Let $R, B \subseteq S$ be two disjoint sets with a total of at most n sites, such that there is a there exists a known vertical or horizontal line that separates R and B. Then, there exists a dynamic data structure that maintains an MBM for R and B with $O(\log^2 n)$ worst-case update time, using $O(n)$ space.*

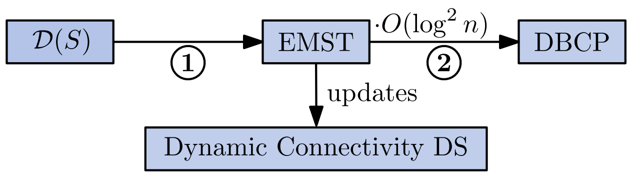


Figure 2 A solution with $O(\log^6 n)$ update time.

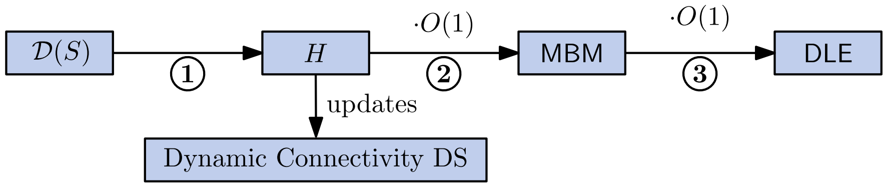


Figure 3 The structure for our data structure.

3 Fully dynamic unit disk graphs

We first consider the case of unit disk graphs. As mentioned in the introduction, this problem was already addressed by Chan et al. [4]. They explained how to combine several known results into a data structure for connectivity queries in fully dynamic unit disk graphs with update time $O(\log^6 n)$ and query time $O(\log n/\log\log n)$.

A visual representation of their approach can be found in Figure 2. In the core, they use a subtree of the Euclidean minimum spanning tree (EMST) of S as a *proxy graph* that accurately represents the connectivity of $\mathcal{D}(S)$. They store this proxy graph in a Holm et al. data structure. In order to update the EMST efficiently, they also maintain several instances of a *dynamic bichromatic closest pair* problem (DBCP). The combination of the running times for the separated data structures then yields the overall running time claimed above. To improve over this result, we replace the EMST by a simpler graph that still captures the connectivity of $\mathcal{D}(S)$. We also replace the DBCP structure by a suitable maximal bichromatic matching (MBM) structure that is based on dynamic lower envelopes (Lemma 2.3). These two changes significantly improve the amortized update time to $O(\log^2 n)$, without affecting the query time. The overall structure behind our method is shown in Figure 3.

We define a *proxy graph* H that represents the connectivity of $\mathcal{D}(S)$. The vertices of H are cells of the grid $\mathcal{G}_1$ of diameter 2 (cf. Section 2). More precisely, we say that two cells σ, τ in $\mathcal{G}_1$ are *neighboring* if $\sigma \in N_{5\times 5}(\tau)$. For $S \subset \mathbb{R}^2$, we define the graph H whose vertices are the *non-empty* cells $\sigma \in \mathcal{G}_1$, i.e., the cells with $\sigma \cap S \neq \emptyset$. We say that a site $s \in S$ is *assigned* to the cell $\sigma \in \mathcal{G}_1$ that contains it, and we let $S(\sigma)$ denote the sites that are assigned to σ. Two cells σ, τ are connected by an edge in H if and only if there is an edge $st \in \mathcal{D}(S)$ with $s \in S(\sigma)$ and $t \in S(\tau)$. Then, H is sparse and represents the connectivity in $\mathcal{D}(S)$, as stated in the following lemma. Its simple proof can be found in the full version.

▶ **Lemma 3.1.** *The proxy graph H has at most n vertices, each with degree $O(1)$. Two sites $s, t \in S$ are connected in $\mathcal{D}(S)$ if and only if their assigned cells σ and τ are connected in H.*

We build a data structure $\mathcal{H}$ as in Theorem 2.1 for H. To query the connectivity between two sites s and t, we first identify the cells σ and τ in $\mathcal{G}_1$ to which s and t are assigned. This requires $O(1)$ time, because when inserting a site u, we can store the assigned cell for u in the satellite data of u. The query is then performed on $\mathcal{H}$, using σ and τ as the query vertices. When a site s is inserted into or deleted in S, only the edges incident to the assigned cell σ are affected. By Lemma 3.1, there are only $O(1)$ such edges. Thus, once the set E of these edges is determined, by Theorem 2.1, we can update $\mathcal{H}$ in time $O(\log^2 n)$.

It remains to find the edges E of H that change when we update S. For each pair σ, τ of neighboring cells in $\mathcal{G}_1$, we maintain a maximal bichromatic matching (MBM) $M_{\{\sigma,\tau\}}$ for $R = S(\sigma)$ and $B = S(\tau)$, as in Lemma 2.3 (note that the special requirements of the lemma are met in our case). By construction, there is an edge between σ and τ in H if and only if $M_{\{\sigma,\tau\}}$ is not empty. When inserting or deleting a site s from S, we proceed as follows: let $\sigma \in \mathcal{G}_1$ be the cell associated to σ. We go through all cells $\tau \in N_{5\times 5}(\sigma)$, and we update $M_{\{\sigma,\tau\}}$ by inserting or deleting s from the relevant set. If $M_{\{\sigma,\tau\}}$ becomes non-empty during an insertion or empty during a deletion, we add the edge $\sigma\tau$ to E and mark it for insertion or deletion, respectively. Putting everything together, we obtain the main result of this section:

▶ **Theorem 3.2.** *There is a dynamic connectivity structure for unit disk graphs such that an update takes amortized time $O\big(\log^2 n\big)$ and a connectivity query takes worst-case time $O(\log n/\log\log n)$, where n is the maximum number of sites. The structure uses $O(n)$ space.*

4 Fully dynamic bounded radius ratio

We extend our structure from Theorem 3.2 to the case of bounded radius ratio Ψ. Now, the running times will depend polynomially on Ψ. The general approach is unchanged, but the varying sizes of the disks introduce new issues. First, we adapt Theorem 3.2 to disks of different sizes. Instead of just $\mathcal{G}_1$, we rely on a hierarchical grid with $\lceil \log \Psi \rceil + 1$ levels. Each site s is assigned to a cell σ of such level that $|\sigma| \le r_s < 2|\sigma|$. Since the disks have different sizes, we can no longer use Lemma 2.3 to maintain the maximal bichromatic matchings (MBMs) between neighboring non-empty grid cells. Instead, we use the more complex structure from Lemma 2.2. This increases the overhead for updating the MBM for each pair of neighboring cells. Furthermore, a disk can now intersect disks from $\Theta(\Psi^2)$ other cells, instead of the $O(1)$-bound from the unit disk case, see Figure 4. Thus, the degree of the proxy graph and the number of edges that need to be modified in a single update becomes much larger. This results in the following theorem, see the full version for details.

