Finding the Girth in Disk Graphs and a Directed Triangle in Transmission Graphs*

Haim Kaplan[1], Katharina Klost[2], Wolfgang Mulzer[2], and Liam Roditty[3]

1 **Tel Aviv University, Israel**
haimk@post.tau.ac.il
2 **Institut für Informatik, Freie Universität Berlin, Germany**
{kathklost,mulzer}@inf.fu-berlin.de
3 **Bar Ilan University, Israel**
liamr@macs.biu.ac.il

Abstract

Suppose we are given a set $S \subset \mathbb{R}^2$ of n point *sites* in the plane, each with an *associated radius* $r_s > 0$, for $s \in S$. The *disk graph* $D(S)$ for S is the undirected graph with vertex set S and an edge between s and t in S if and only if $|st| \leq r_s + r_t$, i.e., if the disks with radius r_s around s and with radius r_t around t intersect. The *transmission graph* $T(S)$ for S is the directed graph with vertex set S and an edge from s to t if and only if $|st| \leq r_s$, i.e., if the disk with radius r_s around s contains the site t.

We consider two problems concerning cycles in disk graphs and transmission graphs. First, we show that the *weighted girth* of a disk graph can be found in $O(n \log n)$ expected time, almost matching the bounds for planar graphs. Second, we present an algorithm for finding a *directed triangle* in a transmission graph in $O(n \log^2 n)$ time. Thus, these problems are much easier for disk and transmission graphs than for general graphs.

1 Introduction

Despite decades of research, many seemingly simple problems on graphs continue to stump researchers. For example, given a simple graph $G = (V, E)$, the best "combinatorial" algorithm to determine whether G contains a *triangle* (i.e., a cycle of length three) requires $O(n^3 \operatorname{polyloglog}(n)/\log^4 n)$ time [13], only a slight improvement over the trivial algorithm. Using fast matrix multiplication, the problem can be solved in $O(n^\omega)$ time, where $\omega < 2.37287$ is the matrix multiplication exponent [7,8]. For planar graphs, the problem becomes much easier: here, the unweighted *girth* (i.e., the length of the shortest cycle) can be found in linear time [5].

Two interesting graph classes that invite further study are *disk graphs* and *transmission graphs*. In both cases, we are given a set $S \subset \mathbb{R}^2$ of n point *sites* in the plane. Each site $s \in S$ has an *associated radius* $r_s > 0$ and an *associated disk* D_s centered around s with radius r_s. The *disk intersection graph* $D(S)$ for S is the undirected graph on S where two sites $s, t \in S$ are adjacent if and only if their associated disks intersect, i.e., if $D_s \cap D_t \neq \emptyset$. The edges of $D(S)$ are weighted according to the euclidean distance of their endpoints. The *directed transmission graph* $T(S)$ for S is the directed graph on S where there is an edge from a site s to a site t if and only if $t \in D_s$. Both graphs are well studied in computational geometry, since they serve as simple theoretical models for geometric sensor networks (see [9] and the

* Supported in part by grant 1367/2016 from the German-Israeli Science Foundation (GIF). W.M. supported in part by ERC StG 757609.

34th European Workshop on Computational Geometry, Berlin, Germany, March 21–23, 2018.
This is an extended abstract of a presentation given at EuroCG'18. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

references therein). Previously, Kaplan *et al.* [10] have studied the girth and triangles in disk intersection graphs. They showed that for a disk intersection graph with n sites, one can compute the unweighted girth in $O(n \log n)$ deterministic time and that one can find a shortest triangle in $O(n \log n)$ expected time. The running time for the unweighted girth is optimal in the algebraic decision tree model [12]. We extend the results of Kaplan *et al.* [10] to the weighted girth in disk graphs and to the triangle problem in transmission graphs.

2 Weighted girth of a disk graph

In this section we consider the problem of finding the *weighted* girth of a disk intersection graph. First, we describe an algorithm that, given a vertex and an abstract graph with some restrictions, finds the shortest cycle in the graph containing that vertex. This algorithm is then used as a subroutine in Section 2.2 to compute the weighted girth of a disk intersection graph.

2.1 Finding the shortest cycle containing a given vertex

Let $G = (V, E)$ be an abstract graph with nonnegative edge weights, such that all shortest paths and all cycles in G have pairwise distinct lengths and such that for all edges $uv \in E$, the shortest path from u to v is the edge uv. Let $|V| = n$ and $|E| = m$. We present an algorithm that, given G and a vertex $s \in V$, computes a shortest cycle in G containing s. A simple randomized algorithm for this problem was presented by Yuster [14]. We give a deterministic algorithm.

