Recognizing a DOG is Hard, but not when it is Thin and Unit

William Evans[1], Mereke van Garderen[2], Maarten Löffler[3], and Valentin Polishchuk[4]

1 **University of British Columbia,Canada**
`will@cs.ubc.ca`
2 **University of Konstanz, Germany**
`mereke.van.garderen@uni-konstanz.de`
3 **Utrecht University, the Netherlands**
`m.loffler@uu.nl`
4 **Linköping University, Sweden**
`valentin.polishchuk@liu.se`

Abstract

We define the notion of *disk-obedience* for a set of disks in the plane and give results for *disk-obedient graphs* (DOGs), which are disk intersection graphs (DIGs) that admit a planar embedding with vertices inside the corresponding disks. We show that in general it is hard to recognize a DOG, but when the DIG is thin and unit (i.e., when the disks are unit disks), it can be done in linear time.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems]: Geometrical problems and computations; Routing and layout

Keywords and phrases graph drawing, planar graphs, disk intersection graphs

Digital Object Identifier 10.4230/LIPIcs.FUN.2016.16

1 A Concise Introduction to Disk-Obedience

Consider a set of disks in the plane. The *disk intersection graph (DIG)* induced by the set has a vertex for every disk and an edge between two vertices whose disks intersect. We will often identify the DIG vertices with the disks that induce the graph, and speak, e.g., about "disks of the DIG". An *embedding* of the DIG assigns a location in the plane for each vertex of the DIG; the graph edges are straight-line segments connecting their endpoints. An embedding is *planar* if the edges do not intersect except at common endpoints.

The question we consider is whether a DIG can be embedded so that its vertices obey the location constraints imposed by the disks, and the embedding is planar. The main definitions in this paper are those of *obedience* and a *DOG*:

▶ **Definition 1.** An embedding of a DIG is called *disk-obedient* if each vertex is contained in its corresponding disk.

▶ **Definition 2.** A DIG is a *disk-obedient graph (DOG)* if it admits a planar disk-obedient embedding.

We study computational complexity of the following problem:

Is the DIG a DOG?

That is, for a given DIG, we want to find out whether or not it has a planar, disk-obedient embedding.

1.1 Motivation and Related Work

DIGs have been extensively studied in the literature due to their wide applications in a variety of domains. In particular, in wireless sensor networks (WSNs), the *unit-disk graph (UDG)* has been the prevailing basic model for communication and sensing, and many generalizations of the UDGs (civilized graphs [19], quasi-UDGs [21, 5, 14], Vietoris-Rips complex [30, 13, 15], bisectored UDGs [27], etc.) have been introduced. The most natural generalization is, of course, to disks of two radii (modeling, e.g., two types of communication devices in the WSN or the asymmetry in sending/receiving range [12, 11]) or of arbitrary sizes. Embedding DIGs in the plane (and/or finding a *realization* of a DIG) was viewed as an important practical problem in WSN and ad hoc networks, because it translates the network connectivity information into virtual coordinates [20] and can serve as the first step in aiding topology extraction [18, 31], which may be subsequently used in geometric routing algorithms (e.g., GOAFR [22] and GFG/GPSR [6]); a planar embedding may be preferred since it provides the cleanest picture of the network.

In graph drawing and network visualization, minimizing crossings in the embedding is the central investigated question. The problem in this paper is related to the so called *Anchored* Planar Graph Drawing (AGD) (shown to be NP-hard in [1]): Given a planar graph G and an initial placement for its vertices, produce a planar straight-line drawing of G such that each vertex is at distance at most 1 from its original position. AGD was motivated by the scenario in which the disks represent cities (disk sizes represent city sizes) and the graph shows some relation between cities; in the drawing, it makes sense to keep a city vertex inside the city. Our DIG/DOG problem is different from AGD since for us, the DIG G whose planar disk-obedient drawing is sought, is fully defined by the disks (G is the DIG), while in AGD, G can be an arbitrary planar graph given in the input *in addition* to the disks. Moreover, [1] studied AGD only for *non-overlapping* disks (as observed in [1], if the disks are allowed to overlap, AGD becomes as difficult as the long standing open problem of strip planarity testing [2]; however, as also noted in [1], if the disks can have different radii and are allowed to overlap, then AGD is trivially hard by reduction from planar drawing extension [28]); for us, the disks must overlap, for otherwise the DOG recognition problem is void (G is empty). Placing vertices inside the disks of a DIG to produce C-oriented planar drawings of bounded-degree graphs was studied in [17]; however, the disks in [17] only touch (no overlap), making crossings impossible.

Our particular motivation for introducing disk-obedience comes from dealing with data uncertainty [7, 26, 25, 8, 24] which arises, e.g., due to objects moving (possibly with different and/or unknown speeds) from some last-known locations (starting possibly at different and/or unknown times). It is often known which objects could currently be close to each other (potential incidents / dog fights / airplanes in conflict / criminal activities), which defines the DIG. Now, one wants to show this graph as clearly as possible, which means (by GD tradition) to draw it without crossings.