▶ **Theorem 4.1.** *There is a dynamic connectivity structure for disk graphs of bounded radius ratio Ψ such that an update takes amortized expected time $O(\Psi^2 \lambda_6(\log n)\log^7 n)$ and a connectivity query takes worst-case time $O(\log n/\log\log n)$, where n is the maximum number of sites at any time. The data structure requires $O(\Psi^2 n \log n)$ expected space.*

To remedy this latter problem – at least partially – we describe in Section 4.1 how to refine the proxy graph so that fewer edges need to be modified in a single update operation. This will reduce the dependence on Ψ in the update time to linear. The query procedure becomes slightly more complicated, but the asymptotic running time remains unchanged.

Note that the approach described above is similar to the method of Kaplan et al. [9, Theorem 9.11] that achieves the same time and space bounds. However, the details of our implementation are crucial for the adaptation in Section 4.1. Most significantly, our implementation uses a hierarchical grid instead of a single fine grid.

4.1 Improving the dependence on Ψ

To avoid an update time dependent on the potentially quadratic number of neighbors, we show how to reduce the degree of the proxy graph H from $\Theta(\Psi^2)$ to $O(\Psi)$. The intuition is that to maintain the connected components of $\mathcal{D}(S)$, it suffices to focus on *maximal* disks that are not contained in any other disk in S. From this, it follows that we only need to consider edges between disks that intersect properly. When we want to perform a connectivity query between sites s and t, we must find appropriate maximal disks that contain s and t. Let D

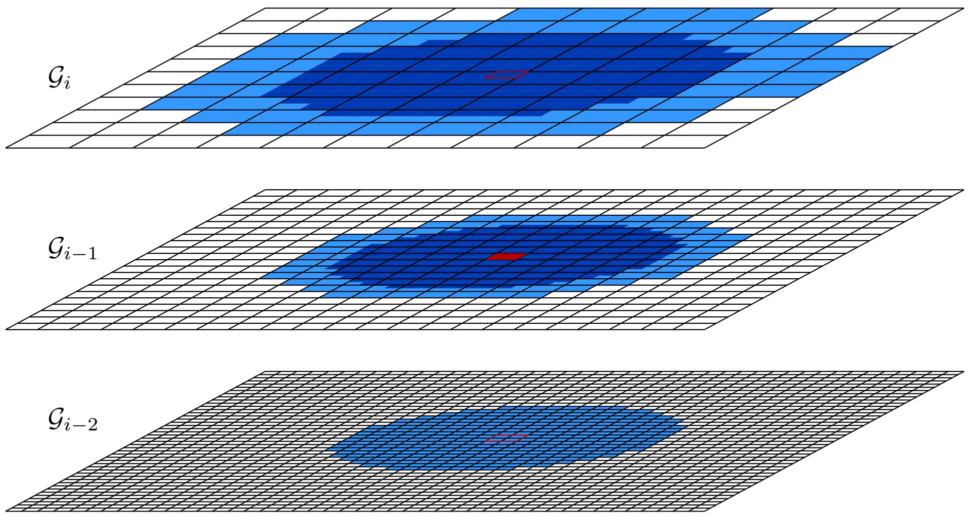


Figure 4 The neighborhood of the red colored cell in $\mathcal{G}_{i-1}$. The area of the neighboring cells in one level beneath is colored in a darker shade.

be a disk and $\sigma \in \mathcal{G}$ a cell. We say that σ is *fully covered* by D if and only if every possible assigned disk of σ is fully contained in D. We call σ *maximal* if and only if there is no larger cell $\tau \supset \sigma$ that is fully covered by D_s.

Given a disk D, the maximal cells in the quadforest $\mathcal{F}$ that are fully covered by D are exactly those that are closest to the root in their quadtrees. Furthermore, the whole subtree of $\mathcal{F}$ that is rooted in a maximal fully covered cell consists of cells that are fully covered by D. The following lemma bounds the number of the different types of cells. See Figure 5.

▶ **Lemma 4.2.** *Let $s \in S$ be a site, and let $\mathcal{N}$ be the cells of $\mathcal{F}$ that may contain a disk that intersects D_s. Write $\mathcal{N} = \mathcal{N}_1 \cup \mathcal{N}_2 \cup \mathcal{N}_3$, where $\mathcal{N}_1$ are the cells that are not fully covered by D_s, $\mathcal{N}_2$ the disks that are maximal fully covered by D_s, and $\mathcal{N}_3$ the disks that are fully covered by D_s, but not maximal with this property. Then, we have $|\mathcal{N}_1 \cup \mathcal{N}_2| = O(\Psi)$ and $|\mathcal{N}_3| = O(\Psi^2)$. Using the quadforest $\mathcal{F}$, we can find $\mathcal{N}_1 \cup \mathcal{N}_2$ in $O(\Psi + \log n)$ time and $\mathcal{N}_3$ in $O(\Psi^2 + \log n)$ time.*

Proof sketch. (Full proof in the full version) The cells of $\mathcal{N}_1$ form an annulus per level. A volume argument shows that they sum up to $O(\Psi)$ altogether. Now, note that every cell in $\mathcal{N}_2$ is either a quadtree root in $\mathcal{F}$ or a child of a cell in $\mathcal{N}_1$. Hence, we have $|\mathcal{N}_1 \cup \mathcal{N}_2| = O(\Psi)$. The bound on $|\mathcal{N}_3|$ follows from the number of neighbors. The retrieval is possible with simple traversal after finding the relevant roots of $\mathcal{F}$ in $O(\log n)$ time. ◀

Now, we show that it is enough to focus on a subset of the edges in the proxy graph. More precisely, let H' be the subgraph of H that is defined as follows: as in H, the vertices of H' are all cells σ that have $S(\sigma) \neq \emptyset$. Two cells σ, τ with $|\sigma| \geq |\tau|$ are adjacent in H' if and only if there are $s \in S(\sigma)$ and $t \in S(\tau)$ such that D_s and D_t intersect *and* such that D_s does *not* fully cover τ. Let σ be a cell in H. We define the *proxy cell* σ' of σ as follows: if there is no disk in S that fully covers σ, then $\sigma' = \sigma$. Otherwise, let $\overline{\sigma} \supseteq \sigma$ be the maximal cell that contains σ and is fully covered by a disk in S, and let D_s, $s \in S$, be a disk of maximum radius that fully covers $\overline{\sigma}$. Then, we set σ' to be the cell with $s \in S(\sigma')$. If there are multiple such disks, the choice is arbitrary.

