Finding Triangles and Computing the Girth in Disk Graphs*

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

Abstract

Let $S \subset \mathbb{R}^2$ be a set of n point *sites*, where each $s \in S$ has an *associated radius* $r_s > 0$. The *disk graph* $\mathrm{D}(S)$ of S is the graph with vertex set S and an edge between two sites s and t if and only if $|st| \leq r_s + r_t$, i.e., if the disks with centers s and t and radii r_s and r_t, respectively, intersect. Disk graphs are useful to model sensor networks.

We study the problems of finding triangles and of computing the girth in disk graphs. These problems are notoriously hard for general graphs, but better solutions exist for special graph graph classes, such as planar graphs. We obtain similar results for disk graphs. In particular, we observe that the unweighted girth of a disk graph can be computed in $O(n \log n)$ worst-case time and that a shortest (Euclidean) triangle in a disk graph can be found in $O(n \log n)$ expected time.

1 Introduction

Disk graphs are geometrically defined graphs that show up in many applications and are defined as follows. We are given a set $S \subset \mathbb{R}^2$ of n point sites in the plane, such that each $s \in S$ has an *associated radius* $r_s > 0$. Let the *disk corresponding to* s, denoted by D_s, be the closed disk with center s and radius r_s. The *disk graph for* S, $\mathrm{D}(S)$, is the graph with vertex set S in which two sites s and t are connected by an (undirected) edge if any only if $D_s \cap D_t \neq \emptyset$ (or equivalently if any only if $|st| \leq r_s + r_t$). In a *weighted disk graph*, the weight of an edge (s, t) is equal to $|st|$: the Euclidean distance between s and t.

Even though disk graphs may be dense, it turns out that many algorithmic problems can be solved faster in disk graphs than in general graphs. For example, we can compute the BFS-tree from any given site in an unweighted disk graph in $O(n \operatorname{polylog}(n))$ expected time [2,9], and we can approximate the shortest paths distances in a weighted disk graph by a sparse *spanner* that can be constructed in $O(n \operatorname{polylog}(n))$ expected time [6, 9].

We give fast and simple algorithms for two classic problems when restricting the input to disk graphs. These problems are known to be challenging in general graphs. Our first problem is to determine whether a given graph contains a triangle (i.e., a complete subgraph on three vertices), and, if so, to find a triangle that minimizes the sum of its edge lengths (in the weighted case). For general unweighted graphs, the problem can be solved in $O(n^\omega)$ time using fast matrix multiplication [7, 8] (where $\omega < 2.37287$ is the matrix multiplication exponent). However, if we insist on *combinatorial* algorithms that do not use algebraic techniques, the fastest known algorithm runs in $O(n^3 \operatorname{polyloglog}(n)/\log^4 n)$ time [11]. In planar graphs, the problem can be solved in $O(n)$ time [4]. In the weighted case for general graphs, progress has been made only very recently [10].

The second problem is computing the *girth*, which is the length of a shortest cycle in G. Again, the best result for general unweighted graphs relies on fast matrix multiplication [8], while for planar graphs, the unweighted girth can be computed in linear time [4].

As we will see, these problems are easier for disk graphs. Indeed, finding triangles in disk graphs is almost a trivial problem. By using the intimate connection between disk graphs and planar graphs and some known results for planar graphs, we can extend our observations regarding triangles in disk graphs to an algorithm that computes the unweighted girth of disk graphs. Computing the length of a shortest triangle is a bit more difficult, but we can exploit the geometric structure of disk graphs to solve the decision version of this problem in $O(n \log n)$ time. Then we use Chan's framework for randomized geometric optimization algorithms [3] to solve the optimization problem in the same expected time.

2 Computing the Unweighted Girth

First, we consider the unweighted girth in disk graphs. Given a disk graph $\mathrm{D}(S)$, we would like to find a cycle in $\mathrm{D}(S)$ with the smallest number of edges. It turns out that this problem is closely related to the problem of computing the girth in planar graphs. The following simple property of disk graphs is the key to our algorithm. It has been observed before by Evens et al. [5]. For completeness, we include a proof.