Just for FUN, we experimented with turning our problem into a game of trying to embed a given DIG in a planar disk-obedient way. The game can be programmed in interactive geometry software like GeoGebra. Initial experience indicates no concrete reason why any of the variants would not make for an engaging game that should totally become the next hit on the iStore. It has been a FUN tradition to analyze computational hardness of various games, and for once, our paper gives the computational complexity treatment to a game *before* the game storms the market. We invite the reader to play with our simple sample instance at `http://tube.geogebra.org/m/BzVYK21A`; it is not hard, but requires a tiny bit of thinking outside the circle (at least a few 9-year-old testers have politely found it non-trivial).

1.2 Paper Outline

Section 2 gives initial observations on disk-obedience, proving that a *shallow* DIG (a DIG with depth 2) is always a DOG. It follows that triangle-free DIGs are DOGs, i.e., that triangles are the only problematic places around which a DIG may be disk-disobedient. This motivates us to look more closely at triangles in DOGs. In Section 3 we show that in general, for DIGs which may have triangles close to each other, testing disk-obedience is NP-hard. In Section 4 we prove that for unit DIGs (i.e., DIGs for unit disks) having close-by triangles is essential for the hardness: we define thin unit DIGs whose triangles are well separated, and give a linear-time algorithm to recognize a thin unit DOG. The biggest question we leave unsolved is the hardness of recognizing general unit DOGs, i.e., deciding whether a given unit DIG is a DOG; we discuss it (and possible related positive results) in Section 5.

2 DIGs with Depth 2 are DOGs

We assume that our DIGs are connected. Of course, only planar graphs can have planar disk-obedient embeddings. However, not all of them do, as Figure 1 shows. Define the *depth of a point* in the plane as the maximum number of disks containing the point, and the *depth of a DIG* as the maximum number of disks having a point in common. If a DIG has depth 5 or larger, it cannot be a DOG since K_5 is not planar. Depth-3 DIGs may be DOGs (e.g., 3 disks that coincide) or not (as Fig. 1 shows); similarly, depth-4 DIGs may be DOGs (e.g., 4 disks that coincide) or not (just add a disk to Fig. 1 to create a depth-4 point – adding more disks to a DIG that is not a DOG will not make a planar disk-obedient embedding possible).

▶ **Observation 3.** Depth-2 DIGs are DOGs.

Proof. Let $D_1, D_2, \ldots, D_n$ be an arrangement of disks with depth at most two, where D_i has center c_i. Embed vertices of the DIG at the disk centers. Suppose edges c_1c_3 and c_2c_4 intersect at point x (Fig. 2), and suppose wlog that x lies in D_1 and D_4. Let $c_1a = D_1 \cap c_1c_3$ and $c_4b = D_4 \cap c_2c_4$. Since D_1 and D_3 intersect, $a \in D_3$ and since D_2 and D_4 intersect $b \in D_2$. Since c_1a and c_4b intersect at x, by the triangle inequality, $|c_1b| + |c_4a| < |c_1a| + |c_4b|$, which implies either $b \in D_1$ or $a \in D_4$ and thus b or a is contained in three disks; a contradiction. ◀

▶ **Corollary 4.** *DIGs without triangles are DOGs.*

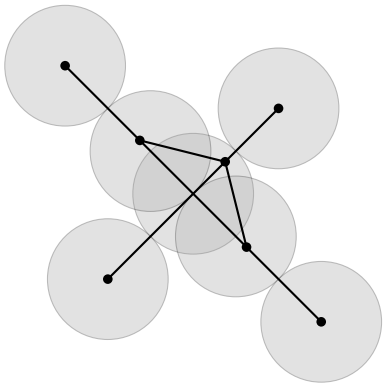

Figure 1 An example of a planar DIG that has no planar disk-obedient embedding.

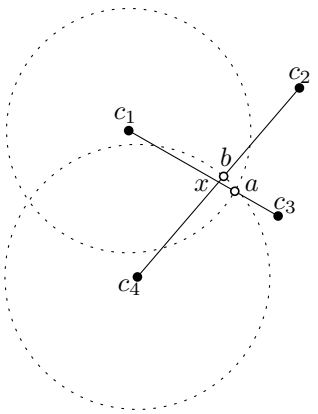

Figure 2 Either a or b is in three disks.

3 Recognizing a DOG is Hard

In this section, we prove that deciding whether a DIG is a DOG is NP-hard. We reduce from planar 3-SAT [23].

Let S be a planar 3-SAT formula. We can lay out its variable-clause graph in a planar way. Globally speaking, we will emulate every edge leading from a variable to a clause by a *variable chain*: a set of disks such that their DIG has exactly two possible planar disk-obedient embeddings. One of these embeddings will encode the value `true`; the other will encode the value `false`. Then, at each variable, we add a *variable gadget*: a configuration of disks that ensures all chains belonging to the same variable have to be in the same state. At each clause, we add a *clause gadget*: a configuration of disks that ensures at least one of the three incident chains must be in the correct state (either `true` or `false`, depending on the literal in the clause). Then, the entire configuration of disks will have a planar disk-obedient embedding if and only if S can be satisfied.