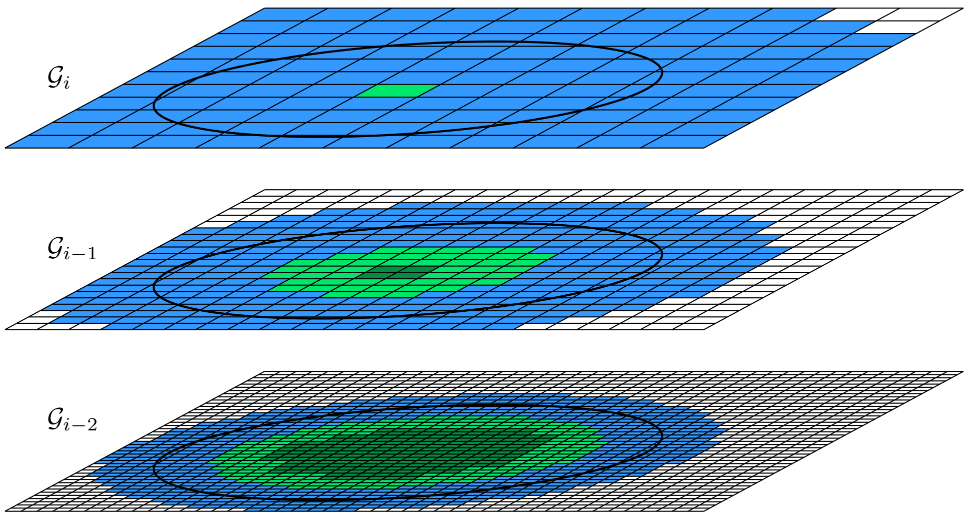
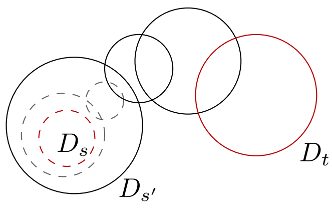
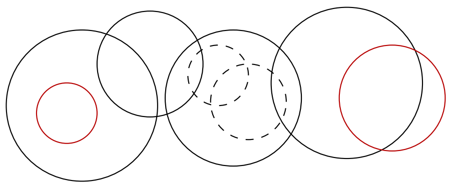


Figure 5 The types of cells that require checking when updating the black disk in Theorem 4.1:
🟦 $\mathcal{N}_1$: not fully covered 🟩 $\mathcal{N}_2$: maximal fully covered 🟩 $\mathcal{N}_3$: fully covered, not maximal.



(a) Omitting the dashed disks and querying for s' instead of s still leads to a valid path to t.



(b) A path between the two red disks can ignore the dashed black disks as intermediates.

Figure 6 Depiction of the arguments in Lemma 4.3.

▶ **Lemma 4.3.** *Let $s, t \in S$ be two sites, and let σ, τ be the cells with $s \in S(\sigma)$ and $t \in S(\tau)$. Let σ' and τ' be the proxy cells for σ and τ. Then, σ' and τ' are connected in H' if and only if s and t are connected in $\mathcal{D}(S)$.*

Proof sketch. (Full proof in the full version) First, suppose that s and t are not connected in $\mathcal{D}(S)$. Since H' is a subgraph of H, it follows that σ' and τ' are not connected in H'.

Next, suppose that s and t are connected in $\mathcal{D}(S)$. We consider a path of *(inclusion) maximal* disks that connects s and t in $\mathcal{D}(S)$, and we show that it induces a path between σ' and τ' in H'. Let $D_{s'}, D_{t'}$ with $s' \in S(\sigma')$, $t' \in S(\tau')$ be the disks of maximum radius which caused σ', τ' to be proxy cells of σ, τ. Now, there is a path π in $\mathcal{D}(S)$ between s' and t' that uses only maximal disks: indeed, along any path in $\mathcal{D}(S)$ between s' and t', we can replace every disk by a maximal disk that contains it, and the resulting path π (possibly after removing duplicate disks) has the required property. See Figure 6. Consider the sequence π' of cells in H' that we obtain by replacing every site u in π by the cell σ_u in H' with $u \in S(\sigma_u)$, and by removing any duplicate cells. We observe that π' is actually a path in H', since the assigned cells for two intersecting maximal disks of S must be adjacent in H'. ◀

Now, our strategy is to maintain the proxy graph H' instead of the graph H, again such that each potential edge of H' is supported by an MBM structure. This will make the updates faster. However, when performing a query, we must be able to find the proxy cells for the query sites efficiently. This requires a further modification of the quadforest $\mathcal{F}$.

▶ **Theorem 4.4.** *There is a data structure for dynamic disk connectivity with expected amortized update time $O(\Psi\lambda_6(\log n)\log^7 n)$ and amortized query time $O(\log n/\log\log n)$. It needs $O(\Psi n\log n)$ expected space.*

Proof sketch. (Full proof in the full version) We may assume that $\Psi = O(n^3)$. We augment the quadforest $\mathcal{F}$: in each cell σ in $\mathcal{F}$, we store the set $\mathcal{C}_\sigma$ of all sites $s \in S$ such that σ is maximal fully covered by D_s. $\mathcal{C}_\sigma$ is organized as a max-heap, ordered by radius r_s.

We describe how to insert a new site s. First, we insert s into the quadforest $\mathcal{F}$. Then, we obtain the sets $\mathcal{N}_1$ and $\mathcal{N}_2$ for s using Lemma 4.2 and insert them into $\mathcal{F}$. For each $\tau \in \mathcal{N}_1$ we insert s into the MBM for σ and τ and update $\mathcal{H}$ of Theorem 2.1 accordingly. For each $\tau \in \mathcal{N}_2$, we insert s into the max-heap $\mathcal{C}_\tau$. A deletion is handled analogously.

