Acta Math. Univ. Comenianae
Vol. LXXXVIII, 3 (2019), pp. 967–972

ON UNIT GRID INTERSECTION GRAPHS AND SEVERAL OTHER INTERSECTION GRAPH CLASSES

I. MUSTAŢĂ AND M. PERGEL

Abstract. We explore what could make recognition of particular intersection-defined classes hard. We focus mainly on unit grid intersection graphs (UGIGs), i.e., intersection graphs of unit-length axis-aligned segments and grid intersection graphs (GIGs, which are defined like UGIGs without unit-length restriction). As side effects we obtain several further nontrivial results.

We show that the explored graph classes are NP-hard to recognized even when restricted to graphs with arbitrarily large girth, i.e., length of a shortest cycle. Next we show that the recognition of these classes remains hard even for graphs with restricted degree (4, 5 and 8 depending on a particular class). For UGIGs we present structural results on the size of a possible representation, too.

1. Introduction

Geometric intersection graphs are graphs with a geometric representation where each vertex is represented by a geometric object and the adjacency of a pair of vertices corresponds to the fact that the objects have a nonempty intersection. They are a practically important part of graph theory as some generally hard problems become efficiently solvable on some intersection classes. These efficient algorithms usually require an intersection representation instead of a graph, which motivates the recognition problem, i.e., the question of whether a given graph has a desired intersection representation.

Our attention focuses on unit grid intersection graphs, called UGIG, i.e., intersection graphs of unit-length axis-aligned line segments in the plane. Our results have influence even on other graph-classes, namely on grid intersection graphs that we denote as GIG defined similarly to UGIG, just without the restriction to unit length [**1, 4, 10**]. Grid intersection graphs have been studied also as PURE-2-DIR or B0-VPG. Obviously UGIGs form a subclass of GIGs. GIGs can be generalized to segment graphs, intersection graphs of straight-line segments in the plane. The topological versions of segment graphs are pseudosegment graphs where pseudosegments do not have to be straight, they just have to keep the topological properties of segments [**2**]. Even this class can be generalized to the class of string graphs, intersection graphs of arc-connected curves in the plane [**5**].

Received May 24, 2019.
2010 *Mathematics Subject Classification.* Primary 97P20, 97K20, 97K30, 03D15.
This research was partially supported by the Czech Science Foundation grant GA19-08554S.

As there exist many classes of intersection graphs, it is useful to avoid exploring each class separately. For optimization problems, typically an efficient algorithm for a superclass yields an algorithm for subclasses. For the recognition problem this is not true. Due to this fact, *sandwiching* was introduced [**7**]. It is an approach close to approximability/inapproximability. A class $\mathcal{B}$ is sandwiched between classes $\mathcal{A}$ and $\mathcal{C}$, if $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{C}$. Given a reduction that produces either graphs from $\mathcal{A}$ or not even graphs from $\mathcal{C}$, this reduction shows hardness for all classes sandwiched between $\mathcal{A}$ and $\mathcal{C}$.

About the recognition of the mentioned classes, many results are known. The recognition problem of string graphs is NP-complete even for graphs without triangles [**9, 6**], for segment and pseudosegment graphs the problem is known to be NP-hard even for graphs with arbitrarily large girth [**7**], it is not known whether the problem is in NP and Cartesian coordinates of endpoints of particular segments in a representation cannot be used as a polynomial certificate of representability [**8**]. It is known that GIGs are NP-complete to get recognized [**5**]; the same article shows that any class between 3-DIR (intersection graphs of straight line segments each using one of three permitted directions) is NP-hard to recognize.

Because for many intersection-defined classes the recognition problem is hard, we are trying to explore the reasons. The idea behind graphs without triangles or with high girth is that it is some form of density (of edges) that makes the recognition hard. This holds, e.g., for polygon-circle graphs, not, e.g., segment graphs [**7**]. As the "density" tries to restrict the number of edges, we tried to restrict it by bounding the maximum degree (as it efficiently bounds the number of edges in the graph). We show that all explored classes are rather resistant to this criterion.