Variable chains

A variable chain is a sequence of kissing unit disks (from now on called *big disks*), together with a number of triples of disks (*small disks*) overlapping each kissing point, as shown in Figure 3a. The embedding of the vertices corresponding to the big disks, together with the edges connecting the vertices, will be called the *variable path*. The embedding of the vertices corresponding to the small disks in a triple, together with the edges connecting the vertices, will be called a *small path*. In a planar disk-obedient embedding a small path should not intersect the variable path. This means that the whole small path will need to be embedded either fully below or fully above the variable path. To be able to embed the small path below (resp., above) the variable path, the variable path must have points of all three small disks below (resp., above) the path. That is, the variable path must intersect either the top or the bottom small disk of a triplet. Both are possible: see Figures 3b and 3c. Moreover, by making the radii of the small disks sufficiently close to the radius of the big disks, we can ensure that two neighbouring triples of small paths cannot be in opposite configurations simultaneously. This way, the DIG of a variable chain will have exactly two distinct planar disk-obedient embeddings.

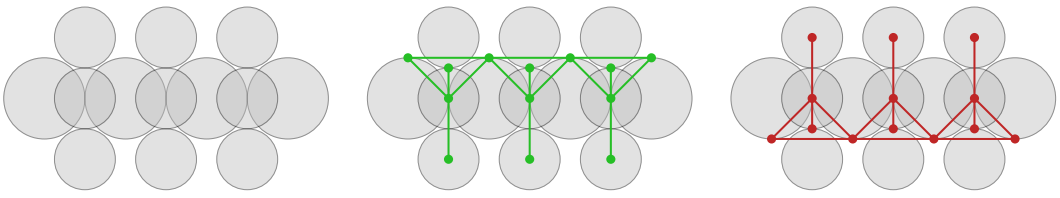

Figure 3 A variable chain and its two valid states.

Alignment gadgets

We use the *variable alignment gadget* to force all chains emanating from the same variable to have the same value. In principle, the number of times a variable appears in a 3-SAT instance is unlimited (though of course bounded by the number of clauses). We will not attempt to design a generic gadget working for an arbitrary-degree variable, but instead

replace each variable vertex by a tree of degree-3 nodes and install the alignment gadget at each node. This way, using multiple gadgets each aligning only three chains, we can align as many chains as required.

To align three chains, we let the three big disks at the ends of the chains kiss. Centered on this location, we add a claw consisting of four small disks. Figure 4a shows the configuration. If all three chains are in the same state, then there exists a planar disk-obedient embedding of the entire alignment gadget, as can be seen in Figures 4b and 4c. It can be seen by inspection that in any other combination of states, there is no planar disk-obedient embedding.

The alignment gadget can also be used as an inverter to make the chain arrive at the clause in the correct orientation (in the inverter, one of the three variable chains will end without connecting to a clause).

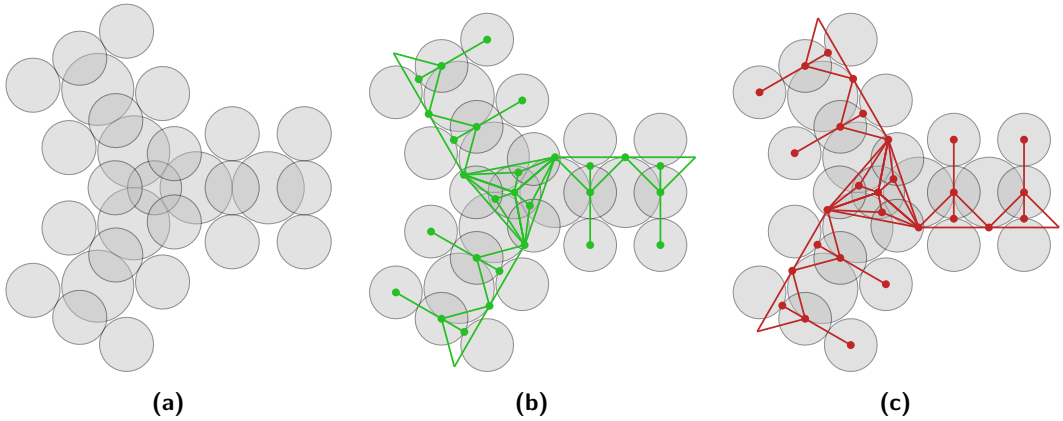

Figure 4 An alignment gadget and its two valid states.