To perform a connectivity query between s and t, let σ and τ be the cells with $s \in S(\sigma)$ and $t \in S(\tau)$. We determine the proxy cell σ' via obtaining the maximal cell $\overline{\sigma} \supseteq \sigma$ that contains σ and is fully covered by a disk from S. Let u be the site of maximum radius in the max-heap $\mathcal{C}_{\overline{\sigma}}$ and set $\sigma' = \sigma_u$. τ' is obtained similarly. Afterwards, $\mathcal{H}$ is queried with σ' and τ' for the final result. By Lemma 4.3, this gives the correct answer. The overall running time for this query procedure is $O(\log n)$, where the bottleneck consists in ascending the quadtree.

The query time can be improved via maintaining every $\Theta(\log\log\Psi)$ levels shortcuts pointing upwards, each pointing to the next. To decide whether to take a shortcut, the respective cells have another max-heap containing all intermediate max-heaps. ◀

5 Semi-dynamic bounded radius ratio

We turn to the semi-dynamic setting, and we show how to reduce the dependency on Ψ from linear to logarithmic. For both the incremental and the decremental scenario, we use the same proxy graph H to represent the connectivity in $\mathcal{D}(S)$. The proxy graph is described in Section 5.1. In Section 5.2 we then describe the data structures using H. For details on how to use the proxy graph in the incremental setting, refer to the full version of this paper.

5.1 The proxy graph

The vertex set of the proxy graph H contains one vertex for each site in S, plus one additional vertex per certain region $A \in \mathcal{A}$ in the plane, to be described below. Each region is defined based on a cell of a quadtree and associated with two site sets, $S_1(A)$ and $S_2(A)$. The first set $S_1(A) \subseteq S$ is defined such that all sites $s \in S_1(A)$ lie in A and have a radius r_s comparable to the size of A, for a notion of "comparable" to be detailed below. A site s can lie in several sets $S_1(A)$. We will ensure that for each region A, the induced disk graph $\mathcal{D}(S_1(A))$ of the associated sites is a clique. The second set $S_2(A) \subseteq S$ contains a site s if it lies in the cell associated to the region A and if r_s is "small". The sites in $S_2(A)$ are all sites with a suitable radius in the associated cell of A that have an edge in $\mathcal{D}(S)$ to at least one site in $S_1(A)$.

The proxy graph H is bipartite, with all edges going between the *site-vertices* and the *region-vertices*. The edges of H connect every region A to all sites in $S_1(A) \cup S_2(A)$. The connections between the sites in $S_1(A)$ and A constitute a sparse representation of the corresponding clique $\mathcal{D}(S_1(A))$. The edges connecting a site in $S_2(A)$ to A allow us to represent all edges in $\mathcal{D}(S)$ between $S_2(A)$ and $S_1(A)$ by two edges in H, and since $\mathcal{D}(S_1(A))$ is a clique, this sparse representation does not change the connectivity between the sites. We will see that the sites in $S_2(A)$ can be chosen such that every edge in $\mathcal{D}(S)$ is represented by two edges in H. Furthermore, we will ensure that the number of regions, and the total size of the associated sets $S_1(A)$ and $S_2(A)$ is small, giving a sparse proxy graph.

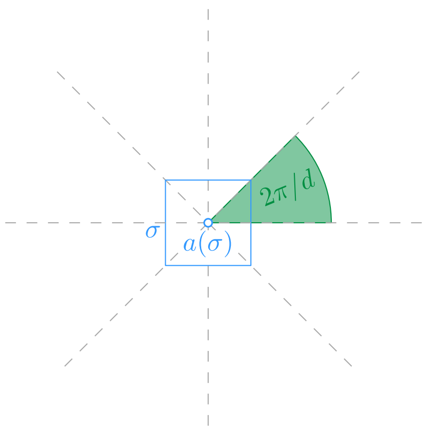


Figure 7 The cones $\mathcal{C}_d$ with angle $2\pi/d$, with apex at the center $a(\sigma)$ of a cell σ.

Now, we describe the details of the regions in $\mathcal{A}$. For each site $s \in S$ we consider the cell $\sigma_s \in \mathcal{G}$ with $s \in \sigma_s$ and $|\sigma_s| \leq r_s < 2|\sigma_s|$ and its (15×15)-neighborhood $N(s)$. We let $\mathcal{N} = \{N(s) \mid s \in S\}$ and construct the quadforest $\mathcal{F}$ for $\mathcal{N}$. This quadforest $\mathcal{F}$ contains quadtrees that cover the lowest $\lfloor \log \Psi \rfloor + 1$ levels of the hierarchical grid $\mathcal{G}$, see the full version for details. The set $\mathcal{A}$ of region-vertices of H is a subset of the set $\mathcal{A}_{\mathcal{F}}$ that contains certain regions for every cell of $\mathcal{F}$. There are three kinds of regions for a cell σ of $\mathcal{F}$: the *outer regions*, the *middle regions*, and the *inner region*.

To describe these regions, we first define for $d \in \mathbb{N}$ a set $\mathcal{C}_d$ of d *cones* with opening angle $2\pi/d$, such that all cones in $\mathcal{C}_d$ have their apex in the origin, have pairwise disjoint interiors, and cover the plane. For a cell $\sigma \in \mathcal{F}$, we denote by $\mathcal{C}_d(\sigma)$ a translated copy of $\mathcal{C}_d$ whose apex has been moved to the center $a(\sigma)$, of σ, as shown in Figure 7.

Let $\Gamma(a, r_1, r_2)$ be the annulus centered at a with inner radius r_1 and outer radius r_2. To define the outer regions for a cell σ, we consider the set $\mathcal{C}_{d_1}(\sigma)$, for some integer parameter d_1 to be determined below. For each $\mathcal{C}_{d_1}$ we set the outer regions to be $\left\{C \cap \Gamma(a(\sigma), \frac{5}{2}|\sigma|, \frac{9}{2}|\sigma|) \mid C \in \mathcal{C}_{d_1}\right\}$. Similarly to this, we define the middle regions as $\left\{C \cap \Gamma(a(\sigma), |\sigma|, \frac{5}{2}) \mid C \in \mathcal{C}_{d_2}\right\}$. Finally, the inner region for σ is the disk with center $a(\sigma)$ and radius $|\sigma|$. See Figure 8 for an illustration of the regions for a cell σ.

We associate a set of sites $S_1(A) \subseteq S$ with each region $A \in \mathcal{A}_{\mathcal{F}}$. The set $S_1(A)$ contains all sites t such that (i) $t \in A$; (ii) $|\sigma| \leq r_t < 2|\sigma|$; and (iii) $\|a(\sigma)t\| \leq r_t + \frac{5}{2}|\sigma|$. This means that the disk D_t has size comparable to $|\sigma|$, a center in A. If t is in a middle or inner region, the third property is trivially true. If t is in an outer region it implies that t intersects the inner boundary of A.

