On the recognition of unit disk graphs and the Distance Geometry Problem with Ranges

Guilherme Dias da Fonseca[a], Vinícius Gusmão Pereira de Sá[a], Raphael Carlos Santos Machado[b], Celina Miraglia Herrera de Figueiredo[a]

[a] *Universidade Federal do Rio de Janeiro — Brazil*
[b] *Instituto Nacional de Metrologia, Qualidade e Tecnologia — Brazil*

Abstract

We introduce a method to decide whether a graph G admits a realization on the plane in which two vertices lie within unitary distance from one another exactly if they are neighbors in G. Such graphs are called unit disk graphs, and their recognition is a known **NP**-hard problem. By iteratively discretizing the plane, we build a sequence of geometrically defined trigraphs—graphs with mandatory, forbidden and optional adjacencies—until either we find a realization of G or the interruption of such a sequence certifies that no realization exists. Additionally, we consider the proposed method in the scope of the more general Distance Geometry Problem with Ranges, where arbitrary intervals of pairwise distances are allowed.

Keywords: unit disk graph, distance geometry, discretization, trigraph

1. Introduction

The fundamental *Distance Geometry Problem* (DGP) consists in determining a set of points whose pairwise distances are known *a priori*. More formally, the input comprises a graph where the weight of each edge indicates the exact intended distance between its incident vertices in a realization of the graph in some metric space.

Email addresses: `fonseca@cos.ufrj.br` (Guilherme Dias da Fonseca), `vigusmao@dcc.ufrj.br` (Vinícius Gusmão Pereira de Sá), `rcmachado@inmetro.gov.br` (Raphael Carlos Santos Machado), `celina@cos.ufrj.br` (Celina Miraglia Herrera de Figueiredo)

Partially supported by CNPq and FAPERJ.

The generality of the DGP embraces a variety of applications, notably in wireless sensor networks, pattern recognition, computational biology and astronomy, to cite but a few. A survey on the theory of Euclidean Distance Geometry and its most important applications, with special emphasis to molecular conformation problems, can be found in [7]. Recently, researchers of multidisciplinary fields were brought together in a workshop dedicated to the theme [1], where a preliminary version of this paper was presented.

We consider a generalization of the DGP where the exact distances between vertices are replaced by intervals. We call it the *Distance Geometry Problem with Ranges* (DGPR). The applicability of such a generalized model is clear. Distance ranges may be due to imprecisions—for example, in measuring intermolecular distances—or to natural relaxations of exact distances into intervals of allowed distances—for instance, in the deployment of a wireless telecommunication infrastructure, where some antennas should be neither too close (to avoid interference) nor too far from one another (to allow for a control communication channel between them).

First and foremost, we focus on the case where each range is either $[0, 1]$ or $(1, \infty)$. Given such restriction, the problem corresponds exactly to a classic **NP**-hard problem in Graph Theory, namely the recognition of unit disk graphs. The technique we introduce, based on an iterative discretization of the continuous solution space along with a tripartition of the edges of an auxiliary graph, is nevertheless general enough to suit the (unrestricted) DGPR with only minor modifications.

This paper is organized as follows. In Section 2, we give a brief account on unit disk graphs, discussing the importance and the difficulty of recognizing them. In Section 3, we introduce the concept of trigraph realizations, which is central in our discretization technique. In Section 4, we propose a method for unit disk graph recognition which has proved to work well for graphs with a small number of vertices. In Section 5, we present some experimental results, including the unprecedented computational recognition of several graphs. In Section 6, we consider the prospects of leveraging our technique to the more general scope of the DGPR, posing a series of open theoretical questions. In Section 7, we make our concluding remarks.

Throughout the text, we denote the vertex set and the edge set of a graph G respectively as $V(G)$ and $E(G)$, as usual. The set of neighbors of v is denoted $N(v)$, with $d(v) = |N(v)|$. An edge incident to vertices v and w is written vw, or, if v or w are numbers, $\{v, w\}$. All norms $\|\cdot\|$ are Euclidean.

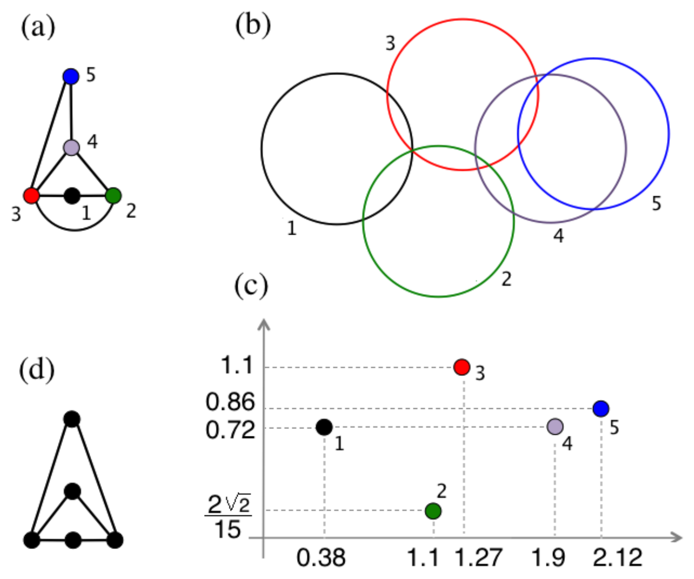

Figure 1: (a) A unit disk graph G; (b) a model of congruent disks for G; (c) a formal UDG realization of G; (d) the graph $K_{2,3}$, which is not a UDG.

2. Unit disk graphs

A *unit disk graph* (UDG) is a graph whose vertices can be mapped to points on the plane and whose edges are defined by pairs of points within unitary Euclidean distance from one another. Unit disk graphs can also be regarded as intersection graphs of coplanar congruent disks, and they have been widely studied in recent times owing to their relevance in wireless sensor networks [8].

Definition 1. *Given a graph G, a* UDG realization *of G is a function $\phi\colon V(G) \to \mathbb{R}^2$ such that, for some real number $d > 0$ and all $u, v \in V(G)$,*

- $\|\phi(u) - \phi(v)\| \leq d$, *if $uv \in E(G)$; and*
- $\|\phi(u) - \phi(v)\| > d$, *if $uv \notin E(G)$.*

A graph is a UDG if and only if it admits a UDG realization. For convenience, the value of d is often taken to be 1, hence the name "unit" disk graph. In this paper, we consider it to be so, unless marked otherwise. The graph in Figure 1(a) is a UDG, whereas the graph in Figure 1(d) is not.

The problem of recognizing whether a given graph is a UDG is **NP**-hard [2], and the fastest known recognition algorithm is doubly exponential [10]. It is not known for the time being whether the problem belongs to **NP**, since the size of the natural certificate comprising the coordinates of the vertices (i.e., a UDG realization) is not polynomially bounded under the classic model of computation over finite strings [9], and no other polynomial certificates are known.

Whereas many applications would benefit from the ability to obtain feasible UDG realizations, being able to prove that some graphs are *not* UDG would also have immediate consequences. For example, in [11], Wu et al. conjecture that the graph in Figure 2(a) is not a UDG. The correctness of such conjecture would imply a decrease from 3.8 to 3.6 in the maximum asymptotic ratio between the size of a maximal independent set[2] and the size of a minimum connected dominating set[3] in any given unit disk graph, thus affecting the approximation factor of algorithms that estimate the latter number by computing the former. Another example was obtained in [3]. The graph in Figure 2(b) is a UDG (a geometric realization was provided in [4]) and corresponds to the worst known instance for an algorithm that approximates the minimum dominating set of a unit disk graph. Such a graph establishes a lower bound of 4.8 for that algorithm's approximation factor, while an upper bound of $44/9 > 4.8$ is the best that has been proved. It is not known, however, whether the graph in Figure 2(c) is a UDG. Being able to decide it either way would tighten one of those bounds, reducing the current gap between them.

To date, all graphs which have successfully been classified with respect to the family of UDG have been done so by trial and error or by ad-hoc geometric methods. In face of the lack of efficient algorithms to solve the UDG recognition problem, a search-oriented computational solution comes in handy, at least for instances of modest size. However, an obvious hindrance to approaching the problem by exhaustive search is the fact that the set of functions from $V(G)$ onto $\mathbb{R}^2$ is uncountable. In the next sections, we try to overcome this fact by judiciously discretizing the solution space.

[2]A set $S \subseteq V(G)$ is an *independent set* if no two vertices of S are adjacent in graph G.

[3]A set $D \subseteq V(G)$ is a (connected) dominating set if (D is connected and), for all $w \in V(G) \setminus D$, there is a vertex $v \in D$ such that $vw \in E(G)$.

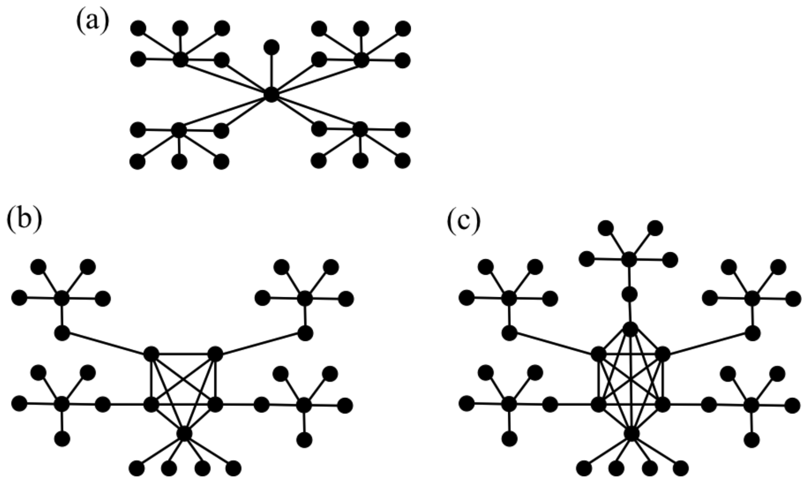

Figure 2: (a) Graph conjectured in [11] not to be a UDG; (b) unit disk graph that sets the lower bound for the approximation factor of the algorithm introduced in [3]; (c) is this graph a UDG?

3. Trigraph realizations

We are about to introduce a novel technique for the recognition of unit disk graphs. A concept that is central to our approach is that of *trigraph realizations*, which, loosely speaking, are placements of vertices on the plane where each pairwise distance either belongs to the intended range or falls out of it by only a narrow margin. The rationale is that if a certain placement of vertices is not even one such trigraph realization, then no small corrective perturbation could turn them into a UDG realization, and hence a whole chunk of the (continuous) solution space can be immediately discarded.