Clause gadgets

Finally, we design *clause gadgets* for the clause vertices of S. In a clause, three variable chains end, and we place them so that they all kiss a single *clause disk* of the same radius as the big disk in the variable chains. The chains make an angle of 150° before they kiss. On top of this, we add a number of much smaller disks (from now on, *tiny disks*), as depicted in Figure 5a. The tiny disks essentially form walls that cannot be intersected by any edge in the solution embedding. We place these walls in such a way that if a variable chain is in the wrong state, the only way for the final variable disk to connect to the clause disk is if the vertex of the clause disk is placed on a specific line (or, in fact, narrow cone) determined by the location of the final vertex of the variable path and a gap in the wall. We generate three such lines—one for each variable—and make sure the three lines do not pass through a common point. Now, if at most two of the three variables are in the wrong state, we can still place the clause vertex, but if all three are in the wrong state, this is no longer possible and there is no planar disk-obedient embedding.

Thus, S can be satisfied if and only if there is a planar disk-obedient embedding, i.e., if the DIG built from S is a DOG.

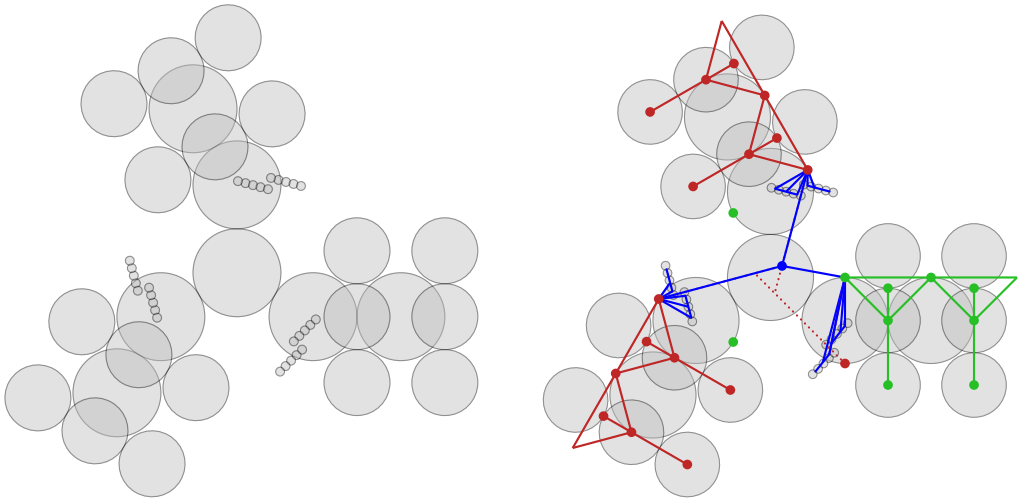

Figure 5 A clause gadget and a possible satisfied state.

4 Recognizing Thin Unit DOGs is Easy

In this section, we study disk-obedience for unit DIGs (i.e. unit disk graphs). We prove that (in accordance with common sense) if potential disk-disobedience spots are distant, each of them can be handled separately.

We start by defining some terms used later in the section:

▶ **Definition 5.** The *hippodrome* $H(A, B)$ of two intersecting disks A, B is the union of all possible segments that could realize the edge AB (i.e., the convex hull of $A \cup B$). Each of the two connected components of the difference $H(A, B) \setminus (A \cup B)$ is called a *delta* and denoted by $\Delta(A, B)$.

Suppose that vertices of a DOG are embedded at their disks centers, and (alas!) the embedding is not planar. Recall from Section 2 that each crossing must involve a triangle (by the proof of Observation 3, edges with endpoints in centers of depth-2 disks do not intersect). Let abc be a triangle in the DOG; let A, B, C be the corresponding disks. If an edge pq of the DOG intersects the edge ab, then the disks P and Q of p and q resp. are not too far from A and B; in fact, they must be within some constant distance < 8 from the triangle (because the hippodromes $H(A, B)$ and $H(P, Q)$ intersect, and a hippodrome has diameter < 4). But in any DOG, the number of disks within a constant distance from any point cannot be too large, or else a depth-5 point appears. Thus, there are only a constant number of disks within distance k of the triangle for any constant k.

Before embedding the (potential) DOG, we color its disks red and green: the disks that are closer than k to any triangle (including all the triangle disks themselves) are *red*, and the others are *green* (the same color applies to both a disk and its vertex). Let monochromatic edges inherit the color from their endpoints. Inspired by work on thin grids, suggesting that problems on them tend to be easier than on general grids [3, 4], we define *thin* DIGs as those composed from triangles connected by (longish) *isolated* paths such that far from the triangles the different paths do not come close to each other. A path in a DIG is *isolated* if the hippodrome of any adjacent disks A and B on the path may only intersect disks of vertices adjacent to A or B.

▶ **Definition 6.** A DIG is *thin* if

- the distance between any two triangles is larger than $3k$ ($k = 16$ suffices)
- removal of all disks within distance $k/2$ of a triangle decomposes the graph into a set of isolated paths.

▶ **Definition 7.** A *thin DOG* is a thin DIG admitting a planar disk-obedient embedding (Fig. 6).