We define $\mathcal{A} \subseteq \mathcal{A}_{\mathcal{F}}$ as the set of regions where $S_1(A) \neq \emptyset$. In the following, we will not strictly distinguish between a vertex from $\mathcal{A}$ and the corresponding region, provided it is clear from the context.

For each region $A \in \mathcal{A}$, we define a set $S_2(A)$ as the set of all sites s such that (i) $s \in \sigma$; (ii) s is adjacent in $\mathcal{D}(S)$ to at least one site in $S_1(A)$; and (iii) $r_s < 2|\sigma|$.

We add an edge sA in H between a site s and a region A if and only if $s \in S_1(A) \cup S_2(A)$. Note that the sets $S_1(A)$ and $S_2(A)$ are not necessarily disjoint, as for the center region defined by a cell σ, a site with $|\sigma| \leq r_s < 2|\sigma|$ will be both in $S_1(A)$ and $S_2(A)$. However, this will adversely affect neither the preprocessing time nor the correctness. The following structural lemma will help us both to show that H accurately represents the connectivity as well as to bound the size of H and the preprocessing time in the decremental setting.

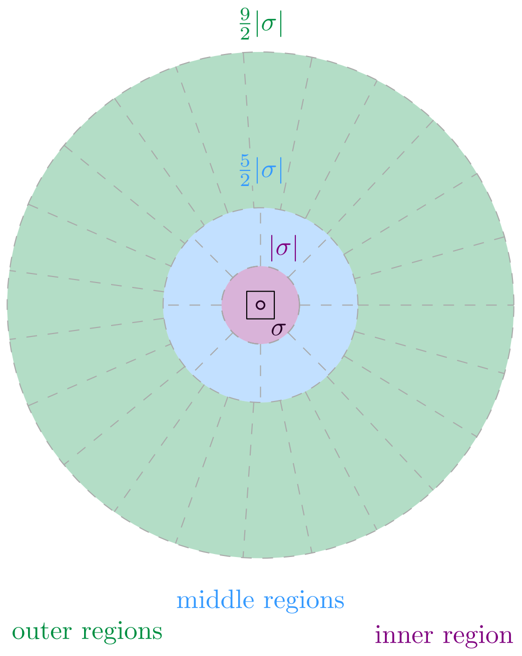


Figure 8 The regions defined by a cell σ.

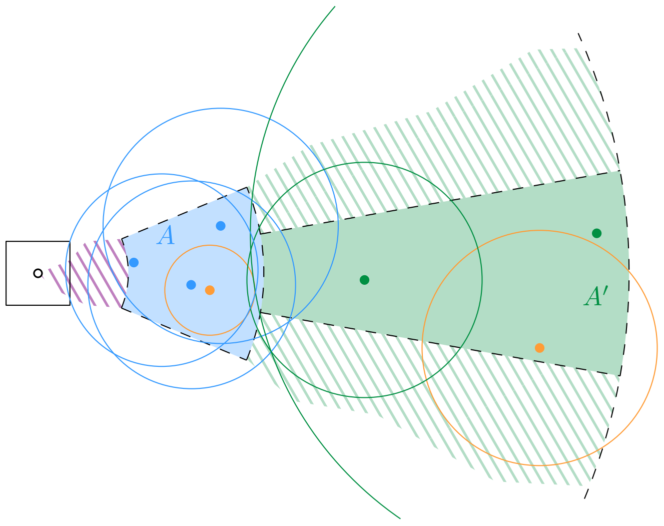


Figure 9 The set $S_1(A)$ is marked blue. The orange site in A is not in the set because its radius is too small. The orange site in A' is not in $S_1(A')$: even though its radius is in the correct range, it does not touch or intersect the inner boundary.

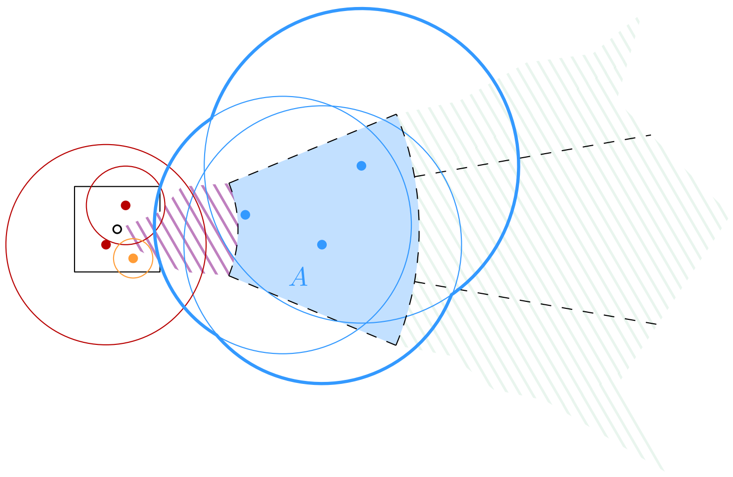


Figure 10 The red sites in σ are in $S_2(A)$. The radius of the orange site is in the correct range, but it does not intersect a site in $S_1(A)$ (marked blue).

▶ **Lemma 5.1.** *Let st be an edge in $\mathcal{D}(S)$ with $r_s \leq r_t$, then*

1. *there is a cell $\sigma \in N(t)$ with $s \in \sigma$ such that σ defines a region A with $t \in A$; and*
2. *all cells that define a region A with $t \in S_1(A)$ are in $N(t)$.*

The proof for Lemma 5.1 can be found in the full version of the paper. Before we argue that H accurately represents the connectivity of $\mathcal{D}(S)$, we show that the associated sites of a region in $\mathcal{A}$ form a clique in $\mathcal{D}(S)$.

▶ **Lemma 5.2.** *Suppose that $d_1 \geq 23$ and $d_2 \geq 8$. Then, for any region $A \in \mathcal{A}$, the associated sites in $S_1(A)$ form a clique in $\mathcal{D}(S)$.*

Proof sketch. (Full proof in the full version) The diameter of the inner and middle regions is at most $2|\sigma|$, thus two sites in $S_1(A)$ always intersect.

If a site t lies in the outer region, we can show that the lines segments that are perpendicular to the boundary rays of the cones, go through t and are inside the cone are contained in D_t. Then any other site t' that has a larger distance to $a(\sigma)$ than t either lies in the convex hull defined by the perpendicular line segments, or $D_{t'}$ contains a line segment that intersects the convex hull, see Figure 11. ◀

Having Lemmas 5.1 and 5.2 at hand, we can now show that H accurately represents the connectivity of $\mathcal{D}(S)$.

