Charles University in Prague

Faculty of Mathematics and Physics

Doctoral Thesis

Martin Pergel

Special Graph Classes and Algorithms on Them

Department of Applied Mathematics

Advisor:
Prof. RNDr. Jan Kratochvíl, CSc.

Study program:
I4 – Discrete models and algorithms

Acknowledgements

I am grateful to my advisor, Jan Kratochvíl. As well I would like to thank my friends, who were supporting me against all disturbing events. Among them mainly Johanka and Mirek Spoustovi. Many thanks also go to all my colleagues working on similar or related topics who were discussing with me a lot, namely and mainly to Cornelia Dangelmayr, Jessica Enright, Mareike Massow, Peter Allen and Mike Newman. As well I am thankful to Stefan Felsner and Lorna Stewart for performing research on related topics and permitting me to cooperate with their workgroups. Also, it is necessary to thank to my basic- and secondary-school teachers of mathematics, Tomáš Már and Václav Beneda, whose effort motivated me to focus on this topic. Simultaneously, I apologize to all whom I forgot to mention on this place and who deserve it, nevertheless, I am grateful for their help.

Preface

This thesis deals with several problems on intersection graphs, namely with problems related to recognition of particular intersection classes. Our results can be divided into three parts. The first part contains results on finding efficient intersection representations for particular classes. We cover this in Chapter 2; this is based on [1, 3].

Chapter 3 deals with generalization of the recognition problem. Given plenty (infinitely many) of intersection classes, it is inefficient to decide for each class separately what is the complexity of its recognition problem. Our aim is to find subchains (or even more complicated subposets) in the partially ordered set (by inclusion) of intersection classes where no polynomially-recognizable class can be found. This chapter is based on [4, 5].

The third topic considered in this thesis is based on [2]. We investigate the hardness of the recognition problem. Namely we ask whether the absence of large cliques or even of short cycles can simplify the recognition problem. The results on this topic are presented in Chapter 4.

Results in chapters 2, 3 and 4 are mainly original (up to exceptions where we present respective references). Chapter 1 provides the necessary definitions, notation and known facts that are used later in the work. Finally, we provide a brief list of results presented in the rest of thesis in context of already known results.

List of publications relevant to thesis

[1] J. Enright, M. Pergel: On Some Particular Problems on Subtree Overlap Graphs, in preparation.

[2] J. Kratochvíl, M. Pergel: Geometric intersection graphs: Do short cycles help? In G. Lin, editor, COCOON, volume 4598 of Lecture Notes in Computer Science, pages 118 – 128. Springer, 2007.

[3] J. Kratochvíl, M. Pergel: Two Results on Intersection Graphs of Polygons, In G. Liotta, editor, Graph Drawing, volume 2912 of Lecture Notes in Computer Science, pages 59 – 70. Springer, 2003.

[4] J. Kratochvíl, M. Pergel: Intersection graphs of homothetic polygons, In: Proceedings of TGGT 2008, ENDM, pages 265 – 268.

[5] M. Pergel: Recognition of Polygon-circle Graphs and Graphs of Interval Filaments is NP-complete, In A. Brandstädt, D. Kratsch and H. Müller, editors, WG, volume 4769 of Lecture Notes in Computer Science, pages 238 – 247. Springer 2007.

Contents

Chapter 1

Introduction

1.1 Introduction to general graph theory

Graph theory was first introduced by Leonhard Euler to prove a simple result on traversing the bridges of Königsberg, and has many real-world applications. Many types of graphs have been defined, including oriented, directed, simple graphs, multigraphs, hypergraphs, finite or infinite (see [42]). We focus on finite undirected simple graphs.

An ordered pair $G = (V, E)$ is called *a graph* if $E \subseteq \binom{V}{2}$, where V is a set of vertices and E is called a set of edges. Given a graph G, we denote the set of its vertices by $V(G)$ and the set of its edges by $E(G)$. When we are considering only one graph, we omit the argument G and simply refer to V and E as to vertex-set and edge-set, respectively. In that case we also denote by n the number of vertices and by m the number of edges. When referring to a particular edge $\{u, v\}$, we use the notation uv.

