

Dynamic Connectivity for Unit Disk Graphs*

Haim Kaplan[†]Wolfgang Mulzer[‡]Liam Roditty[§]Paul Seiferth[‡]

Abstract

Let $S \subset \mathbb{R}^2$ be a set of point sites. The *unit disk graph* $UD(S)$ of S has vertex set S and an edge between two sites s, t if and only if $|st| \leq 1$.

We present a data structure that maintains the connected components of $UD(S)$ when S changes dynamically. It takes $O(\log^2 n)$ time to insert or delete a site in S and $O(\log n / \log \log n)$ time to determine if two sites are in the same connected component. Here, n is the maximum size of S at any time. A simple variant improves the update time to $O(\log n \log \log n)$ at the cost of a slightly increased query time of $O(\log n)$.

1 Introduction

Computing the connected components of a graph G is one of the most fundamental problems in algorithmic graph theory. When G is static, several classic solutions exist, e.g., BFS or DFS. However, if G can change dynamically, the problem becomes much more challenging. In this case, we would like a data structure for *connectivity queries*: given two vertices s and t , are s and t in the same connected component of G ? Additionally, we would like to be able to insert and delete edges or singleton vertices. For general graphs, there is the following result due to Holm et al. [8].

Theorem 1 (Holm et al., Theorem 3) *Let G be a graph with n vertices. There is a deterministic data structure such that edge insertions or deletions in G take amortized time $O(\log^2 n)$, and connectivity queries take worst-case time $O(\log n / \log \log n)$.*

Even though Theorem 1 assumes n to be fixed, we can use a standard rebuilding method to support vertex insertion and deletion within the same amortized time bounds, by rebuilding the data structure whenever the number of vertices changes by a factor of 2. For planar graphs, Eppstein et al. achieved $O(\log n)$ time for both updates and queries [7].

However, the model of edge insertions and deletions may be too restrictive. For example, one natural situation where more powerful operations are needed occurs in *unit disk graphs*. Let $S \subset \mathbb{R}^2$ be a set of

point sites. The *unit disk graph* $UD(S)$ of S has vertex set S and an edge between two sites $s, t \in S$ if and only if the Euclidean distance $|st|$ is at most 1. Now, we want to maintain the connected components of $UD(S)$ as the *vertex set* S changes dynamically. In this case, a single update may change the graph quite dramatically, since one site may have many incident edges. Nevertheless, Chan et al. [5] observed that by combining known results one can derive a data structure with update time $O(\log^{10} n)$ and query time $O(\log n / \log \log n)$. The construction is as follows (see Figure 1): ① let T be the Euclidean minimum span-

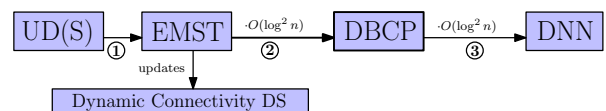


Figure 1: A solution with $O(\log^{10} n)$ update time.

ning tree (EMST) of S . If we remove all edges with length larger than 1 from T , the resulting forest F is a spanning forest for $UD(S)$. Thus, to maintain the components of $UD(S)$, it suffices to maintain the components of F . We create data structure D of Holm et al. to maintain F . Since the EMST has maximum degree 6, inserting or deleting a site from S changes $O(1)$ edges in T . Suppose we can efficiently find the set E of edges that change during an update. Then, we can update the components in F through $O(1)$ updates in D , taking all edges in E of length at most 1. ② To find E , we need to dynamically maintain the EMST T when S changes. This can be done using a technique of Agarwal et al. that reduces the problem to several instances of the *dynamic bichromatic closest pair problem* (DBCP), with an overhead of $O(\log^2 n)$ in the update time [1]. ③ Eppstein showed that the DBCP problem can in turn be solved through a reduction to several instances of the dynamic nearest neighbor problem (DNN) for points in the plane [6]. Again, we incur another $O(\log^2 n)$ factor as overhead in the update time. Using Chan’s DNN structure [4] with amortized expected update time $O(\log^6 n)$, we get a total update time of $O(\log^{10} n)$. We can use D to answer queries in $O(\log n / \log \log n)$ time.

Our Results. We improve the previous result by following a similar approach, but in every step we use a method more specifically tailored to unit disks. Instead of the EMST in ①, we use a much simpler graph

*Supported by GIF project 1161&DFG project MU/3501-1.