2. The results

Our main result whose proof we sketch in the following section is:

Theorem 1. *Between the following three pairs of graph-classes no polynomially recognizable class can be sandwiched even when restricted on graphs with arbitrarily large girth and yet on graphs with the maximum degree (at most)* 5, 4 *and* 8, *respectively:*

- *UGIG and pseudosegment graphs,*
- *GIG and segment graphs,*
- *GIG and string graphs.*

In the rest of this section we show upper and lower bound on the size of a UGIG representation:

Proposition 1. *Any unit grid intersection graph on n vertices can be represented in a grid $(n+1)\times(n+1)$ with all coordinates being multiples of $\frac{1}{n}$. Moreover, we can find such a representation that with respect to each axis no two segments have the same non-integral part of the coordinate.*

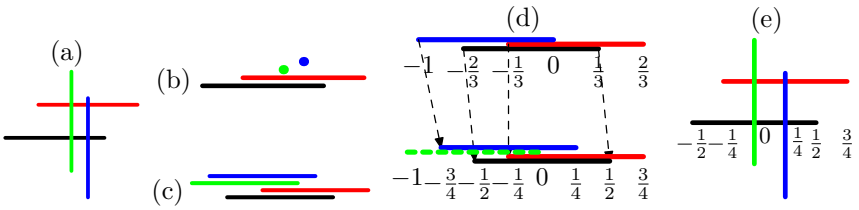

Figure 1. Considering the arrangement from picture (a) we make a projection onto one axis (b), i.e. perpendicular segments collapse into single points. We extend them in one direction (in our case to the left), see picture (c). Picture (d) shows the last iteration of the fourth step of the sweeping algorithm, i.e. extending the so far obtained representation. Picture (e) shows the final representation (after having also swept along the y-axis).

Note that the lower bound for the granularity (multiples of $\frac{1}{n}$) is tight because of $K_{1,n-1}$ which requires this precision for the (distinct) coordinates.

Proof. Will be in the journal version, sketched by Figure 1. □

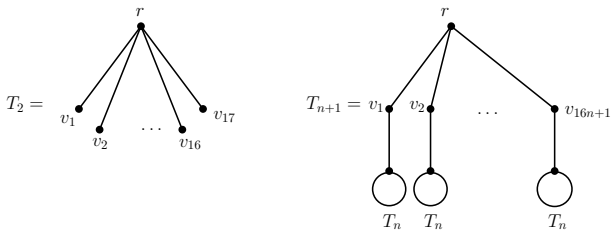

Figure 2. Constructing the family of trees.

In the sequel, we define the *boundary size* as the semiperimeter of the bounding rectangle and by induction (details will be in the journal version) show for trees from Figure 2:

Theorem 2. *For all $n \geq 2$, a UGIG representation of T_n needs a boundary size of at least n.*

As a corollary of Proposition 1, Theorem 1 and a trivial observation, classes UGIG and GIG are NP-complete to recognize even with arbitrarily large girth.

3. The reductions

Here, we sketch the proof of Theorem 1. We reduce planar 3-connected 3-SAT(4) shown to be NP-complete [**6**]. This is a special version of 3-SAT where each variable occurs at most 4 times and the incidence graph (of the formula) is planar and 3-connected (3-connectivity implies a unique planar embedding up to the outer face). The incidence graph is bipartite with one part formed by variables, the other by clauses and an edge means a presence of a variable in a clause. We follow

the ideas of [**7, 3**], i.e., for a planar embedding of the incidence graph, vertex representatives get represented by variable-gadgets or truth-splitters, clause representatives we replace by clause gadgets and the edges (of the incidence graph) we replace by pairs of paths whose left-right orientation represent the truth-assignment.