A sequence $v_0, e_1, v_1, e_2, \ldots, v_{n-1}, e_n, v_n$ where v_i are vertices and e_i are edges and for all i the edge e_i connects the vertices v_{i-1} and v_i is called a walk. A walk with no repeated edges is called a *tour*. Similarly, a walk without repeated vertices is called a *path*. A walk with no repeated vertices except for $v_0 = v_n$ is called a *cycle*. The length of a cycle or of a path refers to the number of its edges. A graph is called acyclic if it contains no cycle. The length of a shortest cycle in a graph G is referred to as the *girth* of G. If there exists a path from u to v in a given graph G, we say that vertex u is reachable from v in G. We say that a graph is connected if any of its vertices is reachable from any (other) vertex. A graph is called a *tree* if it is connected and acyclic. To denote a complete graph on n vertices, we use the notation K_n. A cycle and a path of length n get denoted by C_n and P_n, respectively.

The maximal connected subgraphs of a given graph are said to be its *connected components*. A graph is called (vertex-)k-connected if it has at least $k+1$ vertices and after removal of any $k-1$ vertices it remains connected. Similarly, a graph is called edge-k-connected if it remains connected even after removal of any (at most) $k-1$ edges.

If it is possible to partition the vertex set of a given graph $G = (V, E)$ into disjoint sets A and B such that each edge has one endvertex in A and one in B, such a graph is called *bipartite*.

For a particular vertex we define its *degree* as number of edges incident with it. Minimum (or maximum) degree refer to the degree of a particular vertex, which is the highest (or the smallest, respectively) and for a graph G we denote them by $\Delta(G)$ or $\delta(G)$, respectively.

A clique of a graph is (some) of its complete subgraphs. Maximum clique is a largest complete subgraph present in a given graph. We denote the size of a maximum clique in a graph G by $\omega(G)$. An independent set in a graph is an empty induced subgraph. The size of a maximum independent set is denoted by $\alpha(G)$. When we assign to each vertex a real number called its weight, by maximum weighted independent set and maximum weighted clique we mean an independent set (or a clique, respectively) which maximizes the sum of the vertices contained in it. The chromatic number of a graph G, denoted by $\chi(G)$, is the minimum number of colors such that we can assign to each vertex of G some colour in such a way that any two neighboring vertices have different colours.

Clearly, $\omega(G) \leq \chi(G)$ holds for any graph G. Equality does not hold in general. A graph is perfect if for any its induced subgraph G' it holds that $\omega(G') = \chi(G')$.

When we refer to an *oriented graph*, the edges are *ordered* pairs.

The *complement* of a graph G is its edge-complement. That is, co-$G = (V, \binom{V}{2} \setminus E)$. For a class $\mathcal{C}$ of graphs we define co-$\mathcal{C} = \{\text{co-}G | G \in \mathcal{C}\}$.

A graph $H = (U, F)$ is called a subgraph of a graph $G = (V, E)$ if $U \subseteq V$ and $F \subseteq E$. If for all $u, v \in U$ such that $uv \in E$ also holds $uv \in F$, subgraph H is called an *induced subgraph*.

We say that a graph H is a *minor* of graph G if H can be obtained by a sequence of edge-contractions of a subgraph of G. Similarly, we say that H is an *induced minor* of a graph G if it can be obtained by a sequence of edge-contractions of an induced subgraph of G. One of the most important theorems of Graph Theory is Robertson-Seymour Theorem (also known as Wagner's conjecture) which states that each minor-closed class can be characterized by finite number of forbidden minors. This does not hold for induced minors.

Graphs whose edges can be transitively oriented, are called *comparability-* or *CO-graphs*. This class of graphs has several applications in theory of intersection graphs. One such application is called *mixing*. For a class $\mathcal{C}$ of graphs a graph G is called *$\mathcal{C}$-mixed* if there exist $H = (V, E) \in \mathcal{C}$ and a comparability graph $I = (V, F)$ such that $G = (V, E \cup F)$, $E \cap F = \emptyset$ and the following holds: For any triple of vertices a, b, c in such a graph if $ab \in E$ and $b \leq c$ in the poset associated with the graph I, then $ac \in E$.

Note that comparability graphs are denoted by capital letters, while complements are denoted by lower case!

1.2 Introduction to the computational complexity

For algorithms many computational models were introduced. For the sake of complexity, algorithms are formalized on Turing Machines. Conversely, particular algorithms are usually designed either for model RAM (Random Access Machine) or in pseudocode.

The time complexity of an algorithm A is the maximum number of elementary operations performed by A computing on inputs of length n.

We do not consider the space complexity of algorithms in this work. Therefore, complexity always refers to the time complexity. We consider only *total* algorithms, *i.e.*, algorithms that halt on every input.

Although complexities of particular algorithms for a given problem may differ for different computational models, we are usually interested only in existence of algorithms whose running time is bounded by some polynomial and note that for all mentioned computation models this set is the same.