For some positive $\epsilon \in \mathbb{Q}$, consider the set $Q_\epsilon = \{x \in \mathbb{Q}\colon x = d\epsilon,\ d \in \mathbb{Z}\}$, and let $C_\epsilon = Q_\epsilon \times Q_\epsilon$ be a discrete set of 2-dimensional coordinates. We call C_ϵ an ϵ-*mesh*. The smaller the value of ϵ, the greater the *granularity* of the mesh. We say C_{ϵ_1} is *thinner* than C_{ϵ_2} (equivalently, C_{ϵ_2} is *thicker* than C_{ϵ_1}) if the granularity of C_{ϵ_1} is greater than that of C_{ϵ_2}, i.e. if $\epsilon_1 < \epsilon_2$. A mesh can be regarded as the set of pairwise intersection points of equally spaced vertical and horizontal lines. Sometimes it is also convenient to refer to a *cell* of a mesh, denoting the points of the plane that lie inside the square defined by two consecutive horizontal lines and two consecutive vertical lines. Any graph realization whose image is a subset of C_ϵ is said to be ϵ-*discrete*.

Definition 2. *Let G be a graph and let C_ϵ be a mesh. An* ϵ-discrete trigraph realization *(or simply trigraph realization, if ϵ is clear in the context or can be disregarded) of G is a function $\psi_\epsilon\colon V(G) \to C_\epsilon$ such that, for all $u, v \in V(G)$,*

- $\|\psi_\epsilon(u) - \psi_\epsilon(v)\| < 1 + \epsilon\sqrt{2}$, *if $uv \in E(G)$;*
- $\|\psi_\epsilon(u) - \psi_\epsilon(v)\| > 1 - \epsilon\sqrt{2}$, *if $uv \notin E(G)$.*

Notice that a trigraph realization of G is not necessarily a UDG realization of G. On the other hand, every (discrete) UDG realization of G is a trigraph realization of G. Given a mesh C_ϵ and a trigraph realization ψ_ϵ of G, we define the trigraph $H[\psi_\epsilon]$ as the complete graph whose vertex set is $V(G)$ and where each edge uv in $E(H[\psi_\epsilon])$ denotes one of three possible adjacency types:

(i) a *mandatory* adjacency, if $\|\psi_\epsilon(u) - \psi_\epsilon(v)\| \leq 1 - \epsilon\sqrt{2}$;

(ii) a *forbidden* adjacency, if $\|\psi_\epsilon(u) - \psi_\epsilon(v)\| \geq 1 + \epsilon\sqrt{2}$; or

(iii) an *optional* adjacency, if $1 - \epsilon\sqrt{2} < \|\psi_\epsilon(u) - \psi_\epsilon(v)\| < 1 + \epsilon\sqrt{2}$.

A trigraph realization ψ_ϵ of a graph G can therefore be regarded as a mapping of $V(G)$ onto points of an ϵ-mesh such that the trigraph $H[\psi_\epsilon]$, defined as discussed, satisfies the following conditions for all vertices $u, v \in V(G)$:

- if $uv \in E(G)$, then the corresponding adjacency type, in $H[\psi_\epsilon]$, is either mandatory or optional; and
- if $uv \notin E(G)$, then the corresponding adjacency type, in $H[\psi_\epsilon]$, is either forbidden or optional.

A nice way of visualizing the trigraph $H[\psi_\epsilon]$ is through a *trigraph disk model* similar to the disk model given in Figure 1(b). The single, yet crucial difference is that now each disk actually comprises two concentric circles: an inner, closed circle of diameter $1-\epsilon\sqrt{2}$, and an outer, open circle of diameter $1 + \epsilon\sqrt{2}$. The circular crown between the two circles can be regarded as an area of uncertainty (a "gray area"). Whenever two inner circles intersect, the adjacency between the corresponding vertices is mandatory; whenever two disks do not intersect at all, the adjacency between those vertices is forbidden; and, finally, whenever two gray areas intersect, but the inner circles do not intersect, the adjacency between those vertices is optional. Figure 3 illustrates the concept.

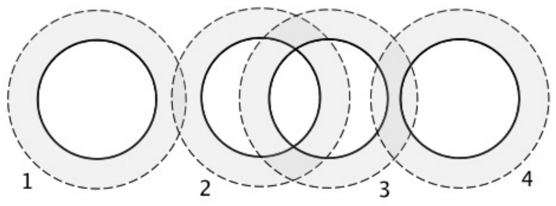

Figure 3: A trigraph disk model, where the only mandatory adjacency is $\{2, 3\}$; the forbidden adjacencies are $\{1, 3\}, \{1, 4\}$ and $\{2, 4\}$; and the optional adjacencies are $\{1, 2\}$ and $\{3, 4\}$.

Definition 3. *Given an ϵ-mesh, for some positive rational ϵ, the* discretization function *$f_\epsilon \colon \mathbb{R}^2 \to C_\epsilon$ is defined as*

$$f_\epsilon(x, y) = \left(\left\lfloor \frac{x}{\epsilon} \right\rfloor \epsilon, \left\lfloor \frac{y}{\epsilon} \right\rfloor \epsilon \right).$$

In other words, $f_\epsilon(x, y)$ returns the bottom-left corner of the cell of C_ϵ containing the point (x, y).

The following two lemmas provide the stopping criteria for our method. The discretization function defined above will be used in the proof of the second lemma and throughout the whole text.

Lemma 1. *Let G be a graph, and let ψ_ϵ be a trigraph realization of G. If all optional adjacencies in $H[\psi_\epsilon]$ correspond to pairs of neighbors in G (alternatively, to pairs of non-neighbors in G), then G is a UDG.*

Proof — If there are no optional adjacencies in $H[\psi_\epsilon]$, then ψ_ϵ is clearly a UDG realization of G. Otherwise, suppose each optional adjacency vw in $H[\psi_\epsilon]$ is such that $vw \in E(G)$. In this case, a UDG realization of G can immediately be obtained by scaling all coordinates in the image of ψ_ϵ through a division by the maximum distance in ψ_ϵ describing an optional adjacency. If, on the other hand, each optional adjacency vw in $H[\psi_\epsilon]$ is such that $vw \notin E(G)$, then, by dividing all coordinates by the maximum distance in ψ_ϵ describing a mandatory adjacency, the ensuing mapping is a UDG realization of G. ■

Lemma 2. *If G is a UDG, then G admits a trigraph realization ψ_ϵ for all positive $\epsilon \in \mathbb{Q}$.*

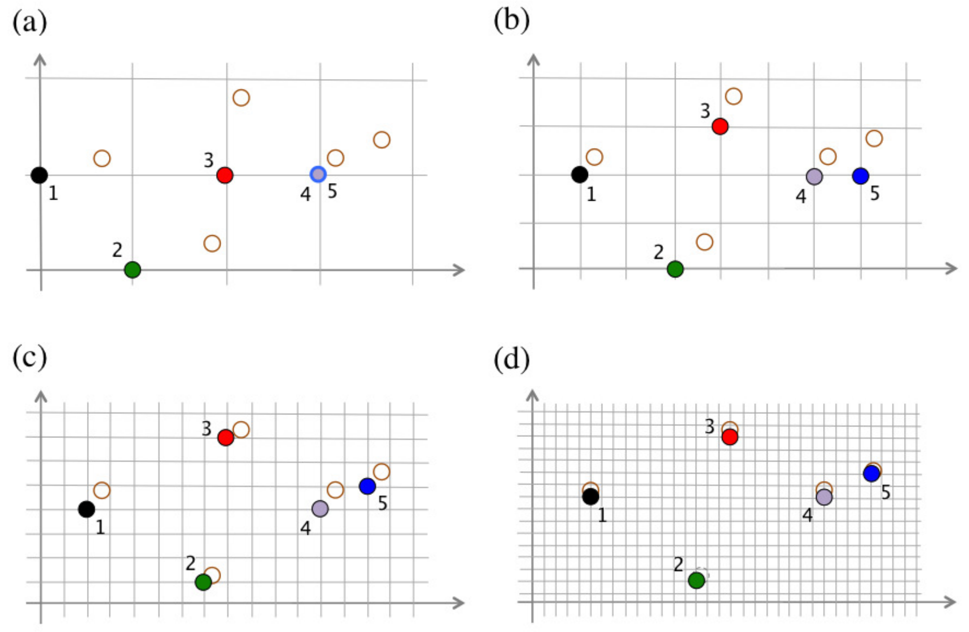

Figure 4: (a–d) Trigraph realizations for the graph in Figure 1(a). Each trigraph realization was obtained by submitting the UDG realization given in Figure 1(c) to the discretization function f_ϵ (see Definition 3), for $\epsilon = 0.6, 0.3, 0.15$ and 0.075, respectively. In (a), vertices 4 and 5 are placed at coincident points. Hollow circles indicate the placement of the vertices before the discretization.