[†]Tel Aviv University, Israel. haimk@post.tau.ac.il

[‡]Institut für Informatik, Freie Universität Berlin, Germany {mulzer,pseiferth}@inf.fu-berlin.de

[§]Bar Ilan University, Israel. liamr@macs.biu.ac.il

on grid cells that also captures the connectivity of $\text{UD}(S)$. Then we can avoid the $O(\log^2 n)$ overhead in ② and ③ and substitute the DNN data structure by a *dynamic lower envelope* (DLE) structure for pseudolines in \mathbb{R}^2 . In Section 2 we review suitable DLE structures and their properties. In Section 3 we prove our first main theorem:

Theorem 2 *There is a dynamic connectivity structure for unit disk graphs such that the insertion or deletion of a site takes amortized time $O(\log^2 n)$ 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.*

In Section 4, we use a grid-based *planar* graph to represent the connectivity of $\text{UD}(S)$. Then we can replace Theorem 1 by the result for planar graphs by Eppstein et al. Updates now take $O(\log n \log \log n)$ time, but the query time slightly increases to $O(\log n)$.

2 Dynamic Lower Envelopes

Let L be a set of *pseudolines* in the plane, i.e., each element of L is a simple continuous curve and any two distinct curves in L intersect in exactly one point. The *lower envelope* of L is the pointwise minimum of the graphs of the curves in L . In Section 3 we need to dynamically maintain the lower envelope of L . Overmars and van Leeuwen show how to maintain the lower envelope of a set of lines with update time $O(\log^2 n)$ such that vertical ray shooting queries can be answered in $O(\log^2 n)$ time [10]. Chan improves this to $O(\log^{1+\epsilon})$ for updates and queries [3]. Using the kinetic heap structure of Kaplan et al. [9] one can obtain $O(\log n \log \log n)$. Brodal and Jacob showed that the optimal bound $O(\log n)$ can be achieved [2]. Except for the last result, one can verify that all these approaches also work with pseudolines; they only need a total ordering of the lines along the lower envelope.

Lemma 3 *Let L be a dynamic set of at most n pseudolines. We can maintain the lower envelope of L with $O(\log n \log \log n)$ amortized update time and $O(\log n)$ amortized query time.*

Remark. The applicability of the result by Brodal and Jacob [2] is not clear to us, and poses an interesting challenge for further investigation.

3 The Data Structure

Let $S \subset \mathbb{R}^2$ be a set of sites. We define an *auxiliary graph* G that represents the connectivity of $\text{UD}(S)$. The vertices of G are cells of a grid. To see if two cells form an edge, we maintain a bichromatic matching of the sites in the grid cells. This matching is updated with the help of two DLE data structures.

The Grid Graph (new ①). Let \mathcal{G} be a planar grid whose cells are disjoint axis-aligned squares with diameter 1. For any grid cell $\sigma \in \mathcal{G}$, the sites $\sigma \cap S$ induce a clique in $\text{UD}(S)$. For $S \subset \mathbb{R}^2$, we define a graph G whose vertices are the *non-empty* cells $\sigma \in \mathcal{G}$, i.e., the cells with $\sigma \cap S \neq \emptyset$. The *neighborhood* $N(\sigma)$ of a cell $\sigma \in \mathcal{G}$ is the 5×5 block of cells in \mathcal{G} with σ in the center. We call two cells *neighboring* if they are in each other's neighborhood. The endpoints of any edge in $\text{UD}(S)$ must lie in neighboring cells. To obtain the edges of G , we connect every pair of distinct neighboring grid cells that contain the endpoints of an edge in $\text{UD}(S)$. By construction, and since the sites inside each cell form a clique, the connectivity between two sites s, t in $\text{UD}(S)$ is the same as for the corresponding cells in G .

Lemma 4 *Let $s, t \in S$ be two sites and let σ and τ be the cells in \mathcal{G} that contain s and t , respectively. There is an s - t path in $\text{UD}(S)$ if and only if there is a σ - τ path in G .*

We build the data structure from Theorem 1 for G . When a site s is inserted into or deleted from S , only $O(1)$ edges in G change, since only the neighborhood of the cell of s is affected. Thus, once the set E of changing edges is determined, we can update G in time $O(\log^2 n)$, by Theorem 1.