An *alphabet* is a finite set of elements (zeroes and ones suffice). Elements of the alphabet are called *letters*. A *word* in a given alphabet is any sequence of letters. A *language* is a set of finite words.

For the sake of simplicity, we may consider a suitable encoding of the input before we subject it to some algorithm. This encoding forms a sequence consisting of zeroes and ones. Thus we may restrict our attention from exploring complexity of a particular problem to the complexity of recognition of a particular formal language. The encoding of the input can be done easily, for further details see [16].

We say that a language L is in class $\mathcal{P}$ if there exist an algorithm A and a polynomial p such that for each x, $A(x)$ decides (correctly) whether $x \in L$ or not, and A has time-complexity at most $p(|x|)$ where $|x|$ refers to the length of x.

Encoding of input instances (for a given particular problem) introduces a bijection between recognition of formal languages and solving decision problems (*i.e.*, problems, whose solution is only 'yes' or 'no'). In further text we often implicitly use this bijection. The first case when we do so is that if there exists a polynomial p such that for all input instances of length n, the running-time of a given algorithm is bounded by $p(n)$, such problem is called *polynomially solvable* and to the time required to do so we refer as to *polynomial time*.

There are several equivalent definitions of the class $\mathcal{NP}$. In combinatorics the most suitable is to consider class $\mathcal{NP}$ as the class of problems for which given an instance of particular problem and a certificate of a solution, we can decide in polynomial time whether the supplied certificate is valid or not. It means, we show that a particular problem is in $\mathcal{NP}$ by showing that we can verify a solution we are provided with for a particular instance in polynomial time.

A problem R is called $\mathcal{NP}$-hard if for all $x \in \mathcal{NP}$ there exists a polynomial reduction (realizable in polynomial time) of x to R. A problem R is called $\mathcal{NP}$-complete if R is $\mathcal{NP}$-hard and $R \in \mathcal{NP}$.

More theory on computational complexity can be found in [16]. Many other classes (mainly between classes $\mathcal{P}$ and $\mathcal{NP}$) are defined. *E.g.*, $\mathcal{ZPP}$, the class of languages recognizable by randomized algorithms that make mistakes with probability 0, but finishes in polynomial time only in the average case. Randomized algorithms enter this thesis only sporadically when we refer to third-party results.

At the end of this section we present $\mathcal{NP}$-hard problems that get later used in our thesis:

1. E3-SAT:
   Instance: A 3-formula ϕ in conjunctive normal form (*i.e.*, $\bigvee_{i=1}^{n} \bigwedge_{j=1}^{3} l_{ij}$, where l_{ij} is a literal, *i.e.*, either a variable or its negation).
   Problem: Is formula ϕ satisfiable (*i.e.*, does there exist an assignment of variables x such that $\phi(x)$ is true)?

2. E3-NAE-SAT:
   Instance: A 3-formula ϕ in conjunctive normal form.
   Problem: Does there exist an assignment x such that in each clause of formula ϕ at least one literal is assigned true and at least one false?

3. PURE-E3-NAE-SAT:
   Restriction of E3-NAE-SAT to input with all literals positive (*i.e.*, there is no negation).

4. E3-NAE-SAT(4):
   The same as E3-NAE-SAT, except each variable occurs at most four times.

5. P3CON-E3-SAT(4):
   Instance: A formula ϕ in conjunctive normal form such that a bipartite graph G_ϕ having in one partition variables and in the other clauses and an edge va says that a vertex v is in clause a is planar and 3-connected. Moreover, each clause of ϕ consists of exactly 3 literals and each variable occurs at most 4 times.
   Problem: Does there exist satisfying assignment for ϕ?

6. DISTINCT-3-COL:
   Instance: A bipartite graph G with partitions A and B. All vertices in partition A have degree 3.
   Problem: Can we color vertices of B to get for each $v \in A$ its three neighbors colored by all three (distinct) colors?

7. K-CON-K-COL:
   Instance: A K-connected graph G.
   Problem: Is it possible to assign K colors to individual vertices of G in such a way that no adjacent pair has the same color?

Problem 6, in fact, is exactly edge-coloring explored in [28]. Other problems can be found, *e.g.*, in [16]. Note that E3-SAT is sometimes called just 3-SAT, but we prefer emphasizing the fact that we expect always all three literals in each clause, not at most three.

1.3 Introduction to the Intersection graphs

For a set system $\mathcal{S} = (S_1, ...S_n)$ we define an appropriate intersection graph $G = (\{V_1, ...V_n\}, E)$ where $V_iV_j \in E$ whenever $S_i \cap S_j \neq \emptyset$. A graph is called an intersection graph if it has an intersection representation.