Proof — Let G be a UDG, and let ϕ be a UDG realization of G. For some positive $\epsilon \in \mathbb{Q}$, let C_ϵ be an ϵ-mesh. Let $\psi_\epsilon = \phi \circ f_\epsilon$ be the mapping from $V(G)$ onto C_ϵ that we obtain by submitting each point in the image of ϕ to the discretization function f_ϵ (see Figure 4). We show that ψ_ϵ is a trigraph realization of G. Let u, v be two vertices of G, and let $r = \|\phi(u) - \phi(v)\|$ and $r' = \|\psi_\epsilon(u) - \psi_\epsilon(v)\|$. The absolute difference $|r' - r|$ is less than $\epsilon\sqrt{2}$, since both points are discretized towards the bottom-left corners of their containing cells, for a maximum relative displacement that is strictly less than the diagonal of a cell. As a consequence, if $uv \in E(G)$ then $r \leq 1$ and $r' < 1 + \epsilon\sqrt{2}$; and if $uv \notin E(G)$ then $r > 1$ and $r' > 1 - \epsilon\sqrt{2}$. The function ψ_ϵ is therefore a trigraph realization of G, which concludes the proof. ■

We remark that, for $\epsilon > 1/\sqrt{2}$, the ϵ-mesh C_ϵ is so thick that, for any graph G, even a trivial function such as $\psi_\epsilon(v) = (0, 0)$ for all $v \in V(G)$

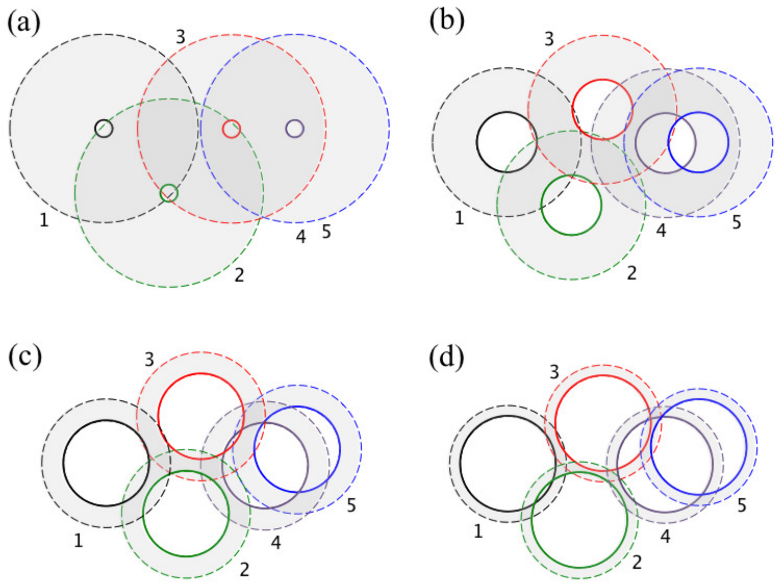

Figure 5: (a–d) Trigraph disk models of increasing granularity, respective to the trigraph realizations given in Figure 4.

would be a perfectly valid trigraph realization of G. Indeed, all pairwise distances in this case would be zero, hence all adjacencies in $H[\psi_\epsilon]$ would be optional, because zero belongs to $(1 - \epsilon\sqrt{2}, 1 + \epsilon\sqrt{2})$ when $\epsilon > 1/\sqrt{2}$. If one thinks about a trigraph disk model for $H[\psi_\epsilon]$, the inner circles would all have diameter zero, and the whole disks would consist of "gray" (optional adjacency) areas. The existence of trivial trigraph realizations in meshes that are too thick renders such meshes useless from the standpoint of the trigraph-based discretization method explained in Section 4. Intuitively, the thinner the mesh, the closer to a UDG realization a trigraph realization is expected to be (see Figure 5).

We conclude this section with a third lemma about UDG realizations. The corollary that follows plays a role in the performance of our method.

Lemma 3. *If G is a UDG, then there is a UDG realization of G in which no distance between two vertices is exactly 1 (that is, no two disks are tangent on the associated disk model), and no distance between two vertices is zero (that is, no two disks are coincident on the associated disk model).*

Proof — Let ϕ be a realization of G with vertices at distance exactly 1 from one another, and let $r = 1 + \gamma$ be the smallest distance in ϕ between two non-neighbors in G. Now, for all $v \in V(G)$, we define $\phi'(v) = \phi(v)/(1 + \gamma')$, for some positive $\gamma' < \gamma$. The ensuing mapping $\phi' \colon V(G) \to \mathbb{R}^2$ is clearly a UDG realization of G with no two vertices at distance exactly 1 from one another, since the distance in ϕ' between two neighbors in G is at most $1/(1 + \gamma') < 1$, and the distance in ϕ' between two non-neighbors in G is at least $(1 + \gamma)/(1 + \gamma') > 1$.

Now let ϕ be a realization of G with no vertices at distance exactly 1 from one another, and let γ be the minimum $|1 - \|\phi(u) - \phi(v)\||$ over all $u, v \in V(G)$. Suppose there are two coincident vertices $w, z \in V(G)$, that is, $\phi(w) = \phi(z)$. We define $\phi' \colon V(G) \to \mathbb{R}^2$ as a mapping that is identical to ϕ except for the image of w, which we define as $\phi'(w) = \phi(w) + (\gamma', 0)$, for some positive $\gamma' < \gamma$. In other words, we perturb slightly the position of one of the former coincident vertices, in such a way that the relative displacement between w and every other vertex of G is strictly less than γ. Now $\phi'(w)$ and $\phi'(z)$ are not coincident, and clearly ϕ' is a UDG realization of G, since the distance in ϕ' between two neighbors in G is at most $1 - \gamma + \gamma' < 1$, and the distance in ϕ' between two non-neighbors in G is at least $1 + \gamma - \gamma' > 1$. By repeating the argument with all pairs of originally coincident vertices, we obtain a realization of G with no two vertices at distance 1 from each other and no coincident vertices. ■

Corollary 1. *For any positive rational ϵ and any two vertices u and v of a unit disk graph G, there is an ϵ-discrete trigraph realization ψ_ϵ of G such that $\psi_\epsilon(u) \neq \psi_\epsilon(v)$.*

Proof — By Lemma 3, if G is a UDG then G admits a realization ϕ such that $\phi(u) \neq \phi(v)$. We can now follow the same reasoning as in the proof of Lemma 2, if we just translate all points in the image of ϕ—or, equivalently, we translate the mesh underneath them—until $\phi(u)$ and $\phi(v)$ belong to distinct cells, and only then we apply the discretization function. ■

4. The algorithm

We now formulate a computational method for the recognition of unit disk graphs. Its basic operation is a search for discrete realizations of the input graph G on a certain ϵ-mesh, until either a (discrete) UDG realization is

Algorithm 1: *UDG_recognition*

```
input: a connected graph $G$, an initial granularity $initial\_epsilon$
ouput: YES, if $G$ is a unit disk graph;
       NO, otherwise
1. $\epsilon \leftarrow initial\_epsilon$
2. repeat
3.     $result \leftarrow has\_discrete\_realization(G, \epsilon)$
4.     if $result$ = YES then return YES
5.     if $result$ = NO or $\epsilon = \epsilon_{min}$ then return NO
6.     $\epsilon \leftarrow \max\{refine\_granularity(\epsilon), \epsilon_{min}\}$
```

Procedure 2: *has_discrete_realization*

```
input: a connected graph $G$, a rational $\epsilon > 0$
ouput: YES, if $G$ admits an $\epsilon$-discrete UDG realization;
       NO, if $G$ admits no $\epsilon$-discrete trigraph realizations;
       TRIGRAPH_ONLY, otherwise
1. $\psi \leftarrow$ an array with $|V(G)|$ null's      /* current coordinates */
2. $\pi \leftarrow$ a permutation of $V(G)$ corresponding to the visitation order
       of a breadth-first-search on $G$
3. return $place\_next\_vertex(G, \epsilon, \pi, 1, \psi)$
```

found or no trigraph realizations are available for G on the considered mesh. The computational effort of conducting an exhaustive search on a mesh that is too thin is, however, obviously high. We therefore start from a moderately thick ϵ-mesh (the initial value ϵ, passed as a parameter, is typically set slightly below $1/\sqrt{2}$, say 0.7), where the number of points is small, and we search through all possible ϵ-discrete trigraph realizations of G. We do so by positioning each vertex $v \in V(G)$, one by one, at a point chosen from a set of *candidate locations* for v on the ϵ-mesh. Whenever a vertex has no candidate locations, we backtrack and attempt different placements of the previous vertices. If a trigraph realization satisfying the conditions of Lemma 1 is ever found, then the algorithm returns YES (G is a UDG). If, on the other

```
Procedure 3: place_next_vertex
input: a connected graph $G$, a rational $\epsilon > 0$,
       a permutation $\pi$ of $V(G)$,
       the index $j$ in $\pi$ of the next vertex to be placed,
       an array $\psi$ with the coordinates of vertices already placed
ouput: YES, if $G$ admits an $\epsilon$-discrete UDG realization;
       NO, if $G$ admits no $\epsilon$-discrete trigraph realizations;
       TRIGRAPH_ONLY, otherwise
 1. $v \leftarrow \pi(j)$
 2. $P \leftarrow \{p \in C_\epsilon \colon \|p - \psi[\pi(k)]\| < 1 + \epsilon\sqrt{2}, \forall k < j, \pi(k) \in N(v)\} \setminus$
        $\{p \in C_\epsilon \colon \|p - \psi[\pi(k)]\| \leq 1 - \epsilon\sqrt{2}, \forall k < j, \pi(k) \notin N(v)\}$
 3. if $j = 1$ then $P \leftarrow \{(0, 0)\}$
 4. if $j = 2$ then $P \leftarrow P \cap \{p \in C_\epsilon \colon\ ordinate(p) = 0,\ abscissa(p) > 0\}$
 5. if $j = 3$ then $P \leftarrow P \cap \{p \in C_\epsilon \colon\ ordinate(p) \geq 0\}$
 6. found_trigraph ← False
 7. for all $p \in P$ do
 8.     $\psi[v] \leftarrow p$
 9.     if $j = |V(G)|$ then
10.         found_trigraph ← True
11.         if is_UDG_realization$(G, \epsilon, \psi)$ then return YES
12.     result ← place_next_vertex$(G, \epsilon, \pi, j + 1, \psi)$
13.     if result = YES then return YES
14.     if result = TRIGRAPH_ONLY then found_trigraph ← True
15. if found_trigraph = False then return NO
16. return TRIGRAPH_ONLY
```

hand, no trigraph realization exists at the considered granularity, then, by Lemma 2, G is not a UDG, and the algorithm returns NO. Finally, if one or more ϵ-discrete trigraph realization do exist, but none satisfies Lemma 1, then the search starts over, iteratively, on a thinner mesh—whose granularity is given by some function *refine_granularity*. A simple such function divides the current granularity by 2, but it is also possible to make it so that the attempted granularities correspond to any other desired sequence, such as $0.7, 0.7/2, 0.7/3, 0.7/4$, etc. Note that, the thinner the mesh, the higher the probability of obtaining a conclusive answer, yet the computational effort is also substantially higher.

Algorithm 1 depicts the basics of the proposed method. The subroutines *get_trigraph_realizations* and *place_next_vertex* are given separately as Procedures 2 and 3. The subroutine *is_UDG_realization* simply checks whether the given trigraph realization satisfies the conditions of Lemma 1. The pseudocodes for *is_UDG_realization* and *refine_granularity* are omited.