▶ **Lemma 5.3.** *Two sites are connected in H if and only if they are connected in $\mathcal{D}(S)$.*

Proof. Let $s, t \in S$. First, we show that if s and t are connected in H, they are also connected in $\mathcal{D}(S)$. The path between s and t in H alternates between vertices in S and vertices in $\mathcal{A}$. Thus, it suffices to show that if two sites u and u' are connected with the same region $A \in \mathcal{A}$, they are also connected in $\mathcal{D}(S)$. This follows directly from Lemma 5.2: if u and u' both lie in $S_1(A)$, they are part of the same clique. Otherwise, $S_2(A)$ is non-empty, and there is at least one site in $S_1(A)$ which intersects the site in $S_2(A)$. Then u is connected to u' via the clique induced by $S_1(A)$, and the claim follows.

Now, we consider two sites connected in $\mathcal{D}(S)$, and we show that they are also connected in H. It suffices to show that if s, t are adjacent in $\mathcal{D}(S)$, they are connected in H. Assume without loss of generality that $r_s \leq r_t$, and let σ be the cell in $N(t)$ with $s \in \sigma$. The cell

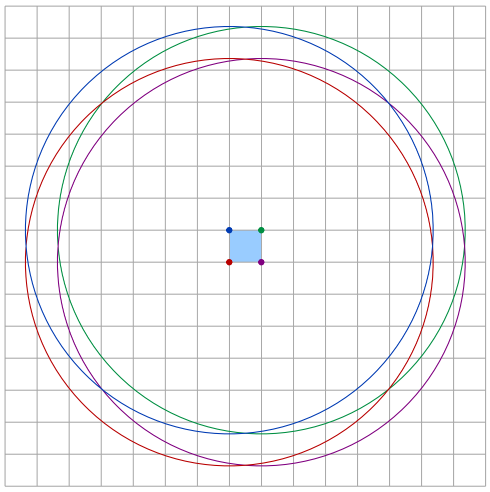


Figure 11 The disk $D(t, \frac{9}{2}|\sigma|)$ is contained in $N_{15\times 15}(\tau)$.

σ exists by the first property of Lemma 5.1, and it belongs to $\mathcal{F}$, since σ lies in the first $\lceil \log \Psi \rceil + 1$ levels of $\mathcal{G}$ and since $\sigma_s \subseteq \sigma$. Thus, we get that $t \in S_1(A)$ for some $A \in \mathcal{A}_{\mathcal{F}}$. As the regions with non-empty sets $S_1(A)$ are in $\mathcal{A}$, by definition, the edge tA exists in H.

Now we argue that $s \in S_2(A)$, and thus the edge As also exists in H. This follows by straightforward checking of the properties of a site in $S_2(A)$. We have $s \in \sigma$ by the definition of σ, and, by assumption, $r_s \leq r_t < 2|\sigma|$. Finally, as t is in $S_1(A)$ and as D_s and D_t intersect, there is at least one site in $S_1(A)$ that intersects D_s. The claim follows. ◀

After we have shown that H accurately represents the connectivity relation in $\mathcal{D}(S)$, we now show that the number of edge in H depends only on n and Ψ, and not on the number of edges in $\mathcal{D}(S)$ or the diameter of S. The proof of the following lemma can be found in the full version.

▶ **Lemma 5.4.** *The proxy graph H has $O(n)$ vertices and $O(n \log \Psi)$ edges.*

5.2 The decremental data structure

The decremental data structure has several components: we store a quadforest containing the cells defining $\mathcal{A}$ and for every $A \in \mathcal{A}$, we store the sets $S_1(A)$ and $S_2(A)$. For each region $A \in \mathcal{A}$, we store a disk revealing structure (RDS) as in Theorem 1.1 with $B = S_1(A)$ and $R = S_2(A)$. Finally, we store the proxy graph H in a Holm et al. data structure $\mathcal{H}$ [8]. See Figure 12 for an illustration.

As usual, the connectivity queries are answered using $\mathcal{H}$. To delete a site s, we first remove from $\mathcal{H}$ all incident edges of s. Then, we go through all regions A with $s \in S_1(A)$. We remove s from $S_1(A)$ and the RDS of A, and we let U be the set of revealed sites from $S_2(A)$ reported by the RDS. We delete each such site $u \in U$ from $S_2(A)$ and the corresponding RDS. Additionally, we delete the edges uA for $u \in U$ from $\mathcal{H}$ for all $u \in U$ that are not also in $S_1(A)$. Next, for each region A with $s \in S_2(A)$, we remove s from $S_2(A)$ and the associated RDS.

This gives us a time bound for the preprocessing time and the main theorem follows.

▶ **Lemma 5.5.** *Given a set S of n sites, we can construct the data structure described above in $O\big(n \log^5 n \lambda_6(\log n) + n \log \Psi \log^3 n\big)$ time.*

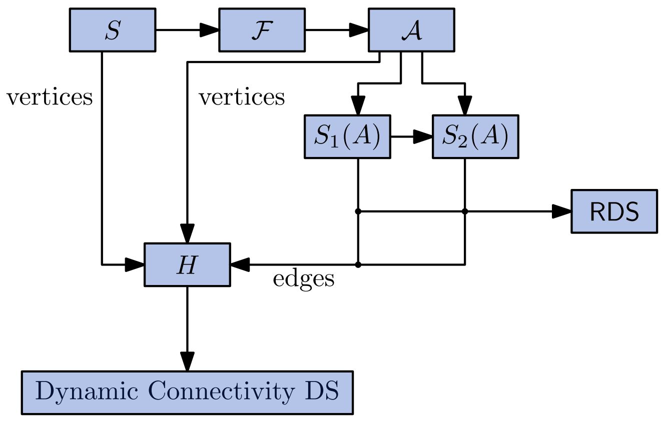


Figure 12 The structure of the decremental data structure.