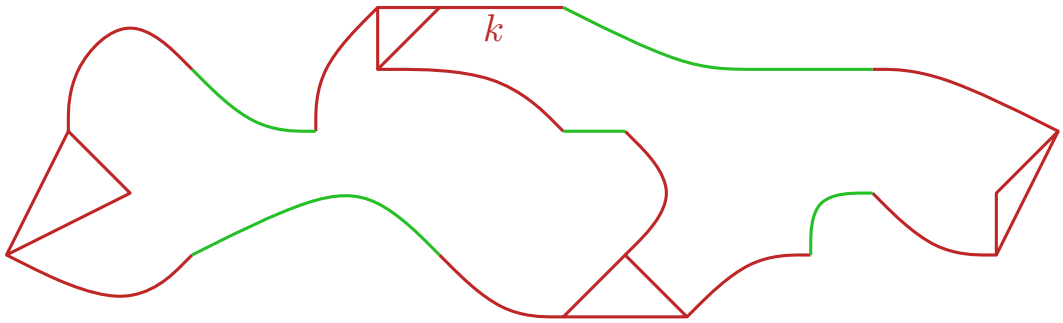

Figure 6 A schematic picture of a thin DOG. Our algorithm places green vertices at the disk centers, and works on each red component independently.

In what follows, we consider only *connected* thin DIGS, since a DIG is a DOG if and only if its components are DOGs. The crucial properties of thin DIGs, following from the definition, are that all green vertices have degree ≤ 2 and are adjacent to vertices (red or green) of degree ≤ 2 (since they are far from any triangle); any pair of adjacent vertices that are close (< 8) to a green vertex are part of an isolated path (thus their hippodrome does not intersect the green disk or its adjacent disks unless they are part of the same path); and no green vertex can be on a short cycle (length $< k/2$) in the DIG (otherwise the cycle would be disconnected from the rest of the graph, since every vertex close to a green vertex has degree ≤ 2).

Given a thin DIG, we check whether each red triangle and its red component is a DOG and obtain a disk-obedient embedding for it. Since there is only a constant number of vertices per red component (otherwise we know the red component is not a DOG), this can be done in constant time (in an appropriate model of computation, like realRAM) – e.g., by formulating DOG recognition as a mathematical program whose variables are the coordinates of the vertices, and checking the program for feasibility. If no "local" red DOG can be found, this certifies that the whole DIG is not a DOG either. Otherwise, if all red components are DOGs, we claim that the whole DIG is also a DOG. To that end, we embed the green vertices at their disk centers and resolve any edge intersections this might create. In what follows, we consider the different cases depending on the colors of the endpoints of intersecting edges. Note that all four of these endpoints cannot be red.

First, suppose there is a green edge g_1g_2 that intersects an edge ab (a and b can be of arbitrary colors); let G_1, G_2, A, B be the corresponding disks, and let $x = g_1g_2 \cap ab$ be the intersection point. Since g_1, g_2 are at the disks centers, $x \in G_1 \cup G_2$. Consider the two cases depending on where x is w.r.t. $A \cup B$:

$x \in A \cup B$: Then $(G_1 \cup G_2) \cap (A \cup B) \neq \emptyset$, and hence there exists an edge (say, g_1a) between a green vertex and one of a, b. We move a to x (Fig. 7, left), removing the intersection and, since the new edges are sub-segments of the old, creating no new intersection.

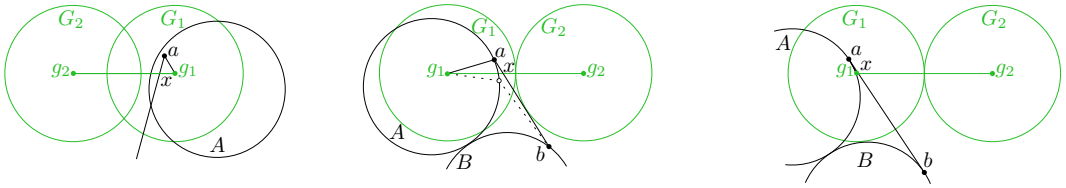

Figure 7 Left: If $x \in A \cup B$, move a to x. Middle: The new edges are dotted and the new location for a is the hollow circle. Right: $x \in \Delta(A, B), \{g_1, g_2\} \cap (A \cup B) = \emptyset$: move g_1 to x.

$x \in \Delta(A, B)$: There are 2 subcases:

- If one of the green centers is inside $A \cup B$ (say, $g_1 \in A$), we rotate g_1a around g_1 until a has moved over g_1g_2 (we initially keep $|g_1a|$ fixed, i.e., moving a on the circular arc, but if a gets onto the boundary of A, we move it along the boundary during the rotation); ba rotates around b at the same time. There is no other edge that the rotated g_1a and ba could intersect, for otherwise there is a triangle or a degree-3 disk (Fig. 7, middle).
- Otherwise, either there is a degree-3 disk among G_1, G_2, A, B (a contradiction), or x is in one of G_1, G_2, meaning that there is an edge between a green vertex and one of a, b (say the edge is g_1a). In this case, we move g_1 along g_1g_2 to x to resolve the intersection (Fig. 7, right).