McDiarmid and Müller have shown [9, Lemma 7.4] that, if G is a UDG, then G admits a realization where the diameter d of the disks (as per Definition 1), as well as the coordinates of the disk centers, are integers of absolute value at most $2^{2^{\Theta(n)}}$, with $n = |V(G)|$. That is to say that G admits a realization where all coordinates are rationals of the form k/d, for $k, d \in \mathbb{Z}$ and both k and d are at most $2^{2^{\Theta(n)}}$. In other words, G admits an ϵ_{min}-discrete realization for $\epsilon_{min} = 1/d = 1/(2^{2^{\Theta(n)}})$. Thus, in a worst-case scenario, the number of iterations performed by our method is $\mathcal{O}(\log 2^{2^{\Theta(n)}}) = \mathcal{O}(2^{\Theta(n)})$. While this may be interesting from a theoretical standpoint, assuring that such a discretize-and-search strategy is destined to work in finite time (as opposed to a possibly endless, inconclusive search in the uncountable space $V(G) \times \mathbb{R}^2$), it is clear that graphs with many vertices may demand exorbitant running time—and that is what motivates a series of pruning-intended enhancements over our core idea.

4.1. Pruning

In most backtracking-based approaches, enormous gain in performance can be noticed if the search trees are pruned as much—and as early—as possible. On that basis, we have devised a number of performance-enhancing improvements on our method's basic ideas.

Refining thicker trigraph realizations into thinner ones. The knowledge acquired during the exploration of thicker meshes can be used to speed up the work on thinner meshes. Note that, if a graph G admits a UDG realization ϕ, then the application of the discretization function f_ϵ yields an ϵ-discrete trigraph realization of G, as argued in the proof of Lemma 2. But then again the discretization function $f_{2\epsilon}$ applied to that very same ϕ would yield a 2ϵ-discrete trigraph realization for G. As a corollary, if a certain placement $\psi_{2\epsilon}$ of vertices on the 2ϵ-mesh is *not* a trigraph realization of G, then no ϵ-discrete trigraph realization of G can exist with vertices placed at points of the ϵ-mesh whose images under $f_{2\epsilon}$ correspond to $\psi_{2\epsilon}$. This is so because clearly $f_{2\epsilon} \circ f_\epsilon(v) = f_{2\epsilon}(v)$ for all $v \in V(G)$. Therefore, we can restrict our search, on the ϵ-mesh, to refinements, so to speak, of each trigraph realization found at the 2ϵ granularity. In other words, we may only attempt a certain placement ψ_ϵ of vertices on the ϵ-mesh if there has been found a trigraph realization $\psi_{2\epsilon}$ such that $f_{2\epsilon}(\psi_\epsilon(v)) = \psi_{2\epsilon}(v)$ for all $v \in V(G)$. Fortunately, the points $\psi_\epsilon(v)$ that satisfy such condition can be easily spotted. They are the bottom-left corners of the four cells of the ϵ-mesh which lie inside the cell of the 2ϵ-mesh whose bottom-left corner is $\psi_{2\epsilon}(v)$. Though we have actually implemented this idea, the actual coding is a bit demanding, and we refrained from transcribing it into the pseudocode presented in this text for the sake of clarity.

Choosing an appropriate placement order. Tackling the most restricted decisions first usually pays off in backtracking-based approaches. Indeed, the order in which the vertices are placed on the mesh, in our method, may have considerable impact on its performance. Rather than considering the vertices in some arbitrary order, we follow the order obtained by a (breadth-first) search performed on the graph (Procedure 2, line 2). By doing so, each vertex v that is considered for placement—except for the very first one—is a neighbor of at least one vertex w already placed on the mesh, thus restricting the candidate points for v to those belonging to the (mandatory or optional) adjacency region surrounding w. As for the initial vertex, however, the fastest YES answers have been in most cases obtained when we started from the vertex with *minimum* degree. While this may be a bit counterintuitive, a possible explanation is that, when we tackle a "heavily restricted" region of the graph too early, we have plenty of room to make unfortunate choices that will only lead to a dead end later in the placement process, leading us to backtrack repeatedly until possibly all arrangements of the vertices in that subgraph have been attempted. By postponing the placement of the troublesome subgraph to a later stage, when plenty of other vertices have already been placed, the limited available room may allow us to backtrack in a cheaper fashion, after placing but a few vertices of that subgraph. On the other hand, if there is a subgraph which is not a UDG, then the algorithm would almost surely save precious time by attempting its placement earlier in the process.

Separating two vertices. In view of Corollary 1, our method may restrict its search to trigraph realizations in which u and v have distinct images, for whichever predefined $u, v \in V(G)$. For simplicity, we choose u and v to be the two first vertices in the placement order.

Disregarding translations, rotations and reflections. Owing to a possible translation of the intended mesh before the discretization function is applied, the first vertex in the placement order can be placed without loss of generality at a cell whose bottom-left corner is the origin $(0,0)$ of the mesh (Procedure 3, line 3). Furthermore, owing to a possible rotation of the mesh, the second vertex in the placement order can always be placed at a cell whose bottom-left corner has non-negative abscissa and ordinate zero. Actually, because of the separation of the two first vertices discussed in the previous paragraph, a strictly positive abscissa can be used instead (Procedure 3, line 4). Finally, owing to a possible reflection over the zero-ordinate axis, the third vertex can always be placed on or above that axis (Procedure 3, line 5). These measures reduce to a significant extent the number of candidate positions for the three first vertices.

Constraining the candidate locations. Every vertex that gets placed on the mesh has a direct impact on the candidate locations of all vertices not yet placed. Say $w \in V(G)$ has just been placed at point $\psi_\epsilon(w)$. Now, for every vertex v to the right of w in the placement order, if v is a neighbor of w in G then v must be placed sufficiently close to w (that is, within distance less than $1 + \epsilon\sqrt{2}$ from w), and if v is not a neighbor of w in G then v must be placed sufficiently far from w (that is, at a distance greater than $1 - \epsilon\sqrt{2}$ from w). Any points of the ϵ-mesh which do not satisfy such restrictions are not even considered as candidate locations (Procedure 3, line 2).

4.2. Further improvements

We now discuss two possible improvements to the overall performance of our method.

Keeping track of candidate positions dynamically. The determination of the set $P(v)$ of candidate locations for each vertex v is carried out as a set operation in line 2 of Procedure 3. It takes into consideration the constraints imposed by all vertices w (both neighbors and non-neighbors of v) to the left of v in the placement order. As a first possible enhancement, it may be worthwhile to keep track of each $P(w)$ throughout the whole execution of the algorithm rather than calculating them on-the-fly. After initializing $P(w)$, for all $w \in V(G)$, with all points of the mesh contained inside a bounding circle of radius $|V(G)|-1$ centered at the origin, the placement of each vertex $w \neq v$ on a point $\psi(w)$ of the ϵ-mesh triggers the following update of $P(v)$, for each v yet to be placed: if w is a neighbor of v in the input graph G, then all points whose distances to $\psi(w)$ are at least $1 + \epsilon\sqrt{2}$ (the forbidden adjacency region around $\psi(w)$) are removed from $P(v)$; otherwise, if w is not a neighbor of v in G, then all points whose distances to $\psi(w)$ are at most $1 - \epsilon\sqrt{2}$ (the mandatory adjacency region around $\psi(w)$) are removed from $P(v)$. By doing so, the next vertex to be placed can be chosen, for instance, among those with the fewest placement options (rather than following a predefined sequence), and—more importantly—the whole search subtree rooted at the current configuration can be immediately dropped whenever one of the vertices yet to be placed is left with an empty set of candidate locations.

Imposing predefined permutations of symmetric homogeneous sets. Suppose a graph G has k pendant vertices with a common neighbor v. Clearly, if G admits a trigraph realization ψ_ϵ for some $\epsilon > 0$, then G admits $k!$ quasi-identical trigraph realizations, one for each permutation of those pendant vertices along the points that correspond to their images under ψ_ϵ. As a consequence, our method may care about just one such permutation without loss of generality, disregarding all candidate placements which do not conform to it. More generally, if $H = \{u_1, \ldots, u_h\}$ is a homogeneous set[4] of G, and the subgraph of G induced by H is either an independent set or a clique[5], then it is possible to restrict the search to mappings ψ_ϵ of G that satisfy $\psi_\epsilon(u_1) \leq \ldots \leq \psi_\epsilon(u_h)$ according to some total order "$\leq$" of the points of the mesh. Our current implementation accepts that such sets H are informed by the user (and some of our computational results indeed benefited from this kind of input), but it does not detect them automatically—which would be a nice feature. It may also be possible to exploit other types of graph automorphisms to shave off computational effort that would otherwise be spent on redundant placement attempts.

We are currently investigating the two aforementioned algorithmic enhancements, along with some additional geometric lemmata expected to al-

[4]A subset S of the vertices of a graph G is a *homogeneous set* of G if, for all $v \in V(G)\backslash S$, either v is a neighbor of all vertices in S or v is a non-neighbor of all vertices in S.

[5]A subset S of the vertices of a graph G is a *clique* of G if all vertices of S are neighbors of one another in G.

low for even more efficient pruning.

4.3. Worst-case instances

Our method never goes any further than the granularity corresponding to McDiarmid and Müller's ϵ_{min}. It does not have to, since it can safely state that the input graph is not a UDG if it fails to find a realization in the ϵ_{min}-mesh. In [9], McDiarmid and Müller not only prove that every unit disk graph on n vertices admits an integer realization where all coordinates are at most $2^{2^{\Theta(n)}}$, but also prove the existence of UDGs which *only* admit integer realizations where at least one coordinate is $2^{2^{\Theta(n)}}$. These latter graphs are certainly worst-case instances for our method.

There is, however, a whole family $\mathcal{F}$ of graphs which, despite *not* being UDG, do admit an ϵ-discrete trigraph realization for all $\epsilon > 0$. Such graphs also constitute input instances for which our method is doomed to halt only at the ϵ_{min} granularity (with a NO answer, in this case).

Take, for instance, the complete bipartite graph[6] $K_{2,3}$, with vertex set $\{1,2\} \cup \{a,b,c\}$. We draw an equilateral triangle T of unitary side, and place vertices 1 and a coincidently on one of its corners, 2 and b on another corner, and vertex c on the remaining corner. Now, for some $\epsilon > 0$, we apply the discretization function f_ϵ on T, producing a (somewhat distorted) "ϵ-discrete equilateral triangle" T'. To check that the ensuing mapping is a trigraph realization of the $K_{2,3}$, just note that the distance between vertices placed at the same corner of T' is zero, describing a mandatory adjacency (unless $\epsilon > 1/\sqrt{2}$, in which case the adjacency is optional), and those vertices are indeed neighbors in the graph; and the distance between all other vertices fall in the range $(1-\epsilon\sqrt{2}, 1+\epsilon\sqrt{2})$, describing optional adjacencies and thus validating the model. The same reasoning applies to the $K_{3,3}$, whose vertex set $\{1,2,3\} \cup \{a,b,c\}$ may be positioned on the ϵ-mesh in exactly the same fashion as the $K_{2,3}$, if we just let the neighbors 3 and c coincide. The complete bipartite graph $K_{x,3}$, however, is not a UDG if $x \geq 2$ [6, Corollary 4.6], hence both the $K_{2,3}$ and the $K_{3,3}$ belong to $\mathcal{F}$. As for the $K_{3,4}$, our method was able to find a NO certificate at the $\epsilon = 7/80$ granularity.