Variable-gadgets must keep the occurrences synchronized, clause-gadgets must be representable exactly for 7 satisfiable assignments. For the 1st case, as a variable gadget we take two vertices connected by an edge and each pair representing an occurrence stems from them to incur the situation of Figure 4. A clause-gadget is depicted in Figure 3, dashed curves depict arbitrarily long paths. For the 2nd case we use the same clause-gadget, but to decrease the maximum degree in the variable-gadget, we stick the 1st and the 2nd occurrence together and we represent the 3rd and 4th by the truth-splitter as depicted in Figure 5 without vertex c. For the 3rd case, we use the truth-splitter in Figure 5 (including vertex c) and a clause-gadget from Figure 6.

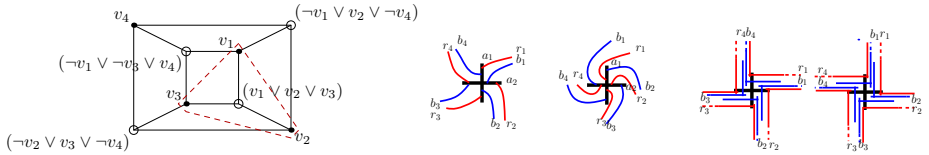

Figure 3. Left: Incidence graph for a formula $(v_1 \vee v_2 \vee v_3) \wedge (\neg v_2 \vee v_3 \vee \neg v_4) \wedge (\neg v_1 \vee v_2 \vee \neg v_4) \wedge (\neg v_1 \vee \neg v_3 \vee v_4)$ (here we have identified the vertex names with the variables/clauses). Middle: Subgraph of the incidence graph for $v_1 \vee v_2 \vee v_3$. Right: The corresponding subgraph whose representability we explore. This subgraph shows the part inside the red dashed triangle in the left picture. Left-right orientation of blue and red vertices/paths determines the truth assignment .

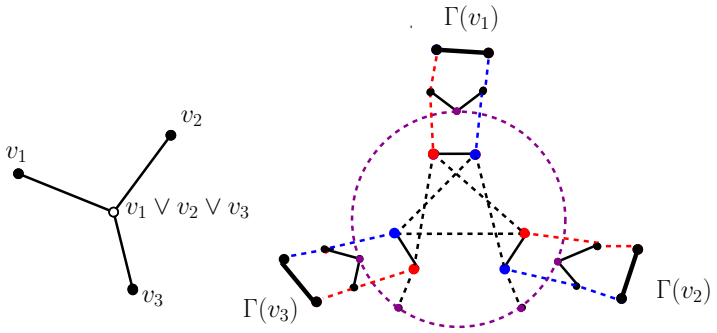

Figure 4. The two distinct representations of the four occurrences incident to a variable gadget. The left pair is a pseudosegment representation, the right one a UGIG representation. In each pair, the left corresponds to the FALSE assignment, the right one to the value TRUE.

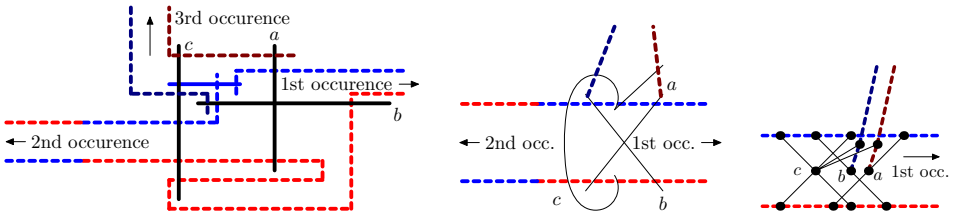

Figure 5. Truth-splitter gadget. First occurrence always goes to the right, second to the left, third upwards. Left: GIG representation, middle: string representation, right: the appropriate graph.