It is a well known fact that every graph is an intersection graph of a suitable set-system. Proof of this fact can be found *e.g.*, in [43].

Therefore we are interested in particular classes of intersection graphs representable usually by arc-connected objects in a plane. For class of objects $\mathcal{M}$ we define class of intersection graphs of $\mathcal{M}$ (usually denoted $IG(\mathcal{M})$)

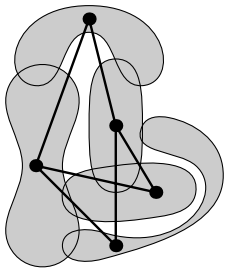


Figure 1.1: Example of a graph with an intersection representation by arc-connected sets in the plane.

Figure 1.2: Example of an INT-representation.

as class of graphs having intersection representation by objects of $\mathcal{M}$. For technical purposes, we consider two objects always to properly intersect (not just touch). Note that this is without loss of generality and whenever we use notion of touching sets, we can by perturbation argument modify it into a representation without touching.

Intersection graphs of geometric objects, namely in the plane, are intensively studied both for their practical motivations and for interesting structural and algorithmic properties. Many hard (NP-complete in general) optimization problems become polynomially solvable when restricted to various classes of intersection graphs. These classes of graphs have motivation *e.g.*, in biology, history or VLSI-circuit design. Intersection-defined classes were introduced both to recognize whether a particular graph has some representation and to provide more efficient algorithms for graph-optimization problems. In both cases we are usually interested in a corresponding intersection representation, which motivates the *recognition problem*, *i.e.*, to determine whether a given graph belongs to a particular class.

As we will be interested mainly in intersection graphs of polygons, subtrees, intervals and filaments, throughout the thesis we use small letters as $a, b, u, v, \ldots$ for vertices of the graph under consideration, given a representation by polygons, P_v denotes the polygon representing vertex v. However, in figures, to avoid multiple subscripts, we will usually omit the symbol P. Similarly, for representation by filaments (curves above a given object), we use symbol F_v for the filament representing vertex v and I_v for its underlying interval. In a similar way we use S_v to denote subtree representing vertex v.

Probably the oldest and simplest of these are *interval graphs,* we also use to denote as *INT-graphs,* intersection graphs of intervals on a line [21],

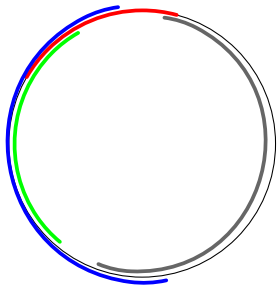


Figure 1.3: Example of a CA-representation.

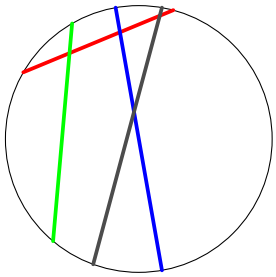


Figure 1.4: Example of a CIR-representation.

whose structure is well understood, they are recognizable in linear time, and for which problems like clique, independent set, dominating set, chromatic number and many more are tractable. It is also interesting to say that INT-graphs are perfect.

Intersection classes yield many inclusions. Many classes contain class of interval graphs, while on the other end of the spectrum are *STRING graphs,* intersection graphs of arc-connected sets in the plane [8, 35, 36], which are hard to recognize [33], and whose recognition was only recently shown decidable [44, 49] and then even more surprisingly in NP [48]. Note that following classes usually contain class of interval graphs and are contained in class of STRING-graphs.

An obvious generalization of class INT is class called *circular-arc* or *CA-graphs* [18]. This class is defined as intersection graphs of arcs of a circle. If we consider an interval representation of a particular graph, we may assume all intervals are finite. When we cut ends of the underlying line to get a segment instead of a line, whose endpoints we grab and stick together, we obtain a CA-representation of the same graph. CA-graphs can be also recognized in polynomial time [53, 13] and these graphs can be colored in polynomial time by fixed number of colors [17].