[6]A *bipartite* graph G is a graph whose vertices are partitioned into two subsets A and B, such that all edges of G are incident to a vertex in A and a vertex in B. A *complete bipartite* graph, denoted $K_{x,y}$, is a bipartite graph with partite sets A and B, with $|A| = x, |B| = y$, such that every vertex in A is adjacent to every vertex in B.

Another example is the $K_{2,4}$, with vertex set $\{1,2\} \cup \{a,b,c,d\}$. If we place the vertices a, b, c and d at the corners of a rhombus consisting of two equilateral triangles with a common side of length 1, and we place the two remaining vertices at the endpoints of that common side, then we can now discretize those points to obtain an ϵ-discrete trigraph realization for all $\epsilon > 0$. Indeed, all adjacencies in the post-discretization trigraph will be optional, except possibly for mandatory adjacencies between the two coincident vertices at each endpoint of the shortest diagonal of the rhombus (the common side of the equilateral triangles) and a forbidden adjacency between the vertices at the endpoints of the longest diagonal. Since the former vertices are indeed neighbors, and the latter, non-neighbors, in the $K_{2,4}$, such adjacency types are in perfect conformity with the requirements of a trigraph realization. However, the $K_{2,4}$ is not a UDG, since it contains an induced $K_{2,3}$. It therefore belongs to $\mathcal{F}$.

Yet another member of $\mathcal{F}$ is the $K_{1,6}$. It consists of a central vertex adjacent to 6 pendant vertices. Place the central vertex of the $K_{1,6}$ at the center of a regular hexagon of unitary side, and each pendant vertex at a corner of that hexagon. By applying the discretization function f_ϵ to the center and corners of the hexagon, the resulting mapping is an ϵ-discrete trigraph realization of the $K_{1,6}$ regardless of ϵ. Indeed, all adjacencies between the central vertex and the pendant vertices will be optional in the associated trigraph, and the adjacencies between two pendant vertices—which are pairwise non-neighbors in the $K_{1,6}$—will be either optional or forbidden in the trigraph. The $K_{1,6}$, however, is not a UDG [8], hence it belongs to $\mathcal{F}$. A similar reasoning shows that the $K_{1,7}$ also belongs to $\mathcal{F}$. Because it contains an induced $K_{1,6}$, it is not a UDG. However, if we place its vertices prior to the discretization exactly as we did to the $K_{1,6}$, with its seventh pendant vertex placed coincidently with the central vertex, we obtain a trigraph realization. The $K_{1,8}$, which is not a UDG for the same reason as the $K_{1,7}$, does *not* belong to $\mathcal{F}$, and indeed a NO certificate was provided by our method, based on the fact that the $K_{1,8}$ admits no ϵ-discrete trigraph realizations when $\epsilon = 7/120$.

We can also argue that all co-bipartite graphs[7] that are not UDG belong to $\mathcal{F}$.[8] By placing the vertices of one of the cliques at a point and the vertices

[7] A co-bipartite graph is the complement of a bipartite graph, that is, its vertices can be partitioned into two cliques A and B, with whatever number of edges between A and B.

[8] We thank the anonymous referee for this accurate observation.

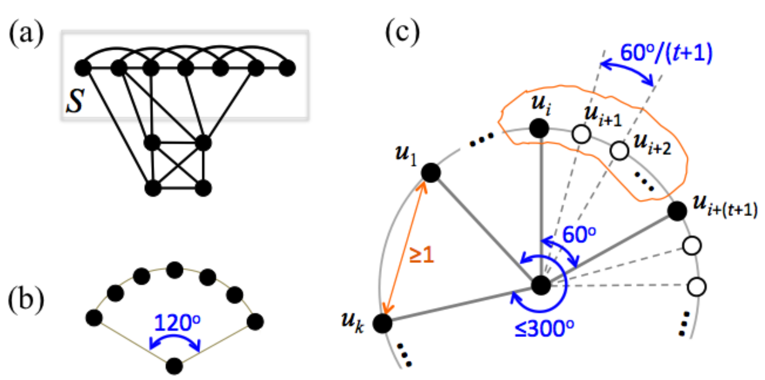

Figure 6: (a) The graph G, whose vertices are partitioned into an induced $(P_7)^2 =: S$ and a K_4; (b) pre-discretization placement of the vertices of G (the vertices of the K_4 are placed coincidently at the center of the arc); (c) general scheme for members of a subfamily of $\mathcal{F}$.

of the other clique at another point at distance exactly 1, the adjacencies between elements from different cliques will all be optional in the trigraph, no matter the granularity ϵ of the discretization function.

Finally, one can possibly come up with rather obscure subclasses of $\mathcal{F}$, so that an interesting characterization of $\mathcal{F}$ seems unlikely. The following lemma illustrates one such subclass.

Lemma 4. *If a graph G has an induced subgraph S which is the power*[9] *of a path, $S = (P_k)^t$, with $k \leq 5t + 6$, and $G \setminus S$ is a clique, then G admits an ϵ-discrete trigraph realization for all positive rational ϵ.*

Proof — Let $u_1, ..., u_k$ be the vertices of S in their order of appearance in P_k, and consider some positive $\epsilon \in \mathbb{Q}$. We construct an ϵ-discrete trigraph realization of G. Place the vertices u_i, for $i = 1, \ldots, k$, consecutively along the boundary of a circle of unitary radius, in such a way that u_j and u_{j+1}, for $j = 1, \ldots, k-1$, are the endpoints of an arc subtending an angle of $60/(t+1)$ degrees, and place the vertices of the clique $G \setminus S$ coincidently at the center of the circle, as illustrated in Figure 6. Now apply the discretization function f_ϵ to all those points. We show that the ensuing mapping ψ_ϵ is a trigraph

[9]The kth power G^k of a graph G is a supergraph of G formed by adding an edge between all pairs of vertices of G within distance at most k from one another.

realization of G. First, if v and w are two vertices of $G \setminus S$, then their images under ψ_ϵ are coincident, defining either a mandatory (if $\epsilon \leq 1/\sqrt{2}$) or optional (if $\epsilon > 1/\sqrt{2}$) adjacency in the trigraph model, and thus conforming to their being neighbors of each other in G. Suppose, now, that v belongs to S, and w to $G \setminus S$. Since the distance between v and w prior to the discretization was exactly 1 (the radius of the circle), after the discretization such distance belongs to the interval $(1-\epsilon\sqrt{2}, 1+\epsilon\sqrt{2})$, defining an optional adjacency, hence it does not matter whether v and w are neighbors in G. We are left with the adjacency type between two vertices u_i and u_j in S, for $1 \leq i < j \leq k$. If $j - i \leq t$, then u_i and u_j are neighbors in $(P_k)^t$, and the central angle $\alpha_{i,j}$ of the smallest arc whose endpoints were occupied by u_i and u_j before the discretization measures

$$\alpha_{i,j} = \min\left\{60 \cdot \frac{j-i}{t+1}, 360 - 60 \cdot \frac{j-i}{t+1}\right\}$$

degrees, which is less than 60 degrees for $|i - j| \leq t$. Consequently, the distance between those endpoints was less than 1, and the post-discretization distance $\|\psi_\epsilon(u_i) - \psi_\epsilon(u_j)\|$ is less than $1 + \epsilon\sqrt{2}$, satisfying the first condition in the definition of a trigraph realization (Section 3, Definition 2). If, on the other hand, $j - i \geq t + 1$, then u_i and u_j are neighbors in $(P_k)^t$, and $\alpha_{i,j}$ measures no less than 60 degrees, once $k \leq 5t+6$ implies $j-i \leq k-1 \leq 5t+5$ (the first term in the expression of $\alpha_{i,j}$ does not exceed 300 degrees). The pre-discretization distance between u_i and u_j was therefore no less than 1, and now $\|\psi_\epsilon(u_i) - \psi_\epsilon(u_j)\|$ is greater than $1 - \epsilon\sqrt{2}$, satisfying the second condition in Definition 2. ■

As an immediate corollary, if G satisfies the condition of Lemma 4 and G is not a UDG, then $G \in \mathcal{F}$.

5. Computational Results

We implemented our technique using the Python language[10], and we were able to classify several small graphs. Our implementation included all the enhancements discussed in Section 4.1. In all tests, we started from granularity $\epsilon = 0.7/2$ and, whenever needed, we used a *refine_granularity* function

[10]The code is available in `https://www.dropbox.com/s/v0if1zloa43bliq/udg_model_generator.py`.

which divides the current ϵ by 2. The sole exception was for input instance $K_{1,8}$, where we started from $\epsilon = 0.7/12$ (avoiding the high computational demands of $\epsilon = 0.7/16$ which would ensue after an inconclusive answer at $\epsilon = 0.7/8$).

5.1. Classifying particular graphs

Figures 7, 8, 9 and 10 depict the instances processed in our first experiment. They include, but are not limited to, all connected graphs (free of isomorphism) with up to 5 vertices and all connected graphs with 6 vertices and up to 8 edges. The results, presented in Tables 1 and 2, omit the coordinates of the UDG realizations for economy of space.

It is noteworthy that the only graph with less than 6 vertices which is not a UDG is the $K_{2,3}$, a graph that belongs to the family $\mathcal{F}$ of "pathological" graphs, as discussed in Section 4.3. We skipped that instance. As for graphs with 6 or more vertices, those which have not been classified by our method belong to one of two groups. The first group comprises those graphs which are known not to be UDG, since they contain an induced $K_{2,3}$. Each of them can easily be shown to belong to $\mathcal{F}$ (by following the same ideas we employed in Section 4.3), hence our method would take a long time to produce a NO certificate. We skipped those instances as well. The second group consists of a single graph that remains undecided. It satisfies the condition of Lemma 4. Therefore, if it is not a UDG, then it belongs to $\mathcal{F}$, and again a NO answer would only be produced by our method after exhausting all possible placements on the ϵ_{min}-mesh. Because that would be too much processing, we interrupted the execution midway through.