Finally, suppose there is an intersection of a bichromatic edge gr with an edge ab. We may assume that ab is a red edge r_1r_2; since the bichromatic case is more restrictive, meaning that the location of a green vertex is at the center of its disk, while a red vertex may be anywhere within its disk. Let G, R, R_1, R_2 be the corresponding disks, and let $x = gr \cap r_1r_2$ be the intersection point.

We claim that at most one of gr_1, r_1r, rr_2, and gr_2 exist. If two exist that share a vertex (e.g., gr_1 and r_1r), then they form a triangle with one of gr or r_1r_2 (contradiction). If two exist that do not share a vertex (e.g., gr_1 and rr_2), then they form a cycle with gr and r_1r_2, contradicting the no-short-cycle property.

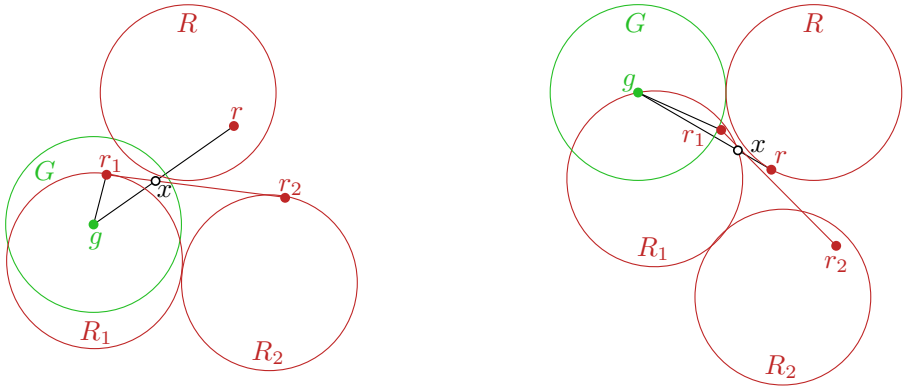

Figure 8 Edge gr_1 is in DIG. Left: $x \in G$. The new location of g is the hollow circle. Right: $x \notin G \cup R_1$. The new location for r_1 is the hollow circle.

Suppose G intersects R_1 or R_2 (say wlog R_1) so gr_1 is an edge in the DIG. If $x \in G$, we can move g to x and eliminate the crossing (Fig. 8, left). This doesn't introduce any new crossings since the new edge segments are sub-segments of the old ones. (Note: Vertex g has one edge to r_1 and one to r; the first is replaced by xr_1 and the second by xr, both of which

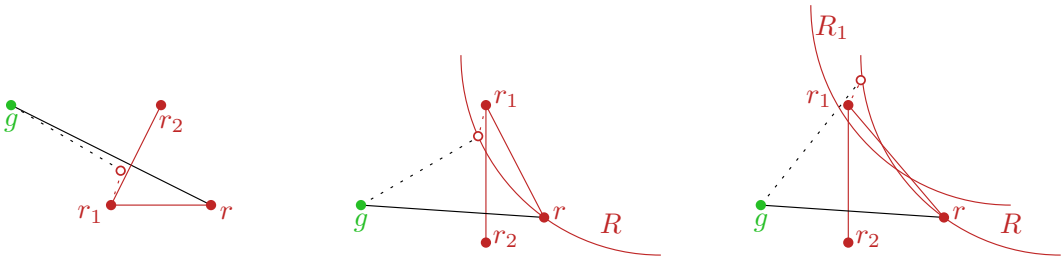

Figure 9 The fixes are dotted, the new locations for r are the hollow circles. Left: $x \in R$. Middle and right: $x \notin R$ (r is shown already pulled onto ∂R).

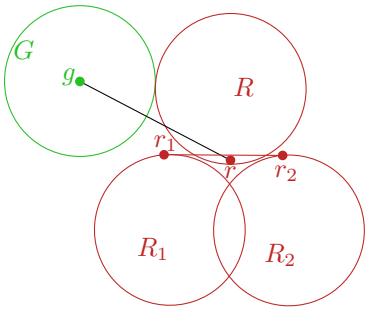

Figure 10 $r \in R$ is inside the hippodrome $H(R_1, R_2)$, which violates path isolation.

are sub-segments of the original drawing. Vertex g has no further adjacent edges.) Similarly, if $x \in R_1$, we can move r_1 to x to eliminate the crossing. If neither of these is true, we move r_1 to the intersection of gr with the boundary of R_1 (Fig. 8, right).

Otherwise, suppose R intersects R_1 or R_2 (say wlog R_1) so rr_1 is an edge. As before, we can eliminate the intersection if $x \in R$ or $x \in R_1$ (Fig. 9, left). If neither of these is true, we move r back towards x along gr as much as possible, i.e., onto the boundary of R. We then rotate gr around g, moving r along the boundary of R (again, no other edges can be in the vicinity of the moved edges since that would imply a vertex with degree at least 3): if the boundary of R intersects r_1r_2, then r is moved until it goes just over the intersection (Fig. 9, middle); otherwise, r is moved until gr jumps over r_1 (Fig. 9, right).