Finding the Edges E (new ②). It remains to find the edges E of G that change when we update S . For this, we maintain for each pair of non-empty neighboring cells a *maximal bichromatic matching* (MBM) between their sites, similar to Eppstein's method [6]. Let $R \subseteq S$ and $B \subseteq S$ be two sets of sites. An MBM between R and B is a maximal set of vertex-disjoint edges in $(R \times B) \cap \text{UD}(S)$, the bipartite graph on $R \cup B$ consisting of all edges of $\text{UD}(S)$ with one endpoint in R and one endpoint in B .

For each pair $\{\sigma, \tau\}$ of neighboring cells in \mathcal{G} , we build an MBM $M_{\{\sigma, \tau\}}$ for $R = \sigma \cap S$ and $B = \tau \cap S$. By definition, there is an edge between σ and τ in G if and only if $M_{\{\sigma, \tau\}}$ is not empty. When inserting or deleting a site s from S , we proceed as follows: let $\sigma \in \mathcal{G}$ be the cell with $s \in \sigma$. We go through all cells $\tau \in N(\sigma)$ and update the MBM $M_{\{\sigma, \tau\}}$ (by inserting or deleting s from the relevant set). If $M_{\{\sigma, \tau\}}$ becomes non-empty during an insertion or becomes empty during a deletion, we add the edge $\sigma\tau$ to E and mark it for insertion or deletion, respectively. We summarize this construction in the following lemma.

Lemma 5 *Suppose we can maintain an MBM for each pair of non-empty neighboring cells with update time $O(U(n))$, where n is the maximum number of sites. Then we can dynamically maintain the adjacency lists of G with update time $O(U(n))$.*

Dynamically Maintaining an MBM (new ③). Let $\sigma \neq \tau$ be two neighboring cells of \mathcal{G} , and let $R = \sigma \cap S$ and $B = \tau \cap S$. We show that an MBM between R and B can be efficiently maintained using two DLE structures for pseudolines. We fix a line ℓ that separates R and B . Since R, B are in two distinct grid cells, we can take a supporting line of one of the four boundaries of σ . We have the following lemma.

Lemma 6 *Let $R, B \subseteq S$ be two sets with a total of at most n sites, separated by a line ℓ . There exists a dynamic data structure that maintains an MBM for R and B with $O(\log n \log \log n)$ update time.*

Proof. We rotate and translate everything such that ℓ is the x -axis and all sites in R have positive x -coordinate. We consider the set U_R of unit disks with centers in R (see Figure 2). Then a site in B forms an edge with *some* site in R if and only if it is contained in the union of the disks in U_R . To detect this, we maintain the lower envelope of U_R . More precisely, consider the following set L_R of pseudolines: for each disk of U_R , take the arc that defines the lower part of the boundary of the disk and extend both ends straight upward to ∞ . We build a data structure D_R

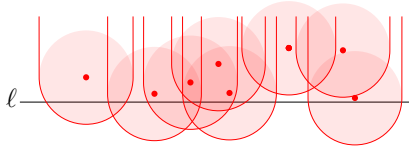


Figure 2: The set L_R induced by R .

for L_R according to Lemma 3. Analogously, we define a set of pseudolines L_B and a dynamic envelope structure D_B for B .

To maintain the MBM M , we store in D_R the currently unmatched sites of R , and in D_B the currently unmatched sites of B . When inserting a site r into R , we perform a vertical ray shooting query in D_B with r to get a pseudoline of L_B . Let $b \in B$ the site for that pseudoline. If $|rb| \leq 1$, we add the edge rb to M , and delete the pseudoline of b from D_B . Otherwise we insert the pseudoline of r into D_R . By construction, if there is an edge between r and an unmatched site in B , then there is also an edge between r and b . Hence, the insertion procedure correctly maintains an MBM. Now suppose we want to delete a site r from R . If r is unmatched, we delete the pseudoline corresponding to r from D_R . Otherwise, we remove the edge rb from M , and we reinsert b as above, looking for a new unmatched site in R for b . Updating B is analogous.