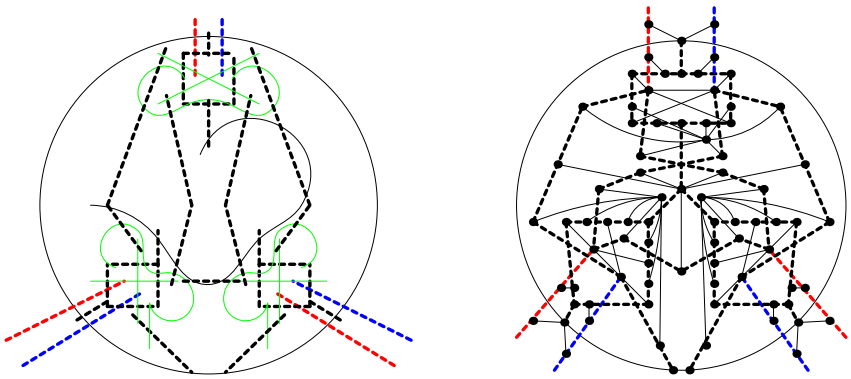

Figure 6. Clause gadget for string graphs. Note that the vertex with maximum degree corresponds to the top green curve (left: representation, right: graph).

4. Conclusion and open problems

We have defined a new measure that can influence the hardness of the recognition problem for graph classes whose recognition is generally hard and we tried to get as tight results for particular graph classes as possible. For several classes, we have almost succeeded, usually 1 or 2 values are unclear. This fact motivates the open problems:

What is the complexity of recognizing for string graphs, segment graphs and GIGs (and preferably all classes between them) when we restrict our attention to graphs with maximum degree 3? How does the answer change when we restrict only to graphs with large girth (note that our reductions are meeting both criteria simultaneously)? We may also ask, what is the complexity of recognizing UGIGs with maximum degree 3 or 4 (both are open so far, our reduction works for degree at least 5). Further we may ask the same problem for string graphs with arbitrarily large girth and maximum degrees from 3 to 7.

For what graph classes does this new measure help more and which classes remain hard even with low degrees? What happens, e.g., with polygon-circle

graphs (that can be polynomially recognized when restricted on graphs with girth at least 6) where the reduction requires vertices of high degree?

What makes the recognition problem hard - namely for all classes between GIGs and pseudosegments?

Thanks to Adam Dingle for proofreading the text.

References

1. Chaplick S., Hell P., Otachi Y., Saitoh T. and Uehara R., *Ferrers dimension of grid intersection graphs*, Discrete Appl. Math. **216** (2017), 130–135.

2. Dangelmayr C., Felsner S. and Trotter W. T., *Intersection graphs of pseudosegments: Chordal graphs*, J. Graph Algorithms Appl. **14** (2010), 199–220.

3. Felsner S., Mustata I. and Pergel M., *The complexity of the partial order dimension problem: Closing the gap*, SIAM J. Discrete Math. **31** (2017), 172–189.

4. Hartman I. B.-A., Newman I. and Ziv R., *On grid intersection graphs*, Discrete Math. **87** (1991), 41–52.

5. Kratochvíl J., *String graphs. II. Recognizing string graphs is np-hard*, J. Combin. Theory Ser. B **52** (1991), 67–78.

6. Kratochvíl J., *A special planar satisfiability problem and a consequence of its np-completeness*, Discrete Appl. Math. **52** (1994), 233–252.

7. Kratochvíl J. and Pergel M., *Geometric intersection graphs: Do short cycles help?*, in: Computing and Combinatorics, COCOON 2007 (Lin G., eds.), Lecture Notes in Comput. Sci. 4598, Springer, 2007, 118–128

8. Kratochvíl J. and Matoušek J., *Intersection graphs of segments*, J. Combin. Theory Ser. B **62** (1994), 289–315.

9. Schaefer M., Sedgwick E. and Štefankovič D., *Recognizing string graphs in np*, J. Comput. System Sci. **67** (2003), 365–380.

10. Uehara R., *Simple geometrical intersection graphs*, in: WALCOM: Algorithms and Computation, WALCOM 2008 (Nakano S., Rahman M.S., eds), Lecture Notes in Comput. Sci. 4921, Springer, 25–33.

I. Mustață, Berlin Institute of Technology, Institut für Mathematik, Berlin, Germany; funded by Berlin Math. School,
e-mail: innereyes@gmail.com

M. Pergel, Department of Software and Computer Science Education (KSVI), Charles University, Prague, Czech Republic,
e-mail: perm@kam.mff.cuni.cz