For 88 out of 98 instances, our method obtained a YES certificate quite instantly. The two NO answers, corresponding to the complete bipartite graphs $K_{3,4}$ and $K_{1,8}$, took considerably longer, even though they benefited largely from the second performance enhancement[11] discussed in Section 4.2.

5.2. Obtaining YES certificates for random graphs

Our first experiment confirmed that YES certificates are much easier to obtain than NO certificates. Indeed, for instances that are UDG, the algo-

[11]Since the partite sets of $V(G)$ in every complete bipartite graph G are homogeneous sets which are also independent sets, it was possible to restrict our search to candidate realizations where the image of vertices in a same partite set conformed with some fixed, predefined ordering.

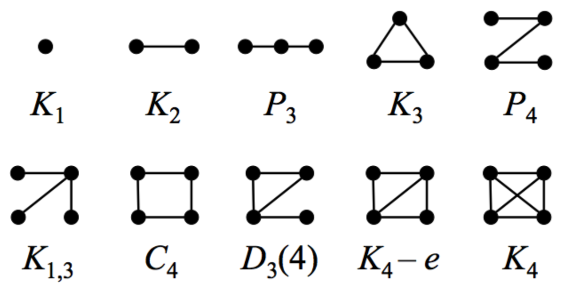

Figure 7: All connected graphs with less than 5 vertices.

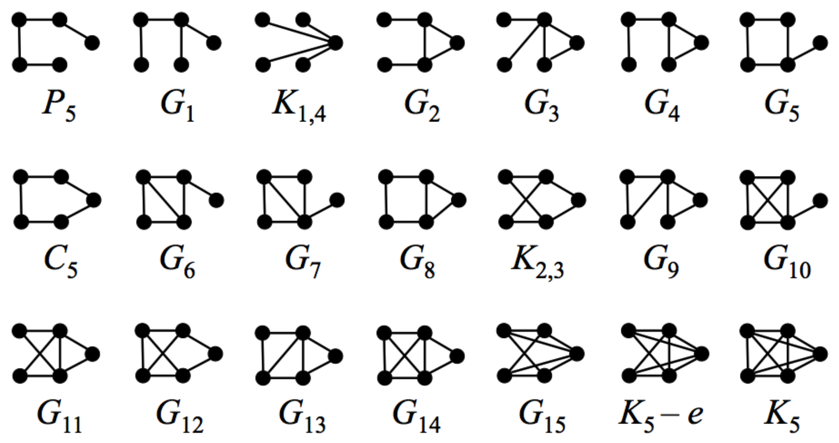

Figure 8: All connected graphs with 5 vertices.

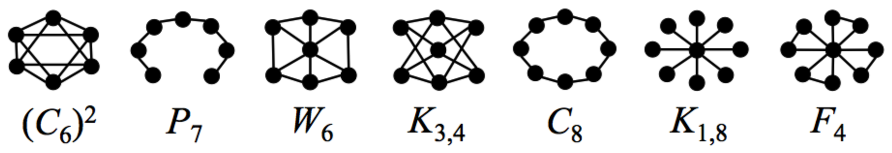

Figure 9: Some connected graphs with up to 9 vertices.

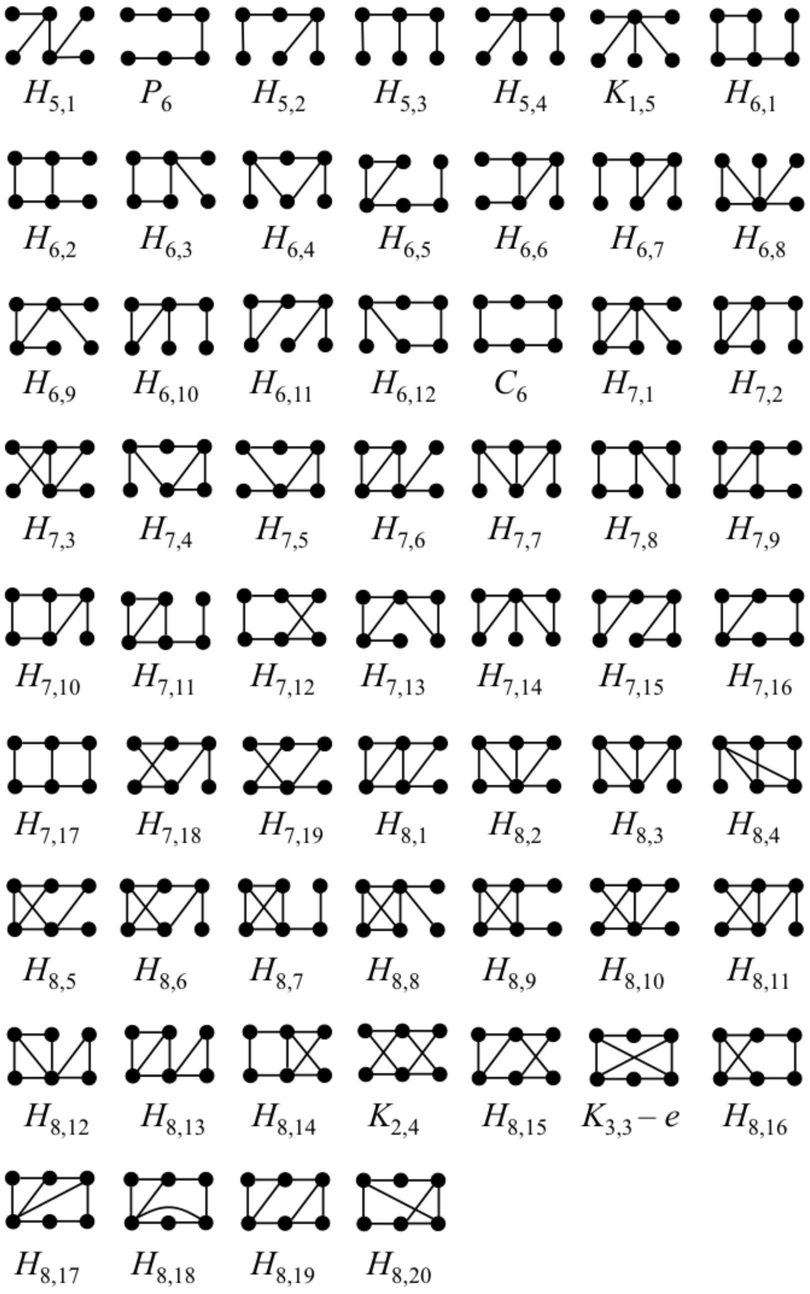

Figure 10: All connected graphs with 6 vertices and at most 8 edges.

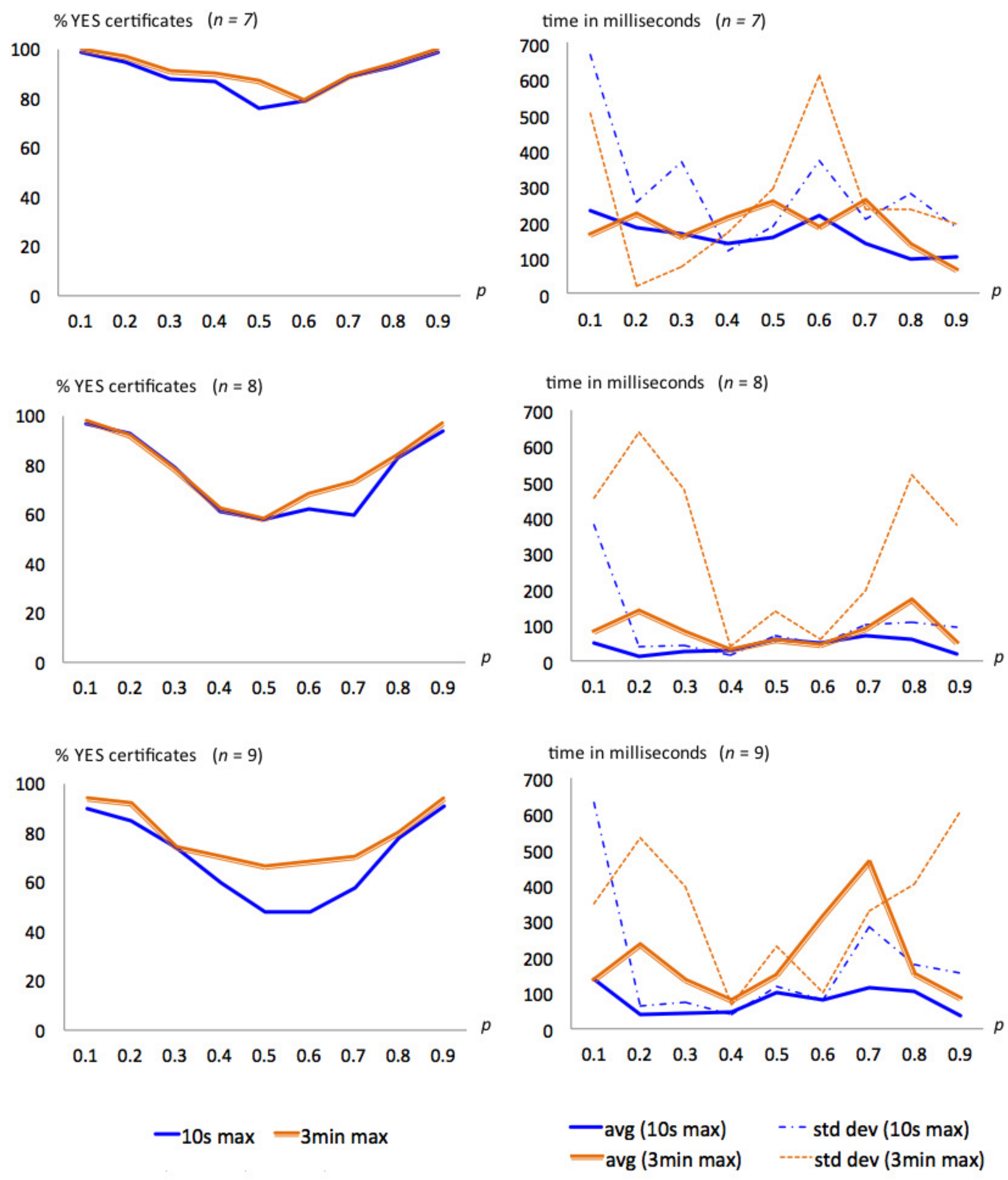

Figure 11: Output obtained for random graphs with 7, 8 and 9 vertices. The horizontal axis corresponds to the probability p of the $G_{n,p}$ model, and the results refer to 100 random instances of each $G_{n,p}$ model in all graphics. The graphics to the left indicate the number of YES certificates obtained within a given time limit. The graphics to the right indicate the average time and the standard deviation to produce a YES certificate.