Inserting and deleting a site requires $O(1)$ insertions, deletions, or queries in D_R or D_B , so the lemma follows. \square

To obtain Theorem 2, we combine Lemma 4, 5, and 6.

4 Improving the Update Time

The bottleneck for the update time in Section 3 lies in the use of Theorem 1. We now define a planar graph G_p that is similar to the grid graph G : it represents the connectivity of $\text{UD}(S)$ and an update of S changes $O(1)$ vertices and edges in G_p . These vertices and edges can be found in $O(1)$ time. Since G_p is planar, we can use the result of Eppstein et al. to maintain the connectivity of G_p with $O(\log n)$ amortized update and worst-case query time [7], giving the next theorem.

Theorem 7 *There is a dynamic connectivity structure for unit disk graphs such that insertion or deletion of a site takes amortized time $O(\log n \log \log n)$ and a connectivity query takes worst-case time $O(\log n)$, where n is the maximum number of sites at any time.*

The Planar Graph. Let $S \subset \mathbb{R}^2$ be a set of sites. For any pair of non-empty grid cells σ, τ , let $M_{\{\sigma, \tau\}}$ be the MBM as above. For any non-empty MBM $M_{\{\sigma, \tau\}}$, we pick an arbitrary edge $rb \in M_{\{\sigma, \tau\}}$ with $r \in \sigma$ and $b \in \tau$ as *representative edge*. Let $T \subseteq S$ be the set of sites incident to a representative edge. We use the unit disk graph $\text{UD}(T)$ as basis for our planar graph G_p . If we contract in each grid cell σ the subgraph of $\text{UD}(T)$ induced by $T \cap \sigma$ to a single vertex, we get the graph G from Section 3. Hence, by Lemma 4, $\text{UD}(T)$ represents the connectivity of $\text{UD}(S)$.

To get G_p from $\text{UD}(T)$, we consider the straight line drawing of $\text{UD}(T)$. For a crossing of two edges st and uv in $\text{UD}(T)$, we add a new site x at the intersection and call x a *crossing site*. We remove st and uv and we add the four new edges sx , xt , ux , and xv . We repeat this operation until there are no more crossings in $\text{UD}(T)$. This is a standard method for making unit disk graphs planar. The next lemma, due to Yan et al. [11], shows that it preserves connectivity.

Lemma 8 *Let ab and uv be edges in $\text{UD}(T)$ that cross. Then a, b, u , and v are in the same connected component of $\text{UD}(\{a, b, u, v\})$.*

Using Lemma 8 we now show that G_p has the same connectivity as $\text{UD}(T)$. Thus, by Lemma 4, G_p represents the connectivity of $\text{UD}(S)$.

Lemma 9 *Let $s, t \in T$ be two sites. Then s and t are connected in $\text{UD}(T)$ if and only if they are connected in G_p .*

Proof. Since going from $\text{UD}(T)$ to G_p only increases the connectivity, all sites s and t connected in $\text{UD}(T)$ are also connected in G_p .

For the other direction, let $s = p_1, \dots, p_k = t$ be a path in G_p between $s, t \in T$. For each p_i , we define

a set $V_i \subseteq T$ as follows: if p_i is a site in T , we set $V_i = \{p_i\}$. Otherwise, p_i is a crossing site, created by a crossing of two edges uv and ab in $\text{UD}(T)$. We set $V_i = \{a, b, u, v\}$. By Lemma 8, the sites a, b, u, v are in the same connected component of $\text{UD}(T)$. Furthermore, we have $V_{i-1} \cap V_i \neq \emptyset$, since $p_{i-1}p_i$ is a proper subsegment of an edge e in $\text{UD}(T)$, and at least one endpoint of e lies in V_{i-1} .