Lemma 1 *Let $\mathrm{D}(S)$ be a disk graph that is not plane. (By this we mean that the embedding obtained by connecting each pair of adjacent sites s and t by a*

*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

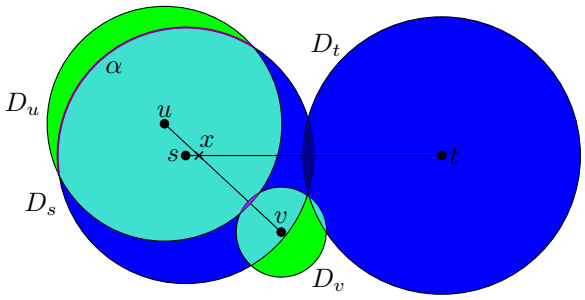

Figure 1: If $\mathrm{D}(S)$ is not plane, then three disks intersect in a common point.

straight segment between s and t, has at least one pair of segments that cross in their relative interior.) Then, there are three sites whose disks intersect in a common point.

Proof. Suppose that the two edges st and uv intersect at a point x. The sites s, t, u, and v are pairwise distinct, and without loss of generality, we may assume that: (i) $x \in D_s \cap D_u$; (ii) $r_s \geq r_u$; (iii) the edge st lies on the x-axis, with s to the left of t; and (iv) the site u lies above the x-axis, and the site v lies below the x-axis; see Figure 1.

Now consider the arc $\alpha = D_u \cap \partial D_s$. If $\alpha = \emptyset$, we are done, since then $D_u \subset D_s$, and $D_s \cap D_u \cap D_v \neq \emptyset$. Furthermore, α must contain a point below the x-axis, because v lies below the x-axis and otherwise we would hence have $D_s \cap D_u \cap D_v \neq \emptyset$. Since u is above the x-axis and since $r_s \geq r_u$, α must intersect the x-axis. This intersection must be to the left of s, since otherwise we would have $D_s \cap D_t \cap D_u \neq \emptyset$. Again since $r_s \geq r_u$, it follows that u lies to the left of s. Now we see that α contains no point that lies below the x-axis and to the right of s. From this and the fact that x lies to the right of s, we can conclude that $uv \cap \partial D_u \in D_s$, and hence $D_s \cap D_u \cap D_v \neq \emptyset$. □

By Lemma 1, we know that if $\mathrm{D}(S)$ is not plane, then the girth is 3. On the other hand, if $\mathrm{D}(S)$ is plane, and if the embedding is available, the girth can be found in $O(n)$ time. More precisely, we can use the following result by Chang and Lu [4].

Theorem 2 (Theorem 1.1 in [4]) *Let G be an unweighted planar graph with n vertices. The girth of G can be computed in $O(n)$ time.*

Combining Lemma 1 and Theorem 2, we immediately obtain a fast algorithm for computing the girth in disk graphs.

Theorem 3 *Let $\mathrm{D}(S)$ be a disk graph with n vertices. We can compute the unweighted girth of $\mathrm{D}(S)$ in $O(n \log n)$ worst-case time.*

Proof. We use a standard sweep-line algorithm to compute the arrangement of the disks corresponding to S [1]. The intersections of the disk boundaries are reported one by one, and the total time to report the first m intersections is $O(n \log n + m \log n)$ [1]. Since every edge in $\mathrm{D}(S)$ corresponds to two unique intersection points, it follows that as soon as $6n - 13$ intersection points have been reported, it must be the case that $\mathrm{D}(S)$ is not plane, and hence, by Lemma 1, the girth is 3. Otherwise, we obtain an explicit representation of $\mathrm{D}(S)$, and we can test in $O(n)$ time whether is it plane. If this is not the case, we again output that the girth is 3. Finally, if $\mathrm{D}(S)$ is plane, we determine the unweighted girth in $O(n)$ time using Theorem 2. □

3 Finding a Shortest Triangle

Now we consider the situation where each edge in $\mathrm{D}(S)$ is weighted according to its Euclidean length. We would like to find a shortest triangle in $\mathrm{D}(S)$, i.e., a triangle that minimizes the sum of its edge lengths. First, we focus on the decision problem: given a parameter $W > 0$, does $\mathrm{D}(S)$ contain a triangle with weight at most W? Once an algorithm for the decision problem is available, a solution for the optimization problem will follow through a straightforward application of Chan's randomized framework for geometric optimization problems [3]. In the end, we will prove the following theorem.