▶ **Theorem 5.6.** *The data structure handles m site deletions in overall $O\big((n\log^5 n + m\log^7 n)\lambda_6(\log n) + n\log\Psi\log^4 n\big)$ time. Furthermore, it correctly answers connectivity queries in $O(\log n/\log\log n)$ amortized time.*

6 Semi-dynamic arbitrary radius ratio

We extend the approach from Section 5 to obtain a decremental data structure with a running time that is independent of Ψ. The cost for dropping the dependence on Ψ is replacing the additive $O\big(n\log\Psi\log^4 n\big)$ term in the running time of Theorem 5.6 with an additional $O(\log n)$ factor in the first term. The $O\big(n\log\Psi\log^4 n\big)$ term in Theorem 5.6 arose from the total size of the sets $S_2(A)$, and thus from the height of the quadtrees in $\mathcal{F}$. We can get rid of this dependency by using a *compressed quadtree* $\mathcal{Q}$ instead of $\mathcal{F}$. The height and size of $\mathcal{Q}$ do not depend on the radius ratio of the diameter of S, but only on n. Nonetheless, the height of $\mathcal{Q}$ could still be $\Theta(n)$, which is not favorable for our purposes. In order to reduce the number of edges in our proxy graph to $O(n\log n)$, we use a *heavy path decomposition* of $\mathcal{Q}$ in combination with a *canonical decomposition* for every heavy path. Let $\operatorname{diam}(S) = \max_{s,t\in S}\|st\|$. To simplify our arguments, we assume without loss of generality that S and its associated radii are scaled all associated radii are at least 1. This allows us to keep working with our hierarchical grid $\mathcal{G}$, as defined in Section 2.

Compressed quadtrees. The quadtree defined for a set $\mathcal{C}$ of $O(n)$ cells as in Section 2, has $O(n)$ leaves and height $O(\log(|\rho|))$, where ρ is the smallest cell in $\mathcal{G}$ that contains all cells of $\mathcal{C}$. This height can be arbitrarily large, even if n is small. To avoid this, we use the notion of a *compressed quadtree* $\mathcal{Q}$ as defined by Har-Peled [7] among others. $\mathcal{Q}$ has $O(n)$ vertices, height $O(n)$, and it can be constructed in $O(n\log n)$ time [2,7]. While the latter construction algorithm is stated for planar point sets it can be applied by considering a set of $O(n)$ virtual sites, similar to a construction of Har-Peled [7], see Figure 13.

Heavy paths. Let T be a rooted ordered tree. An edge $uv \in T$ is called *heavy* if v is the first child of u that maximizes the total number of nodes in the subtree rooted at v. Otherwise, the edge uv is *light*. By definition, every interior node in T has exactly one child that is connected by a heavy edge. A *heavy path* is a maximum path in T that consists only of heavy edges. The *heavy path decomposition* of T is the set of all the heavy paths in T. The following lemma summarizes a classic result on the properties of heavy path decompositions.

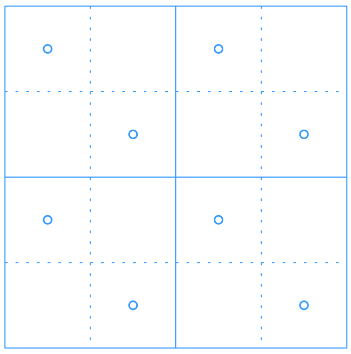


Figure 13 Four cells from $N_{15\times 15}(\sigma)$ with the virtual sites.

▶ **Lemma 6.1** (Sleator and Tarjan [12]). *Let T be a tree with n vertices. Then, the following properties hold:*

1. *Every leaf-root path in T contains $O(\log n)$ light edges;*
2. *every vertex of T lies on exactly one heavy path; and*
3. *the heavy path decomposition of T can be constructed in $O(n)$ time.*

The proxy graph. The general structure of the proxy graph is as in Section 5.1, and we will often refer back to it. We still have a bipartite graph with S on one side and a set of regions vertices on the other side. The regions will again be used to define sets $S_1(A)$ and $S_2(A)$ that will determine the edges. However, we will adapt the regions A and define them based certain *subpaths* of the compressed quadtree $\mathcal{Q}$ instead of single cells. Furthermore, we will relax the condition on the radii in the definition of the sets $S_1(A)$.

As usual, for a site $s \in S$, let σ_s be the cell in $\mathcal{G}$ with $s \in \sigma_s$ and $|\sigma_s| \le r_s < 2|\sigma_s|$. Let $N(s)$ be the (15×15)-neighborhood of σ_s. Let $\mathcal{N} = \{N(s) \mid s \in S\}$, and let $\mathcal{Q}$ be the compressed quadtree for $\mathcal{N}$. Now, let $\mathcal{R}$ be the heavy path decomposition of $\mathcal{Q}$, as in Lemma 6.1. For each heavy path $R \in \mathcal{R}$, we find a set $\mathcal{P}_R$ of *canonical paths* such that every subpath of R can be written as the disjoint union of $O(\log n)$ canonical paths. To be precise, for each $R \in \mathcal{R}$, we build a *biased* binary search tree T_R with the cells of R in the leaves, sorted by increasing diameter. The weights in the biased binary search tree are chosen as described by Sleator and Tarjan [12]: for a node σ of R, let the weight w_σ be the number of nodes in $\mathcal{Q}$ that are below σ (including σ), but not below another node of R below σ. Then, the depth of the leaf σ in T_R is $O(\log(w_R/w_\sigma))$, where w_R is the total weight of all leaves in T_R. We associate each vertex v in T_R with the path induced by the cells in the subtree rooted at v, and we add this path to $\mathcal{P}_R$. Using this construction, we can write every path in $\mathcal{Q}$ that starts at the root as the disjoint union of $O(\log n)$ canonical path:

▶ **Lemma 6.2.** *Let σ be a vertex of $\mathcal{Q}$, and let π be the path from the root of $\mathcal{Q}$ to σ. There exists a set $\mathcal{P}_\pi$ of canonical paths such that: (i) $|\mathcal{P}_\pi| = O(\log n)$; and (ii) π is the disjoint union of the canonical paths in $\mathcal{P}_\pi$.*

Proof sketch. (Full proof in full version) By Lemma 6.1 there are $O(\log n)$ heavy paths $R_1, \dots, R_k$ along π. The subpaths defined by the search paths to the smallest cell of a heavy path R_i partition π. Furthermore, by summing over the weights of the leaves of the biased binary search tree, we get that the overall number of canonical paths for π is $O(\log n)$. ◀

The vertex set of the proxy graph H again consists of S and a set of regions $\mathcal{A}$. We define $O(1)$ regions for each canonical path R in a similar way as in Section 5.1. Let σ be the smallest cell and τ the largest cell of R. The *inner* and *middle regions* of R are defined