Finally, suppose none of the edges gr_1, r_1r, rr_2, and gr_2 exist. The intersection x must happen in a delta of R_1, R_2, with the segment r_1r_2 "cutting off" a cap of R in which x lies (Fig. 10). This implies that disk R intersects the hippodrome $H(R_1, R_2)$, which violates the path-isolation property.

5 Discussion

We showed that in general recognizing DOGs is hard, but recognizing thin unit DOGs is easy. Our hardness proof in Section 3 uses disks of different radii, and a natural question is how hard it is to recognize a unit DOG (i.e., the complexity of deciding planar disk-obedience for unit DIGs). We were not able to settle the complexity of the problem, and comment on it here.

Some parts of our hardness construction can be done with unit disks. For instance, the variable chains can be made to work with unit disks by carefully placing them (see our game instance at `http://tube.geogebra.org/m/BzVYK21A`). Similarly, it is possible to create a splitter gadget with unit disks. However, the clause gadget crucially relies on the abilities to

build "walls" of small disks, and while it is possible to grow these small disks significantly, it appears not to be possible to make them the same size as the other disks. In principle, we could potentially try to reduce from a different NP-hard version of SAT (not the standard 3SAT) to deciding disk-obedience for UDGs – Schaefer's dichotomy theorem [29, 9] tells which versions are hard; however, were not able to find a hard version whose `true` (and only `true`) clauses could be encoded by DOGs.

As far as our positive result goes, we believe that our problem is fixed-parameter tractable with the running time of the FPT algorithm depending on the ratio ρ of largest to smallest radius of the disks. Indeed, the fact that it is enough to look only at a certain number k of red vertices around each triangle is based on the packing argument (the disks that can intersect a triangle edge must be close to the triangle, and if there are too many such disks, they will define a point of high depth), which works for DIGs with arbitrary-radius disks (k will be a function of ρ, of course). Similarly, the arguments about non-intersection of the colorful edges (or about fixing the intersections) can be potentially adapted to apply to different-size disks, thus possibly extending the whole algorithm to general DOGs.

Our problem can be naturally extended to shapes other than disks, defining the intersection graph; for any class of shapes it may be asked whether a given -IG is a -OGs. For instance, rectangle intersection graphs (RIGs) have also been studied earlier [10, 16] but the question "Is the RIG a ROG?" has yet to be answered (even in its simplest form "Is the SIG a SOG?" when the rectangles are squares). It could be interesting to see for which shapes the -IG/-OG problem is polynomially solvable and also for which shapes the "shape-obedience game" is fun to play.

Acknowledgements. We thank the anonymous reviewers for their helpful comments. We also thank Boris Klemz for noticing that our thin DIGs were not quite thin enough in an earlier draft. Research on the topic of this paper was initiated at the 2nd International Workshop on Drawing Algorithms for Networks in Changing Environments (DANCE 2015) in Langbroek, The Netherlands, supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.023.208. W. Evans is supported by NSERC. M. van Garderen is supported by the European Union's 7th Framework Programme for research, technological development and demonstration under grant agreement n° 1133 (Project CARIB) and ERC grant agreement n° 319209 (Project NEXUS1492). The project CARIB is financially supported by the HERA JRP (www.heranet.info) which is co-funded by AHRC, AKA, BMBF via PT-DLR, DASTI, ETAG, FCT, FNR, FNRS, FWF, FWO, HAZU, IRC, LMT, MHEST, NWO, NCN, RANNÍS, RCN, VR and The European Community FP7 2007-2013, under the Socio-economic Sciences and Humanities programme. M. Löffler is supported by the Netherlands Organisation for Scientific Research (NWO) under grant 639.021.123. V. Polishchuk is supported by grant 2014-03476 from the Sweden's innovation agency VINNOVA.

References

1 Patrizio Angelini, Giordano Da Lozzo, Marco Di Bartolomeo, Giuseppe Di Battista, Seok-Hee Hong, Maurizio Patrignani, and Vincenzo Roselli. Anchored drawings of planar graphs. In *Graph Drawing*, volume 8871 of *Lecture Notes in Computer Science*, pages 404–415. Springer, 2014.

2 Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, and Fabrizio Frati. Strip planarity testing. In Stephen K. Wismath and Alexander Wolff, editors, *Graph Drawing*, volume 8242 of *Lecture Notes in Computer Science*, pages 37–48. Springer, 2013.

3 Esther M Arkin, Michael A Bender, Erik D Demaine, Sándor P Fekete, Joseph SB Mitchell, and Saurabh Sethia. Optimal covering tours with turn costs. *SIAM Journal on Computing*, 35(3):531–566, 2005.

4 Esther M Arkin, Sándor P Fekete, Kamrul Islam, Henk Meijer, Joseph SB Mitchell, Yurai Núñez-Rodríguez, Valentin Polishchuk, David Rappaport, and Henry Xiao. Not being (super) thin or solid is hard: A study of grid Hamiltonicity. *Computational Geometry*, 42(6):582–605, 2009.