We run Dijkstra's algorithm to determine the shortest path tree T for s in G in $O(n \log n + m)$ time. Then, we traverse T to find for each $v \in V$ the vertex $b[v] \in V$ that comes after s on the shortest path from s to v. This takes $O(n)$ steps. Finally, we iterate over all edges $e \in E$ that do not occur in T. For each such edge $e = uv$, we check if $b[u] \neq b[v]$. If this is the case, then e closes a cycle in T that contains s. We determine the length of this cycle in $O(1)$ time, using the shortest path distances and the length of e. We return the shortest such cycle. Overall, the algorithm requires $O(n \log n + m)$ time. The following lemma shows the shortest cycle in G that contains s is of the desired form.

▶ **Lemma 2.1.** *The shortest cycle in G that contains s consists of two paths in the shortest path tree T of s, and one additional edge.*

Proof. Let $C = (v_0 = s), v_1, v_2, \ldots, v_{\ell-1}, s$ be the shortest cycle in G containing s, where all vertices v_i are pairwise distinct and $\ell \geq 3$. For $v_i \in C$, let $d_1(v_i)$ be the length of the path $s, v_1, \ldots, v_i$, and let $d_2(v_i)$ be the length of the path $v_i, v_{i+1}, \ldots, s$. Let $\pi(v_i)$ denote the shortest path from s to v_i, and let $|v_i v_{i+1}|$ be the length of the edge $v_i v_{i+1}$.

Suppose that C is not of the desired from. Let v_k, v_{k+1} be the edge on C with $d_1(v_k) < |v_k v_{k+1}| + d_2(v_{k+1})$ and $d_2(v_{k+1}) < d_1(v_k) + |v_k v_{k+1}|$. By our assumptions on G, the edge $v_k v_{k+1}$ exists and $k \neq 0, \ell - 1$. We distinguish two cases.

First, suppose that $\pi(v_k) \cap \pi(v_{k+1}) = \{s\}$. Consider the cycle C' given by $\pi(v_k)$, the edge $v_k v_{k+1}$, and $\pi(v_{k+1})$. Since $s \neq v_k, v_{k+1}$ and since the edge $v_k v_{k+1}$ does not appear on $\pi(v_k)$ and $\pi(v_{k+1})$, it follows that C' is a proper cycle. Furthermore, by assumption, C' is strictly shorter than C, because $\pi(v_k)$ is shorter than $d_1(v_k)$ or $\pi(v_{k+1})$ is shorter than $d_2(v_{k+1})$. This contradicts our choice of C.

Second, suppose that $|\pi(v_k) \cap \pi(v_{k+1})| \geq 2$. Since $\pi(v_k)$ and $\pi(v_{k+1})$ are shortest paths, their intersection is a prefix of each path. By the assumption on G, at least one of $v_1, v_{\ell-1}$ is not in $\pi(v_k) \cup \pi(v_{k+1})$. Without loss of generality, this vertex is v_1. Let $j \geq 1$ be the smallest index such that $v_j \in \pi(v_k) \cup \pi(v_{k+1})$. We have $j \in \{2, \dots, k\}$. Consider the cycle C' that starts at at s, follows C along $v_1, v_2, \dots$ up to v_j, and then returns along $\pi(v_k)$ or $\pi(v_{k+1})$ to s. By construction, C' is a proper cycle. Furthermore, $C' \neq C$, because even if $j = k$, the path $\pi(v_k)$ does not use the edge $v_k v_{k+1}$ due to the choice of k. Finally, C' is strictly shorter than C, because the second part of C' from v_j to s follows a shortest path and is thus strictly shorter than $d_2(v_j)$. Again, C' contradicts our choice of C. ◀

2.2 Computing the girth

We describe an algorithm to compute the weighted girth of a disk intersection graph $D(S)$. First, we find the shortest triangle in the disk graph $D(S)$. This takes $O(n \log n)$ expected time using the algorithm of Kaplan *et al.* [10].

If $D(S)$ contains no triangle, then it is plane [6] [10, Lemma 1]. Thus, we can explicitly construct $D(S)$ with a sweep line algorithm in time $O(n \log n)$ and determine the girth of this weighted graph with an appropriate algorithm for planar graphs.