We prove by induction that all sites in $\bigcup_{i=1}^j V_i$ lie in the same connected component of $\text{UD}(T)$, for $j = 1, \dots, k$. For $j = 1$, this is clear. Now, consider V_j . If $V_{j-1} \cap V_j \neq \emptyset$, then the claim follows by induction, since all sites in V_j are in the same component. Otherwise, $V_j = \{p_j\}$, p_j is a site in T , and there is an edge in $\text{UD}(T)$ between p_j and $\bigcup_{i=1}^{j-1} V_i$, implying the claim. By setting $j = k$, we now have that s and t are connected in $\text{UD}(T)$. \square

Maintaining G_p . We maintain an MBM between any two neighboring non-empty grid cells and we pick one representative edge for each MBM. Let s be a site we want to insert or delete from S . Let σ be the grid cell containing s . We update for all $\tau \in N(\sigma)$ the MBM $M_{\{\sigma, \tau\}}$, and we collect the sites of all representative edges that need to be inserted or deleted in two sets I and D : if $M_{\{\sigma, \tau\}}$ changes from empty to non-empty, we pick a representative edge for $M_{\{\sigma, \tau\}}$ and put its two endpoints into I . If we delete the representative edge of $M_{\{\sigma, \tau\}}$, we put its two endpoints into D , and, if possible, we pick a new representative edge for $M_{\{\sigma, \tau\}}$. We put the endpoints of the new edge into I . Since $|N(\sigma)| = O(1)$, the sets I and D contain $O(1)$ to be added or deleted from G_p .

Next, we show how to update G_p with a site s in I or D . First we insert or delete s in $\text{UD}(T)$ and determine which edges change in $\text{UD}(T)$. Each such edge may create or delete several edges in G_p that need to be handled. The next lemma shows that s can create or delete $O(1)$ edges in G_p and that these edges can be found in $O(1)$ time. This finishes the proof of Theorem 7.

Lemma 10 *Let s be a site in I or D . Updating G_p with s changes $O(1)$ edges and vertices. They can be found in $O(1)$ time.*

Proof. Suppose that $s \in I$, i.e., we want to insert s . Let σ be the cell containing s . We add s to T and collect all edges in $\text{UD}(T)$ incident to s in a set E as follows: we start with $E = \emptyset$. First, for each $t \in T \cap \sigma$ we add the edge st to E . Since σ has diameter 1, all these edges are valid edges in $\text{UD}(T)$. Next, we go through all cells $\tau \in N(\sigma)$. We check for all sites $t \in \tau \cap T$ if $|st| \leq 1$. If so, we add st to E .

To update G_p , we find all edges in G_p crossed by edges in E . Since all edges in E and in G_p cross $O(1)$ grid cells, and since each grid cell contains $O(1)$

sites and crossing sites, this can be done in $O(1)$ time. We add all these edges to E , and we perform the planarization procedure on E . This gives all edges and vertices in G_p that need to be changed, in $O(1)$ time.

Deleting a site is done in a similar manner. \square

References

- [1] P. K. Agarwal, H. Edelsbrunner, O. Schwarzkopf, and E. Welzl. Euclidean minimum spanning trees and bichromatic closest pairs. *DCG*, 6(5):407–422, 1991.
- [2] G. S. Brodal and R. Jacob. Dynamic planar convex hull. In *Proc. 43rd FOCS*, pages 617–626, 2002.
- [3] T. M. Chan. Dynamic planar convex hull operations in near-logarithmic amortized time. *JACM*, 48(1):1–12, 2001.
- [4] T. M. Chan. A dynamic data structure for 3-D convex hulls and 2-D nearest neighbor queries. *JACM*, 57(3):Art. 16, 15, 2010.
- [5] T. M. Chan, M. Pătraşcu, and L. Roditty. Dynamic connectivity: connecting to networks and geometry. *SICOMP*, 40(2):333–349, 2011.
- [6] D. Eppstein. Dynamic Euclidean minimum spanning trees and extrema of binary functions. *DCG*, 13:111–122, 1995.
- [7] D. Eppstein, G. F. Italiano, R. Tamassia, R. E. Tarjan, J. Westbrook, and M. Yung. Maintenance of a minimum spanning forest in a dynamic plane graph. *J. Algorithms*, 13(1):33–54, 1992.
- [8] J. Holm, K. de Lichtenberg, and M. Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *JACM*, 48(4):723–760, 2001.
- [9] H. Kaplan, R. E. Tarjan, and K. Tsioutsoulis. Faster kinetic heaps and their use in broadcast scheduling (extended abstract). In *Proc. 12th SODA*, pages 836–844, 2001.
- [10] M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *JCSS*, 23(2):166–204, 1981.
- [11] C. Yan, Y. Xiang, and F. F. Dragan. Compact and low delay routing labeling scheme for unit disk graphs. *CGTA*, 45(7):305–325, 2012.