5 Lali Barrière, Pierre Fraigniaud, Lata Narayanan, and Jaroslav Opatrny. Robust position-based routing in wireless ad hoc networks with irregular transmission ranges. *WCMC*, pages 141–153, 2003. doi:10.1002/wcm.108.

6 Prosenjit Bose, Pat Morin, Ivan Stojmenović, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless networks*, 7(6):609–616, 2001.

7 Kevin Buchin, Irina Kostitsyna, Maarten Löffler, and Rodrigo I Silveira. Region-based approximation algorithms for visibility between imprecise locations. In *ALENEX*, pages 94–103. SIAM, 2015.

8 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.

9 Hubie Chen. A rendezvous of logic, complexity, and algebra. *ACM Comput. Surv.*, pages 2:1–2:32, December 2009. doi:10.1145/1592451.1592453.

10 M.B. Cozzens. *Higher and Multi-dimensional Analogues of Interval Graphs*. PhD thesis, Rutgers University, 1981.

11 Hongwei Du, Xiaohua Jia, Deying Li, and Weili Wu. Coloring of double disk graphs. *J. Global Opt*, pages 115–119, 2004. doi:10.1023/B:JOGO.0000006750.85332.0f.

12 Alon Efrat, Sándor P Fekete, Joseph SB Mitchell, Valentin Polishchuk, and Jukka Suomela. Improved approximation algorithms for relay placement. *ACM Transactions on Algorithms*, 12(2):20, 2015.

13 M. Gromov. Hyperbolic groups. In S. M. Gersten, editor, *Essays in Group Theory*, pages 75–263. Springer New York, 1987.

14 Marja Hassinen, Joel Kaasinen, Evangelos Kranakis, Valentin Polishchuk, Jukka Suomela, and Andreas Wiese. Analysing local algorithms in location-aware quasi-unit-disk graphs. *Discr Appl Math*, pages 1566–1580, 2011. doi:10.1016/j.dam.2011.05.004.

15 Jean-Claude Hausmann. On the Vietoris-Rips complexes and a cohomology theory for metric spaces. In F. Quinn, editor, *Annals of Mathematics Studies*, volume 138, pages 175–188, Princeton, N.J., 1995. Princeton University Press. Prospects in topology : proceedings of a conference in honor of William Browder.

16 Hiroshi Imai and Takao Asano. Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane. *Journal of Algorithms*, 4(4):310 – 323, 1983. doi:http://dx.doi.org/10.1016/0196-6774(83)90012-3.

17 Balázs Keszegh, János Pach, and Domotor Palvolgyi. Drawing planar graphs of bounded degree with few slopes. *SIAM Journal on Discrete Mathematics*, 27(2):1171–1183, 2013.

18 Alexander Kröller, Sándor P Fekete, Dennis Pfisterer, and Stefan Fischer. Deterministic boundary recognition and topology extraction for large sensor networks. In *SoDA*, pages 1000–1009, 2006.

19 Sven O Krumke, Madhav V Marathe, and SS Ravi. Models and approximation algorithms for channel assignment in radio networks. *Wireless networks*, 7(6):575–584, 2001.

20 Fabian Kuhn, Thomas Moscibroda, and Rogert Wattenhofer. Unit disk graph approximation. In *Proceedings of the 2004 joint workshop on Foundations of mobile computing*, pages 17–23. ACM, 2004.

21 Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Ad hoc networks beyond unit disk graphs. *Wireless Networks*, 14(5):715–729, 2007. doi:10.1007/s11276-007-0045-6.

22 Fabian Kuhn, Rogert Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing: of theory and practice. In *PoDC*, pages 63–72, 2003.

23 David Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982. doi:10.1137/0211025.

24 Maarten Löffler. Existence and computation of tours through imprecise points. *IJCGA*, pages 1–24, 2011. doi:10.1142/S0218195911003524.

25 Maarten Löffler and Wolfgang Mulzer. Unions of onions: Preprocessing imprecise points for fast onion decomposition. *JoCG*, 5(1):1–13, 2014.

26 Maarten Löffler and Marc van Kreveld. Largest and smallest convex hulls for imprecise points. *Algorithmica*, 56(2):235–269, 2010.

27 John Nolan. Bisectored unit disk graphs. *Networks*, 43(3):141–152, 2004. doi:10.1002/net.10111.

28 Maurizio Patrignani. On extending a partial straight-line drawing. *Int. J. Found. CS*, pages 1061–1070, 2006. doi:10.1142/S0129054106004261.

29 Thomas J. Schaefer. The complexity of satisfiability problems. In *SToC'78*, STOC '78, pages 216–226. ACM, 1978. doi:10.1145/800133.804350.

30 L. Vietoris. Über den höheren zusammenhang kompakter räume und eine klasse von zusammenhangstreuen abbildungen. *Mathematische Annalen*, 97(1):454–472, 1927. doi:10.1007/BF01447877.

31 Yue Wang, Jie Gao, and Joseph S B Mitchell. Boundary recognition in sensor networks by topological methods. In *12th annual conference on Mobile computing and networking*, pages 122–133, 2006.