If $D(S)$ contains a triangle, its length W can serve as an upper bound for the length of the shortest cycle in $D(S)$. We use the same partition of S into *large* and *small* sites as Kaplan *et al.* [10]. Namely, we set $\ell = W/12\sqrt{2}$, and we call all sites with radius at least ℓ *large* and the remaining sites *small*. Still following Kaplan *et al.*, we cover the plane with four overlapping axis parallel grids G_1, G_2, G_3, and G_4. The *open* grid cells have side length 4ℓ, and the grids are defined such that the points $(0, 0)$, $(2\ell, 0)$, $(0, 2\ell)$ and $(2\ell, 2\ell)$ are vertices of G_1, G_2, G_3, and G_4, respectively.

We want to find the shortest cycle with at least four vertices and with length at most W. First, we consider cycles that consist only of small sites. From the choice of ℓ, it follows that there is no triangle consisting only of small sites: otherwise, there would be a triangle of length at most $3 \cdot 4\ell < W$, contradicting the choice of W. Thus, the subgraph D' of $D(S)$ induced by the small vertices is plane [6] [10, Lemma 1]. As before, we can compute D' and its girth directly, using a plane sweep and known results for planar graphs. Let Δ_1 be this girth.

Finally, we consider cycles with at least one large site. By the choice of ℓ, every triangle that is completely contained in an open grid cell has length less than W. Since there are no such triangles in $D(S)$, we can apply Lemma 6 of Kaplan *et al.* [10] to conclude that each grid cell contains $O(1)$ large sites.

By the triangle inequality, in a cycle of length less than W, the maximum distance between any two sites is less than $W/2$. Thus, any such cycle containing a given site $s \in S$ completely lies in a rectangle with side length W around s. This corresponds to a 7×7 neighborhood $N(\sigma)$ around a grid cell σ containing s. Since $N(\sigma)$ consists of $O(1)$ cells and since each cell contains $O(1)$ large sites, there are $O(1)$ large sites in $N(\sigma)$.

We iterate over all grid cells σ. For each σ, we consider all large sites $s \in \sigma$. As discussed, we must find the shortest cycle containing s in the subgraph $D(S_\sigma)$ of $D(S)$ induced by the sites $S_\sigma = S \cap N(\sigma)$. Suppose $D(S_\sigma)$ contains n'_σ small sites and n''_σ large sites. Since the graph induced by the small sites is plane and since $n''_\sigma = O(1)$, the graph $D(S_\sigma)$ has $O(n_\sigma)$ edges. This means that we can explicitly compute $D(S_\sigma)$ in time $O(n_\sigma \log n_\sigma)$ and apply the algorithm from Section 2.1 in order to compute the shortest cycle containing s in time $O(n_\sigma \log n_\sigma)$. Let Δ_2 be the length of the shortest cycle encountered in this step. If we also want to output the shortest cycle in the end, we also store a pointer to σ and s. Since each small site is involved only in a constant number of neighborhoods, we have: $\sum_{i=1}^{4} \sum_{\sigma \in G_i} n_\sigma = O(n)$, and thus the overall running time of this step is $O(n \log n)$. In the end, we return $\min\{W, \Delta_1, \Delta_2\}$. Thus, we obtain the following theorem:

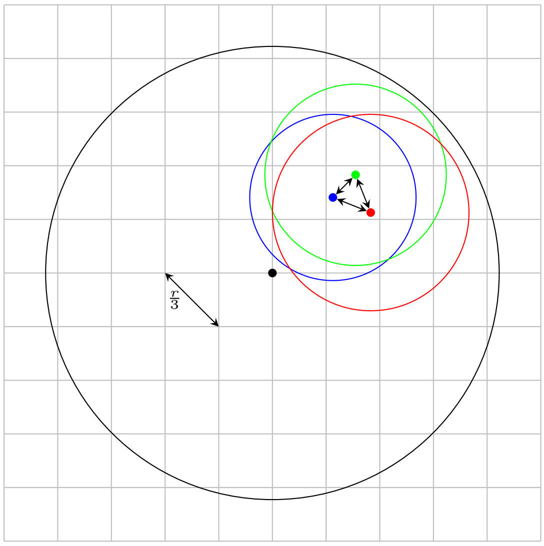

Figure 1 Three disks with associated radius at least $r/3$ are in the same grid cell form a clique

▶ **Theorem 2.2.** *Given a set S of n point sites in $\mathbb{R}^2$ with associated radii, we can compute the weighted girth of $D(S)$ in $P(n) + O(n \log n)$ expected time, where $P(n)$ is the time needed to compute the weighted girth of a planar graph with real edge weights.*

▶ **Corollary 2.3.** *Using the algorithm of Łącki and Sankowski [11], we can compute the weighted girth of a disk graph in $O(n \log n)$ expected time.*

3 Directed triangles in transmission graphs

In this section we consider directed triangles in transmission graphs. Given a disk transmission graph $T(S)$ we want to decide, if this graph contains at least one directed triangle.