Theorem 4 *Let $\mathrm{D}(S)$ be a disk graph with n vertices, where the edges are weighted according to their Euclidean lengths. We can compute a shortest triangle in $\mathrm{D}(S)$ in $O(n \log n)$ expected time, if it exists.*

3.1 The Decision Problem

Let $S \subset \mathbb{R}^2$ and a weight $W > 0$ be given. To decide if $\mathrm{D}(S)$ contains a triangle with weight at most W, we proceed as follows. We classify the sites as *small* and *large*, depending on their associated radius. This yields four possibilities for our desired triangle. To investigate each such possibility, we use a grid whose cells have a diameter proportional to W. First, we consider only triangles where all three vertices are small and lie in the same grid cell. This can be done using the tools from the previous section. If no cell contains such a triangle, we can show that the graph must be sparse and that we need to check only few further triangles. Details follow.

The Four Cases. Set $\ell = W/(12\sqrt{2})$. We say that a site $s \in S$ is *small*, if $r_s < \ell$, and *large*, otherwise. Depending on the number of small and large vertices, we classify the triangles in $\mathrm{D}(S)$ into four types:

(SSS) 3 small vertices, 0 large vertices

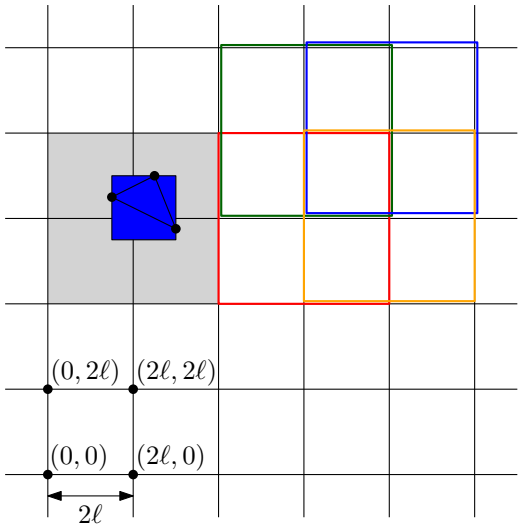

Figure 2: The four shifted grids, with a cell from each grid shown in red, orange, green, and blue, respectively. Every square with side length at most 2ℓ is wholly contained in a single grid cell.

(SSL) 2 small vertices, 1 large vertex

(SLL) 1 small vertex, 2 large vertices,

(LLL) 0 small vertices, 3 large vertices.

Next, we define an appropriate grid that helps us to detect triangles of type (SSS).

The Grid. Let G_1 be the grid whose cells are pairwise disjoint, axis-parallel squares with diameter $W/3$. The cells of G_1 partition the plane, and G_1 is aligned such that the origin $(0, 0)$ is a vertex of G_1. Observe that the cells of G_1 have side length $4\ell = W/(3\sqrt{2})$. We want to ensure that any triangle of type (SSS) in $\mathrm{D}(S)$ is completely contained in a single grid cell. For this, we make three copies G_2, G_3, and G_4 of G_1 and we shift them by 2ℓ (half the side length of a cell) in x-direction, in y-direction, and in both x- and y-direction, respectively. In other words, G_2 has $(2\ell, 0)$ as a vertex, G_3 has $(0, 2\ell)$ as a vertex, and G_4 has $(2\ell, 2\ell)$ as a vertex, see Figure 2.

Lemma 5 *Let Δ be a triangle formed by three vertices $a, b, c \in \mathbb{R}^2$ such that each edge of Δ has length at most 2ℓ. Then, there is a cell $\sigma \in G_1 \cup G_2 \cup G_3 \cup G_4$ with $a, b, c \in \sigma$.*

Proof. By assumption, we can enclose Δ with a square of side length 2ℓ. By construction, this square must be completely contained in cell of one of the four grids, see Figure 2. □

It follows immediately that any triangle of type (SSS) lies in a single grid cell, as desired.