input	time	UDG	$0.7/\epsilon$
K_1	0.249s	yes	2
K_2	0.217s	yes	2
P_3	0.245s	yes	2
K_3	0.266s	yes	2
P_4	0.228s	yes	2
$K_{1,3}$	0.264s	yes	2
C_4	0.235s	yes	4
$D_3(4)$	0.256s	yes	2
$K_4 - e$	0.250s	yes	2
K_4	0.250s	yes	2
P_5	0.215s	yes	2
G_1	0.198s	yes	2
$K_{1,4}$	0.185s	yes	4
G_2	0.197s	yes	2
G_3	0.180s	yes	2
G_4	0.181s	yes	2
G_5	0.180s	yes	2
C_5	0.196s	yes	4
G_6	0.196s	yes	2
G_7	0.181s	yes	2
G_8	0.179s	yes	2
$K_{2,3}$	$\in \mathcal{F}$	no	—
G_9	0.179s	yes	2
G_{10}	0.179s	yes	2
G_{11}	0.183s	yes	2
G_{12}	0.180s	yes	4
G_{13}	0.215s	yes	2
G_{14}	0.184s	yes	2
G_{15}	0.217s	yes	4
$K_5 - e$	0.218s	yes	2
K_5	0.224s	yes	2

Table 1: Output obtained for graphs in Figures 7 and 8. The third column indicates whether the input graph is a UDG. The $K_{2,3}$ belongs to the family $\mathcal{F}$—as discussed in Section 4.3—and was intentionally skipped.

rithm stops whenever it finds a UDG realization. On the other hand, for graphs that are not UDG, the algorithm must exhaust all possible placements at some granularity—having found no trigraph realizations—before it outputs a NO answer. Moreover, worst-case YES instances are apparently rarer (see Section 4.3). In practice, if one knows beforehand that some small graph G is realizable—as seems to be the case in most Distance Geometry applications—then one can expect our method to find a realization for G in reasonable time with high probability.

In our second experiment, we employed our method to classify a number of random graphs with $7, 8$ and 9 vertices from the $G_{n,p}$ model, where an instance G is generated with a predefined number n of vertices, and where an edge vw belongs to G with probability p for all $v, w \in V(G)$. We did not consider disconnected instances. Furthermore, we defined a maximum allowed execution time per instance, after which we would halt the execution and accept an inconclusive answer. Had we not done so, the overall time

input	time	UDG	$0.7/\epsilon$
$H_{5,1}$	0.249s	yes	2
P_6	0.247s	yes	2
$H_{5,2}$	0.276s	yes	2
$H_{5,3}$	0.233s	yes	2
$H_{5,4}$	0.267s	yes	4
$K_{1,5}$	0.859s	yes	8
$H_{6,1}$	0.271s	yes	2
$H_{6,2}$	0.183s	yes	2
$H_{6,3}$	0.198s	yes	4
$H_{6,4}$	0.195s	yes	2
$H_{6,5}$	0.180s	yes	2
$H_{6,6}$	0.179s	yes	2
$H_{6,7}$	0.180s	yes	2
$H_{6,8}$	0.180s	yes	4
$H_{6,9}$	0.180s	yes	4
$H_{6,10}$	0.179s	yes	2
$H_{6,11}$	0.179s	yes	2
$H_{6,12}$	0.196s	yes	4
C_6	0.179s	yes	2
$H_{7,1}$	0.181s	yes	4
$H_{7,2}$	0.196s	yes	2
$H_{7,3}$	0.179s	yes	4
$H_{7,4}$	0.181s	yes	2
$H_{7,5}$	0.231s	yes	4
$H_{7,6}$	0.197s	yes	2
$H_{7,7}$	0.216s	yes	2
$H_{7,8}$	0.223s	yes	4
$H_{7,9}$	0.232s	yes	2
$H_{7,10}$	0.231s	yes	4
$H_{7,11}$	0.218s	yes	2
$H_{7,12}$	$*$	?	—
$H_{7,13}$	0.225s	yes	2
$H_{7,14}$	0.217s	yes	2
$H_{7,15}$	0.234s	yes	2
$H_{7,16}$	0.202s	yes	4
$H_{7,17}$	0.198s	yes	4
$H_{7,18}$	$\in \mathcal{F}$	no	—
$H_{7,19}$	$\in \mathcal{F}$	no	—
$H_{8,1}$	0.184s	yes	2
$H_{8,2}$	0.180s	yes	4
$H_{8,3}$	0.180s	yes	2
$H_{8,4}$	0.250s	yes	4
$H_{8,5}$	0.180s	yes	2
$H_{8,6}$	0.179s	yes	2
$H_{8,7}$	0.181s	yes	2
$H_{8,8}$	0.180s	yes	2
$H_{8,9}$	0.196s	yes	2
$H_{8,10}$	0.213s	yes	4
$H_{8,11}$	0.179s	yes	2
$H_{8,12}$	0.180s	yes	2
$H_{8,13}$	0.180s	yes	2
$H_{8,14}$	0.247s	yes	4
$K_{2,4}$	$\in \mathcal{F}$	no	—
$H_{8,15}$	$\in \mathcal{F}$	no	—
$K_{3,3} - e$	$\in \mathcal{F}$	no	—
$H_{8,16}$	$\in \mathcal{F}$	no	—
$H_{8,17}$	0.200s	yes	2
$H_{8,18}$	0.222s	yes	2
$H_{8,19}$	0.214s	yes	2
$H_{8,20}$	0.418s	yes	4
$(C_6)^2$	0.300s	yes	4
P_7	0.286s	yes	2
W_6	0.320s	yes	4
$K_{3,4}$	1h54min	no	8
C_8	0.319s	yes	2
$K_{1,8}$	22h56min	no	12
F_4	0.494s	yes	4

Table 2: Output for graphs in Figures 9 and 10. The instance $H_{7,12}$, marked "$*$", satisfies the condition of Lemma 4 (it consists of an induced P_4 and a K_2) and remains undecided. Instances marked "$\in \mathcal{F}$" present an induced $K_{2,3}$ and are known to belong to the family $\mathcal{F}$.

to decide thousands of instances would be prohibitive. Still, a considerable number of YES certificates were produced rather quickly. Since we employed a random graph model whose instances are not always unit disk graphs, the actual ratio of fast YES answers produced by our method must be even higher if the input is restricted to graphs known to be UDG. The results displayed in Figure 11 can therefore be regarded as estimates on lower bounds for the probability that our method finds a realization for unit disk graphs of different sizes in reasonable time.

It is also interesting to notice that higher ratios of YES certificates were obtained when the expected density of the $G_{n,p}$ instance—which grows with p—was either too low or too high. This result probably matches one's intuition, since the vertices of too sparse a graph have but a few neighbors imposing proximity constraints, while the vertices of a dense graph may flock together without too many non-neighborhood restrictions.

6. The Distance Geometry Problem with Ranges

The promising results presented in Section 5 invite us to leverage the proposed technique to the Distance Geometry Problem with Ranges, where each pair of adjacent vertices in a graph G is assigned its own interval of accepted distances. No distance restrictions are imposed over pairs of vertices that are not adjacent in G. One looks for a realization of G respecting those accepted distances in some (continuous) metric space. For ease of exposition, we henceforth assume the latter to be the Euclidean plane.

In Section 3, we introduced the idea of a discrete trigraph realization. Such concept, which allows for a pragmatic discretization of the space, extends naturally to the much more general DGPR. One defines a discretization function and obtains an upper bound, as tight as possible, to the pairwise relative displacement it may produce. Then the accepted intervals are relaxed just enough to incorporate that maximal relative displacement. If, even with such a relaxation, a certain placement of vertices is not at all valid, then a whole bunch of otherwise candidate solutions (on the continuous space) can be immediately ruled out. Some additional difficulties appear, though, when arbitrary ranges are allowed.

A first obstacle is the existence of perfectly realizable instances which admit no realization *with only rational coordinates.* An easy example is the complete graph on 3 vertices whose edges are assigned the same interval $[1, 1]$ of accepted distances. A unitary equilateral triangle is a realization (the only one) for such an instance, yet at least one coordinate will be irrational. The following three examples—actually, three "variations on a theme"—give a bit more of intuition on the existence of such irrational DGPR instances, even when the ranges are not trivial intervals.

The first example is a K_7 (the complete graph on 7 vertices) with an edge set partitioned into two groups. The edges of the first group induce a $K_{1,6}$ and are assigned the interval $[0, 1]$ of accepted distances; those of the second group—all remaining edges—are assigned the range $[1, \infty)$.[12] Such a DGPR instance has a unique realization (short of rotations, translations and reflections), where the coordinates are $(0, 0)$, $(1, 0)$, $(1/2, \sqrt{3}/2)$, $(-1/2, \sqrt{3}/2)$, $(-1, 0)$, $(-1/2, -\sqrt{3}/2)$ and $(1/2, -\sqrt{3}/2)$. The second example consists again of a K_7 with two groups of edges: those which induce a 6-wheel[13] and are assigned the range $[0, 1]$; and the remaining edges, which are assigned the range $[1, 2]$. This instance also admits a unique realization, with exactly the same coordinates as those in the previous example. The third example consists yet again of a K_7, but now all edges get the interval $[1, 2]$ of accepted distances. The only possible realization for such a DGPR instance is the same one as in the previous examples.

It is clear that our method shall not find a realization for such instances. This poses the following question:

- For a positive integer d, which DGPR realizable instances admit a realization (in $\mathbb{R}^d$) with only rational coordinates?

A sufficient condition, so that small perturbations may lead each vertex with irrational coordinates to a nearby point with only rational coordinates, is given by the following lemma:

Lemma 5. *Let G be a DGPR instance whose distance intervals are all open, or all left-open, or all right-open, and let d be a positive integer. If G admits a realization in $\mathbb{R}^d$, then G admits a realization in $\mathbb{R}^d$ where all coordinates are rational.*

Proof — Let ϕ be a realization of G in $\mathbb{R}^d$. We first consider the case where all intervals are open, i.e. they are both left- and right-open. Suppose $\phi(v)$,

[12]Notice the similarity between this instance and a typical instance for the UDG problem, where the ranges are either $[0, 1]$ or $(1, \infty)$.