First we consider the following structural lemma. It gives a condition on the disks that will help us find certain triangles.

▶ **Lemma 3.1.** *Let D be a disk of radius r. If D contains more than 152 sites with associated radius at least $r/3$, then $T(S)$ has a directed triangle.*

Proof. We cover D with a grid, where each cell has diameter $r/3$. Each grid cell has side length $\sqrt{2}r/6$, so we need at most 76 such cells (see Figure 1). By our choice of the diameter, for each site $s \in D$ with $r_s \geq r/3$, the associated disk D_s completely covers the grid cell that contains s.

If D contains more than 152 sites with associated radius at least $r/3$, the pigeonhole principle shows that one grid cell contains at least three such sites. Since the corresponding disks contain the complete grid cell, these three sites form a directed clique in $T(S)$. In particular, there is a directed triangle. ◀

Now we show how the condition of Lemma 3.1 can be checked for a given disk transmission graph. This will later be the first part of the algorithm to find a triangle.

▶ **Lemma 3.2.** *In $O(n \log^2 n)$ time, we can check whether S contains a site s such that D_s contains more than 152 sites with associated radius at least $r_s/3$. Furthermore, if every disk contains at most 152 such sites, we can find all these sites in $O(n \log^2 n)$ time.*

Proof. We use the halfspace range reporting structure by Afshani and Chan [1]. This structure allows us to preproccess a planar n-point set $P \subset \mathbb{R}^2$ in $O(n \log n)$ time so that for any query point $q \in \mathbb{R}^2$ and for any $k \in \{1, \dots, n\}$, we can find the k nearest neighbors of q in time $O(\log n + k)$ [4]. We will actually need a semi-dynamic version of this data structure that supports insertions. For this, we apply the classic Bentley-Saxe transform to obtain a structure with $O(\log n)$ amortized insertion time and $O(\log^2 n + k \log n)$ worst-case query time [3].

We consider the sites by decreasing radius. Our range reporting data structure will always contain all sites with associated radius at least $r_s/3$, where s is the current site. When processing $s \in S$, we first insert all sites with radius at least $r_s/3$ that are not yet present in the data structure. Then, we query the 153 nearest neighbors of s in the structure, and we determine which of them lie in D_s. If all of them do, then $T(S)$ contains a triangle. Otherwise, we store this set with s. One such query takes $O(\log^2 n)$ time, for a total of $O(n \log^2 n)$ time. The total time to sort the sites by descending radius and for inserting them into the structure is $O(n \log n)$. The claim follows. ◀

With Lemma 3.2 we now know how to check if a graph contains a triangle because of the condition of Lemma 3.1. Furthermore Lemma 3.2 allows us to find for each site s all sites with radius at least $r_s/3$, contained in D_s. In the next lemma we show how, given this information, we can find a triangle in a transmission graph were no disk obeys the condition of Lemma 3.1.

▶ **Lemma 3.3.** *Suppose we are given a set S of n sites such that for each $s \in S$, the disk D_s contains at most 152 sites with associated radius at least $r_s/3$ and such that these sites are known. We can find a directed triangle in $T(S)$ in $O(n \log^2 n)$ time, if it exists.*

Proof. We need a static nearest neighbor data structure for the additively weighted euclidean distance. Using an appropriate Voronoi diagram, this can be done with $O(n \log n)$ preprocessing time and $O(\log n)$ query time [2]. We will have queries of the following form: given a query point $q \in \mathbb{R}^2$, find the nearest site to q whose radius lies in a given interval. For this, we build a perfect binary search tree on S, sorted by radius. In each inner vertex v of the tree, we store an additively weighted Voronoi diagram for all disks in the subtree of v. The weight for each site s is $-r_s$.

This tree can be constructed in $O(n \log^2 n)$ time in bottom up fashion. Given a query point q and a radius range (r, r'), we must perform $O(\log n)$ queries to the Voronoi diagrams, since we can follow the paths to r and r' and query all the diagrams of tree vertices whose intervals are completely contained in (r, r'). Thus, the query time is $O(\log^2 n)$.