Finding Triangles Inside Grid Cells. Next, we search for triangles that are completely contained in one grid cell. (These are not necessarily triangles of type (SSS).) For this, we go through all nonempty grid cells $\sigma \in \bigcup_{i=1}^{4} G_i$, and we search for a triangle in the disk graph $\mathrm{D}(S \cap \sigma)$ induced by the sites lying in σ. This can be done using Theorem 3. Since the grid cells have diameter $W/3$, any such triangle has weight at most W. Thus, we can return YES if a triangle is found. If we do not find any triangles, we can conclude by Lemma 5 that $\mathrm{D}(S)$ has no triangle of type (SSS). Since each site lies in a constant number of grid cells, and since we can compute the grid cells for a given site in $O(1)$ time, the total running time for this step is $O(n \log n)$. A simple volume argument yields the following lemma.

Lemma 6 *Let $\sigma \in \bigcup_{i=1}^{4} G_i$ be a nonempty grid cell, and suppose that σ does not contain a triangle. Then, σ contains $O(1)$ large sites.*

Proof. Suppose that σ contains at least 19 large sites. We partition σ into 3×3 congruent squares with side length $(4/3)\ell$. Each square has diameter $(4\sqrt{2}/3)\ell < 2\ell$, and by the pigeonhole principle, there is at least one square τ with at least $\lceil 19/9 \rceil = 3$ large sites. Since the associated radius of a large site is at least ℓ, the large sites in τ form a triangle in σ, contrary to our assumption. □

Triangles of Type (LLL). Now suppose that no triangle from $\mathrm{D}(S)$ is contained in a single grid cell. To find triangles of type (LLL), we iterate through all large sites $s \in S$. Let $\sigma \in G_1$ be the grid cell containing s. We define the *neighborhood* $N(\sigma)$ of σ as the 5×5 block of cells in G_1 that is centered at σ. Since the diameter of a grid cell is $W/3$, any pair $u, v \in S$ of sites that form a triangle with s of weight at most W must be contained in $N(\sigma)$. Let $S_\ell \subseteq S$ denote the large sites. By Lemma 6, we have $|N(\sigma) \cap S_\ell| = O(1)$. Thus, we can check in constant time whether s participates in a triangle of type (LLL). Hence, the total time to detect triangles of type (LLL) is $O(n)$.

Triangles of Type (SLL). This case is similar to the algorithm for triangles of type (LLL). This time, we iterate over all small sites $s \in S$. For each small $s \in S$, we check all pairs of large vertices contained in $N(\sigma) \cap S_\ell$, where $\sigma \in G_1$ is the cell containing s. This requires $O(n)$ time.

Triangles of Type (SSL). Now, consider an edge $e = ab \in D(S)$ between two small sites a and b. The edge e has length at most 2ℓ, and by construction it is completely contained in a single grid cell $\sigma \in \bigcup_{i=1}^{4} G_i$. To check if e participates in a triangle of

type (SSL) with weight at most W, we check all $O(1)$ large vertices in $S_\ell \cap N(\sigma)$.

We repeat this process for all edges of $\mathrm{D}(S)$ that lie in a single grid cell, and we claim that this requires $O(n \log n)$ time. Indeed, we know that no grid cell $\sigma \in \bigcup_{i=1}^{4} G_i$ contains a triangle (otherwise, we would have detected it previously). Then, by Lemma 1, it follows that $\mathrm{D}(S \cap \sigma)$ is plane, for all grid cells $\sigma \in \bigcup_{i=1}^{4} G_i$. In particular, $\mathrm{D}(S \cap \sigma)$ has $O(|S \cap \sigma|)$ edges and can be computed in time $O(|S \cap \sigma| \log |S \cap \sigma|)$. Since each vertex is contained in $O(1)$ grid cells, the total time to detect triangles of type (SSL) is $O(n \log n)$, as claimed.

Wrapping up. We summarize the previous discussion with the next lemma.