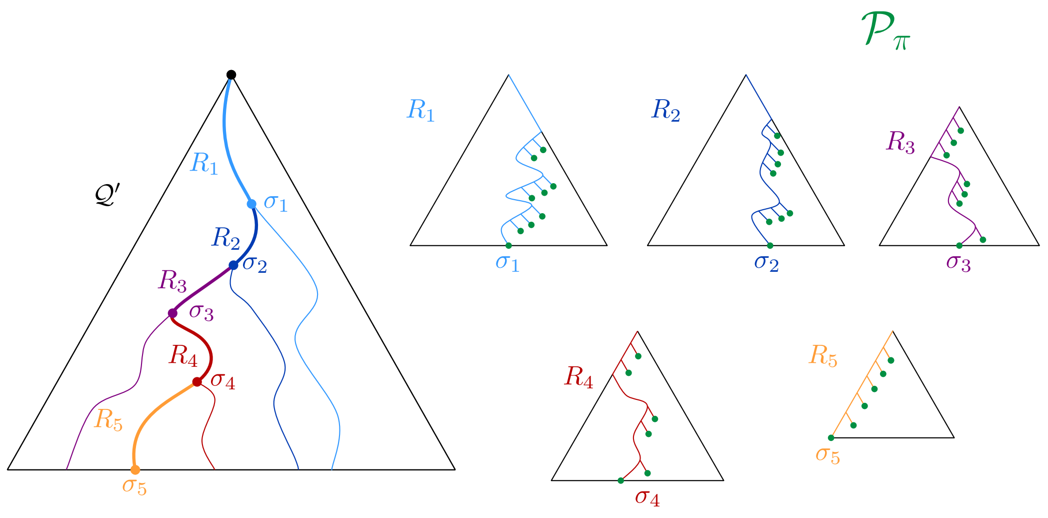


Figure 14 Illustration of Lemma 6.2. On the left, we see the decomposition of R into $R_1, \ldots R_k$. On the right, the vertices defining $\mathcal{P}_\pi$ are depicted in green.

as in Section 5.1, using σ as the defining cell. For the *outer regions* of R, we extend the outer radius of the annulus: they are defined as the intersections of the cones in $\mathcal{C}_{d_1}$ with the annulus of inner radius $\frac{5}{2}|\sigma|$ and outer radius $\frac{5}{2}|\sigma| + 2|\tau|$, again centered at $a(\sigma)$. The set $\mathcal{A}$ now contains the regions defined in this way for all canonical paths.

Given a region $A \in \mathcal{A}$ for a canonical path R with smallest cell σ and largest cell τ, we can now define the sets $S_1(A)$ and $S_2(A)$. These definitions are similar to the analogous sets in Section 5.1. The set $S_1(A)$ contains all sites t such that (i) $t \in A$; (ii) $|\sigma| \leq r_t \leq 2|\tau|$; and (iii) $\|a(\sigma)t\| \leq r_t + \frac{5}{2}|\sigma|$. The definition for $S_2(A)$ is also similar to Section 5.1, using canonical paths instead of cells. Let $s \in S$ be a site, and π_s be the path in $\mathcal{Q}$ from the root to σ_s. Let $\mathcal{P}_s$ be the decomposition of π_s into canonical paths as in Lemma 6.2. Let A be a region, defined by a canonical path P. Then, $s \in S_2(A)$ if (i) $P \in \mathcal{P}_s$; and (ii) s is adjacent in $\mathcal{D}(S)$ to at least one site in $S_1(A)$. These are basically the conditions we had in Section 5.1. However, as the definition is restricted to those canonical paths in $\mathcal{P}_{\pi_s}$, not all sites satisfying these conditions are considered. Using similar arguments as in Section 5.1, this suffices to make sure that the proxy graph represents the connectivity, while also ensuring that each site s lies in few sets $S_2(A)$.

The graph H is now again defined by connecting each region $A \in \mathcal{A}$ to all sites in $s \in S_1(A) \cup S_2(A)$. By similar considerations as in Section 5, we obtain a decremental data structure for disk graphs with arbitrary radii. The details can be found in the full version.

References

1 Pankaj K. Agarwal, Ravid Cohen, Dan Halperin, and Wolfgang Mulzer. Maintaining the union of unit discs under insertions with near-optimal overhead. In *Proc. 35th Annu. Sympos. Comput. Geom. (SoCG)*, pages 26:1–26:15, 2019. doi:10.4230/LIPIcs.SoCG.2019.26.

2 Kevin Buchin, Maarten Löffler, Pat Morin, and Wolfgang Mulzer. Preprocessing imprecise points for Delaunay triangulation: Simplified and extended. *Algorithmica*, 61(3):674–693, 2011. doi:10.1007/s00453-010-9430-0.

3 Timothy M. Chan. Dynamic geometric data structures via shallow cuttings. *Discrete Comput. Geom.*, 64(4):1235–1252, 2020. doi:10.1007/s00454-020-00229-5.

4 Timothy M. Chan, Mihai Pătraşcu, and Liam Roditty. Dynamic connectivity: Connecting to networks and geometry. *SIAM J. Comput.*, 40(2):333–349, 2011. doi:10.1137/090751670.

5 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* MIT Press, 3rd edition, 2009.

6 David Eppstein, Giuseppe F Italiano, Roberto Tamassia, Robert E Tarjan, Jeffery Westbrook, and Moti Yung. Maintenance of a minimum spanning forest in a dynamic plane graph. *J. Algorithms*, 13(1):33–54, 1992. doi:10.1016/0196-6774(92)90004-V.

7 Sariel Har-Peled. *Geometric Approximation Algorithms*, volume 173. American Mathematical Society, 2011. doi:10.1090/surv/173.

8 Jacob Holm, Kristian de Lichtenberg, and Mikkel Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *J. ACM*, 48(4):723–760, 2001. doi:10.1145/502090.502095.

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

10 Chih-Hung Liu. Nearly optimal planar k nearest neighbors queries under general distance functions. In *Proc. 31st Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 2842–2859, 2020. doi:10.1137/1.9781611975994.173.

11 Micha Sharir and Pankaj K. Agarwal. *Davenport-Schinzel sequences and their geometric applications.* Cambridge University Press, 1995.

12 Daniel D. Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *J. Comput. System Sci.*, 26(3):362–391, 1983. doi:10.1016/0022-0000(83)90006-5.

13 Mikkel Thorup. Near-optimal fully-dynamic graph connectivity. In *Proc. 32nd Annu. ACM Sympos. Theory Comput. (STOC)*, pages 343–350, 2000.