We iterate over the sites by decreasing radius. We will check for each site $s \in S$ if it is the site with smallest radius in a directed triangle in $T(S)$. Suppose there is such a triangle of the form $s \to t \to u \to s$. Thus, we have $r_s \le r_t$ and $r_s \le r_u$. Since $t \in D_s$, there are at most 152 known candidates for t. Having fixed such a candidate t, there are two cases regarding u:

1. $r_u \ge r_t/3$: in this case, having fixed t, there are only 152 known candidates for u, and all of them can be checked in $O(1)$ time.
2. $r_u < r_t/3$: by definition, we have $s \in D_u$. From this, it follows that that $D_u \subset D_t$. Thus, to find a triangle of the desired kind, it is enough to find any site u with $r_u < r_t/3$ and

with $s \in D_u$. This can be done by finding the nearest site to s with radius in $(r_s, r_t/3)$. As explained, this takes $O(\log^2 n)$ time.

Since we iterate over all sites, this results in a total running time of $O(n \log^2 n)$. ◀

Now we can combine the Lemma 3.2 and Lemma 3.3 to get the following theorem:

▶ **Theorem 3.4.** *Given a set S of n point sites in $\mathbb{R}^2$ with associated radii, we can find a directed triangle in the associated directed transmission graph $T(S)$ in time $O(n \log^2 n)$.*

Proof. First we use the procedure described in Lemma 3.2 in time $O(n \log^2 n)$. If it finds a triangle, we return yes. Otherwise we use the resulting information, to apply the algorithm from Lemma 3.3. This results in an algorithm with $O(n \log^2 n)$ running time. ◀

References

1. Peyman Afshani and Timothy M. Chan. Optimal halfspace range reporting in three dimensions. In *Proc. 20th Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 180–186, 2009.
2. Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi Diagrams and Delaunay Triangulations*. World Scientific Publishing, 2013.
3. Jon Louis Bentley and James B. Saxe. Decomposable searching problems I: Static-to-dynamic transformation. *J. Algorithms*, 1(4):301–358, 1980.
4. Timothy M. Chan and Konstantinos Tsakalidis. Optimal deterministic algorithms for 2-d and 3-d shallow cuttings. *Discrete Comput. Geom.*, 56(4):866–881, 2016.
5. Hsien-Chih Chang and Hsueh-I Lu. Computing the girth of a planar graph in linear time. *SIAM J. Comput.*, 42(3):1077–1094, 2013.
6. William S. Evans, Mereke van Garderen, Maarten Löffler, and Valentin Polishchuk. Recognizing a DOG is hard, but not when it is thin and unit. In *Proc. 8thInternat. Conf. Fun w. Algorithms (FUN)*, pages 16:1–16:12, 2016.
7. François Le Gall. Powers of tensors and fast matrix multiplication. In *Proc. 39th Internat. Symp. Symbolic and Algebraic Comput. (ISSAC)*, pages 296–303, 2014.
8. Alon Itai and Michael Rodeh. Finding a minimum circuit in a graph. *SIAM J. Comput.*, 7(4):413–423, 1978.
9. Haim Kaplan, Wolfgang Mulzer, Liam Roditty, and Paul Seiferth. Spanners and reachability oracles for directed transmission graphs. In *Proc. 31st Int. Sympos. Comput. Geom. (SoCG)*, pages 156–170, 2015.
10. Haim Kaplan, Wolfgang Mulzer, Liam Roditty, and Paul Seiferth. Finding triangles and computing the girth in disk graphs. In *Proc. 33rd European Workshop Comput. Geom. (EWCG)*, pages 205–208, 2017.
11. Jakub Łącki and Piotr Sankowski. Min-cuts and shortest cycles in planar graphs in $O(n \log \log n)$ time. In *Proc. 19th Annu. European Sympos. Algorithms (ESA)*, pages 155–166, 2011.
12. Valentin Polishchuk. Personal communication. 2017.
13. Huacheng Yu. An improved combinatorial algorithm for Boolean matrix multiplication. In *Proc. 42nd Internat. Colloq. Automata Lang. Program. (ICALP)*, pages 1094–1105, 2015.
14. Raphael Yuster. A shortest cycle for each vertex of a graph. *Inform. Process. Lett.*, 111(21-22):1057–1061, 2011.