Lemma 7 *Let $\mathrm{D}(S)$ be a disk graph with n vertices, and let $W > 0$. We can decide in $O(n \log n)$ worst-case time whether $\mathrm{D}(S)$ contains a triangle of weight at most W.*

Finally, to solve the optimization problem, we employ the following general lemma due to Chan [3]. Let Π be a *problem space*, and for a problem $P \in \Pi$, let $w(P) \in \mathbb{R}$ be its *optimum* and $|P| \in \mathbb{N}$ be its *size*.

Lemma 8 (Lemma 2.1 in [3]) *Let $\alpha < 1$, $\varepsilon > 0$, and $r \in \mathbb{N}$ be constants, and let $\delta(\cdot)$ be a function such that $\delta(n)/n^{\varepsilon}$ is monotone increasing in n. Given any optimization problem $P \in \Pi$ with optimum $w(P)$, suppose that within time $\delta(|P|)$, (i) we can decide whether $w(P) < t$, for any given $t \in \mathbb{R}$, and (ii) we can construct r subproblems $P_1, \ldots, P_r$, each of size at most $\lceil \alpha |P| \rceil$, so that*

$$w(P) = \min\{w(P_1), \ldots, w(P_r)\}.$$

Then, we can compute $w(P)$ in total expected time $O(\delta(|P|))$.

For the first condition of Lemma 8, we use Lemma 7. For the second condition we construct four subsets $S_0, \ldots, S_3$ of S as follows: we enumerate the sites in S as $S = \{s_1, \ldots, s_n\}$, and we put each site s_i into all sets S_j with $j \not\equiv i \pmod 4$. Then, for any three sites $a, b, c \in S$, there is at least one subset S_j with $a, b, c \in S_j$. Hence, applying Lemma 8 with $\alpha = 3/4$, $\varepsilon = 1$, $r = 4$, and $\delta = O(n \log n)$ establishes Theorem 4.

4 Conclusion

Once again, disk graphs prove to be a simple and useful graph model where difficult algorithmic problems admit faster solutions. It would be interesting to find a *deterministic* $O(n \log n)$ time algorithm for finding a shortest triangle in a disk graph. Also, we are currently working on extending our results to the girth problem in *weighted* disk graphs and in directed transmission graphs.

Acknowledgments. We like to thank Günther Rote for helpful comments.

References

[1] M. de Berg, O. Cheong, M. van Kreveld, and M. H. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, third edition, 2008.

[2] S. Cabello and M. Jejčič. Shortest paths in intersection graphs of unit disks. *Comput. Geom. Theory Appl.*, 48(4):360–367, 2015.

[3] T. M. Chan. Geometric applications of a randomized optimization technique. *Discrete Comput. Geom.*, 22(4):547–567, 1999.

[4] H.-C. Chang and H.-I. Lu. Computing the girth of a planar graph in linear time. *SIAM J. Comput.*, 42(3):1077–1094, 2013.

[5] W. S. Evans, M. van Garderen, M. Löffler, and V. Polishchuk. Recognizing a DOG is hard, but not when it is thin and unit. In *Proc. 8th FUN*, pages 16:1–16:12, 2016.

[6] M. Fürer and S. P. Kasiviswanathan. Spanners for geometric intersection graphs with applications. *J. of Computational Geometry*, 3(1):31–64, 2012.

[7] F. Le Gall. Powers of tensors and fast matrix multiplication. In *Proc. 39th Internat. Symp. Symbolic and Algebraic Comput. (ISSAC)*, pages 296–303, 2014.

[8] A. Itai and M. Rodeh. Finding a minimum circuit in a graph. *SIAM J. Comput.*, 7(4):413–423, 1978.

[9] H. Kaplan, W. Mulzer, L. Roditty, P. Seiferth, and M. Sharir. Dynamic planar Voronoi diagrams for general distance functions and their algorithmic applications. In *Proc. 28th SODA*, pages 2495–2504, 2017.

[10] L. Roditty and V. Vassilevska Williams. Minimum weight cycles and triangles: Equivalences and algorithms. In *Proc. 52nd Annu. IEEE Sympos. Found. Comput. Sci. (FOCS)*, pages 180–189, 2011.

[11] H. Yu. An improved combinatorial algorithm for Boolean matrix multiplication. In *Proc. 42nd Internat. Colloq. Automata Lang. Program. (ICALP)*, pages 1094–1105, 2015.