[13]A k-wheel is a k-cycle plus a $(k + 1)$th vertex adjacent to all vertices in the cycle. The graph labeled W_6 in Figure 9 is a 6-wheel.

for some $v \in V(G)$, has an irrational coordinate, and let $w_1, \ldots, w_{d(v)}$ be the neighbors of v in G, with (a_i, b_i) the range of acceptable distances associated to each edge vw_i, for $i = 1, \ldots, d(v)$. The locus B_i of all candidate locations for v with respect to w_i is the difference of two balls centered in $\phi(w_i)$, an open one with radius b_i and a closed one with radius a_i, hence B_i is an open set. Consequently, the intersection $B = \cap_{i=1,\ldots,d(v)} B_i$ is also an (non-empty, because ϕ is a realization of G) open set. We can therefore replace the irrational image $\phi(v)$ by any point in B with only rational coordinates—by the density of rationals, there exists such a point—and the ensuing mapping is still a realization of G. By repeating this procedure for each point with an irrational coordinate, we obtain the desired realization of G with only rational coordinates.

Now we consider the case where some of the intervals are right-closed, so that each edge $vw \in E(G)$ is associated to an interval which is either (a_{vw}, b_{vw}) or $(a_{vw}, b_{vw}]$. If we define c as

$$c = \max_{vw \in E(G)} \left\{ \frac{a_{vw}}{\|\phi(v) - \phi(w)\|} \right\},$$

we can multiply all coordinates in the image of ϕ by $(c+1)/2$ to obtain a realization ϕ' of G where $\|\phi'(v) - \phi'(w)\| < b_{vw}$ for all $v, w \in V(G)$. As a consequence, it is always possible to replace the original intervals by intervals that are open on both sides. By repeating now the argument for open intervals, we infer the existence of a realization of G which is free of irrational coordinates.

The case where some of the intervals are left-closed is analogous. ■

If one of the conditions in Lemma 5 are met, the existence of a solution with only rational coordinates ensures that a slightly modified Algorithm 1 can find a realization of G in finite time *if a realization exists*. However, we must also determine a criterion that allows the algorithm to stop even when the input graph G is *not* realizable. In the case of unit disk graphs, the result of McDiarmid and Müller provides the most refined granularity at which we must perform our search. Unfortunately, we know of no analogous for the more general DGPR. And so we pose the next question, borrowing the idea of an ϵ-discrete realization from Section 3.

- For which DGPR instances G can one determine a rational ϵ (or, better yet, the maximum rational ϵ) such that, if G is realizable, then G admits an ϵ-discrete realization?

Here again we are able to establish a sufficient condition on the ranges of the input instance. The main tool is a result from Grigor'ev and Vorobjob [5] that determines the (absolute) size of the solution of a system of polynomial inequalities. We recall such result in the form given in [9].

Lemma 6 (Grigor'ev & Vorobjob [5]). *For each $d \in \mathbb{N}$ there exists a constant $C = C(d)$ such that the following hold. Suppose that $h_1, ..., h_k$ are polynomials in n variables with integer coefficients and degrees $deg(h_i) < d$. Suppose further that the bitsizes of all coefficients are less than B. If there exists a solution $(x_1, ..., x_n) \in \mathbb{R}^n$ of the system $\{h_1 \geq 0, ..., h_k \geq 0\}$, then there also exists one with $|x_1|, ..., |x_n| \leq \exp[(B + \ln k)C^n]$.*

Theorem 7. *Let G be a realizable instance of the DGPR with $V(G) = \{v_i\}_{i=1}^n$, and let (a_{ij}, b_{ij}) be the range associated to each edge $v_iv_j \in E(G)$. If, for all i, j, the bounds a_{ij} and b_{ij} are rational numbers not greater than some fixed constant c, then G admits a realization where all coordinates are rational numbers with bitsize $\mathcal{O}(2^n)$.*

Proof — The realizability of a DGPR instance G obviously does not change if we multiply all the distance ranges by some constant factor $k > 0$. Therefore, there is a clear correspondence between a realization ϕ of G and an integer solution $(x_1, y_1, \ldots, x_n, y_n, k)$ of the system of $2|E(G)| + 1$ inequalities

$$\begin{cases} (x_i - x_j)^2 + (y_i - y_j)^2 > (ka_{ij})^2 \\ (x_i - x_j)^2 + (y_i - y_j)^2 < (kb_{ij})^2 \\ k > 0, \end{cases}$$

where the two first inequalities apply to all i, j such that $v_iv_j \in E(G)$. Indeed, if we let $\phi(v_i) = (x_i/k, y_i/k)$ for all i, then ϕ is a realization of G with only rational coordinates. Furthermore, if all integers in the solution of such system have bitsizes $\mathcal{O}(2^n)$, then so does each coordinate in the image of ϕ.

To prove the $\mathcal{O}(2^{2^n})$ bound for the absolute value of those integers (or, equivalently, the $\mathcal{O}(2^n)$ bound for their bitsizes), we follow the same argument, based on Lemma 6, that was used in [9] for realizations of unit disk graphs with bounded integer coordinates. We are in a position to do that because all our distance ranges are open intervals, that is, all the inequalities in the above system are strict. After replacing the r^2's in McDiarmid and Müller's original text [9, proof of Lemma 7.4] with the appropriate $(ka_{ij})^2$'s and $(kb_{ij})^2$'s, and not without some tedious manipulations, we obtain the intended bound. ∎

Theorem 7 implies that the DGPR analogous of Algorithm 1 has a NO-answer stopping criterion if all ranges are open intervals. However, a simple scaling argument identical to the one used in the proof of Lemma 5 proves it sufficient that all intervals are left-open (or all intervals are right-open).

Another important question regards the complexity of the DGPR. After all, as discussed in Section 2, not even when restricted to ranges $[0, 1]$ and $(1, \infty)$ is the DGPR known to be in **NP**. Considering the above observations about the inexistence of rational realizations for certain realizable inputs, it is natural to consider the following question:

- What is the complexity of the DGPR?

In addition to the above computational questions, the DGPR brings about at least one additional "geometric flavored" question that establishes a connection between Distance Geometry and Combinatorial Geometry:

- What is the maximum number of points that can be placed at pairwise distance at least d and at most $D - d$?

This is the case where the DGPR instance is a complete graph with all ranges $[d, D - d]$, and corresponds precisely to the problem of packing disks of diameter d in a circle of diameter D.

7. Conclusion and future directions

A slight modification in the definition of the Distance Geometry Fundamental Problem disclosed a connection with a classic recognition problem in Graph Theory. We believe that techniques for recognizing and obtaining realizations for some classes of geometric graphs can contribute to—and benefit from—the ever-growing repertoire of tools being put together in the field of Distance Geometry.

Proving that a graph is a UDG is fairly simple when one knows a realization of the graph, a natural certificate. On the other hand, no method other than exhausting a huge discrete space whose coordinates have $2^{2^{\Theta(n)}}$ bits was known so far for proving that a graph is *not* a UDG. In this sense, the possibility of exhibiting a computational certificate for a NO answer in affordable time—even if restricted to graphs of modest size—may probably be regarded as the main contribution of our trigraph-based method.

Two are the main research lines that open up from the current work. First, there are plenty of reasonably small graphs—for example, those in Figures 2(a) and 2(c)—whose membership to the UDG class remains unknown. Those graphs certainly deserve definitive answers. The proposed approach stands good chances of providing such answers, specially in light of numerous promising improvements that may still be attempted on the current pruning criteria (see Section 4.2). Second, the generalization of the Distance Geometry Fundamental Problem into the form of the Distance Geometry Problem with Ranges—a formulation that shows a strong potential for applications—evidences the connections between Distance Geometry, Graph Theory and Combinatorial Geometry. We believe that several nice theoretical problems might show up once these connections are more clearly established.

Acknowledgements

We thank László Kozma for the inspiring ideas and insightful discussions. We also thank the anonymous referees for so many valuable suggestions.

References

[1] A. Andrioni, R. de Freitas, C. Lavor, L. Liberti, N. Maculan, A. Mucherino. Workshop on Distance Geometry and Applications (DGA 2013). `http://dga2013.icomp.ufam.edu.br/`.

[2] H. Breu, D. G. Kirkpatrick, Unit disk graphs recognition is **NP**-hard, Comp. Geom. 9(1–2) (1998) 2–24.

[3] G. D. da Fonseca, C. M. H. de Figueiredo, V. G. Pereira de Sá, R. C. S. Machado, Linear-time approximation for dominating sets and independent dominating sets in unit disk graphs, in Workshop on Approximate and Online Algorithms, Lect. Notes Comp. Sci. 7846 (2013) 82–92.

[4] G. D. da Fonseca, C. M. H. de Figueiredo, V. G. Pereira de Sá, R. C. S. Machado, Efficient sub-5 approximations for dominating sets in unit disk graphs, Theor. Comput. Sci. 540–541 (2014) 70–81.

[5] D. Y. Grigor'ev, V. V. Vorobjov, Solving systems of polynomial inequalities in subexponential time, J. Symb. Comput. 5(1–2) (1988) 37–64.

[6] E. J. van Leeuwen, Approximation algorithms for unit disk graphs, Tech. Rep. UU-CS-2004-066, Institute of Information and Computing Sciences, Utrecht University (2004).

[7] L. Liberti, C. Lavor, N. Maculan, A. Mucherino, Euclidean distance geometry and applications, SIAM Review 56(1) (2014), 3–69.

[8] M. V. Marathe, H. Breu, H. B. Hunt III, S. S. Ravi, D. J. Rosenkrantz, Simple heuristics for unit disk graphs, Networks 25(2) (2005) 59–68.

[9] C. McDiarmid, T. Müller, Integer realizations of disk and segment graphs, J. Comb. Theory, Series B 103(1) (2013) 114–143.

[10] J. Spinrad, Efficient Graph Representations, Fields Inst. monographs (2003), AMS.

[11] W. Wu, H. Du, X. Jia, Y. Li, S. C.-H. Huang, Minimum connected dominating sets and maximal independent sets in unit disk graphs, Theor. Comput. Sci. 352(1–3) (2006) 1–7.