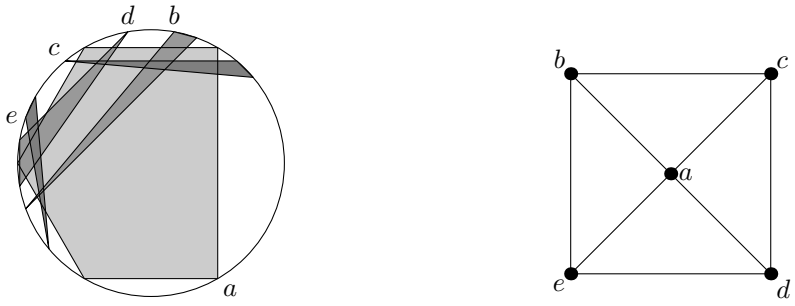


Figure 1.5: PC-representation and an appropriate PC-graph

If we take straight line segments (chords) inscribed into a circle, we get class of circle graphs, also called *CIR*. Also circle-graphs can be recognized in polynomial time [3, 6, 50]. This class seems very similar to class of CA-graphs, but there is a lot of differences. Main difference is that circle graphs cannot be colored in polynomial time even by k colors, where $k > 3$ is fixed [54].

Overlap graphs are defined similarly as intersection graphs. Again, given a set system, each set corresponds to a vertex, but an edge corresponds only to such a pair of vertices, that not only their corresponding sets have nonempty intersection, but also both their symmetric difference are nonempty. Similarly to class INT, we may define class of interval-overlap graphs (IO) and it is well known that interval-overlap graphs are exactly circle graphs.

A natural common generalization of circle and circular-arc graphs is a class of *polygon-circle graphs*. This class was first suggested by M. Fellows [in personal communication with applicant's advisor] in 1988, when it was pointed out that this class of graphs is closed under taking induced minors. Under the name of *spider graphs*, polygon-circle graphs appeared in [31], where several claims that could help to design polynomial recognition algorithm were presented. Later the recognition problem for PC-graphs was posed, *e.g.*, by Spinrad [52].

Note that as circle graphs are not perfect, a structural property of graph classes which many intersection graphs possess is *near-perfectness* in the sense of Gyarfás [24]. A graph class is near-perfect if the chromatic number of each of its graphs is bounded by a function of the clique number of the graph. (For perfect graphs this function is identity.) Polygon-circle graphs are near-perfect as shown in [32].

The class of PC-graphs was generalized by Gavril into the class of *interval-filament graphs (IFA-graphs)* which are defined as the intersection graphs of filaments above a given line in a plane. Filaments are curves with endpoints on the given line. The endpoints of each filament define an interval on the

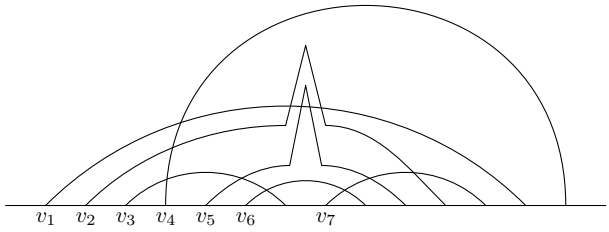


Figure 1.6: Representation by interval-filaments.

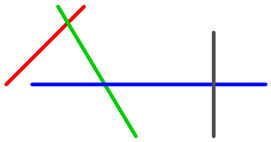


Figure 1.7: SEG-representation.

line. Filaments belonging to disjoint intervals *must not intersect each other* and filaments over overlapping invervals *must* intersect each other. IFA-graphs were defined by Gavril [20] and characterized by the mixed property as complements of (co-INT)-mixed graphs. They were equivalently described as caterpillar-overlap graphs [23]. Descriptions by the mixed property, as well as subcaterpilar-overlap description shows testability of membership in class IFA in polynomial time. Despite the structural characterization, the recognition has been open since. IFA-graphs contain PC-graphs and therefore Gavril's polynomial algorithm for the problem of the maximum weighted clique and maximum weighted independent set [20] immediately yield polynomial algorithms for PC-graphs and also for CIR-, CA-graphs.

In terms of filaments, we may similarly define class of *circular filament graphs* (shortly CFA) and *subtree-filament graphs (SFA).* Again, filaments above disjoint structures (circular arcs or subtrees of a tree, respectively) *must not* intersect each other and, again, filaments above overlapping structures *must* mutually intersect. Also these two classes were described by the mixed-property [20]. Recently, [11] it was shown that subtree filament graphs are exactly subtree overlap graphs (shortly SOG, *i.e.*, overlap graphs of subtrees of a tree).

In the thesis, we are also interested in *SEG or segment graphs,* which are intersection graphs of segments in a plane, *CONV or convex graphs, i.e.*, intersection graphs of polygons in a plane and *PDISK or pseudo-disk graphs,* intersection graphs of pseudo-disks[1] in a plane.

[1]pseudo-disks are sets in a plane having at most two common points on a boundary

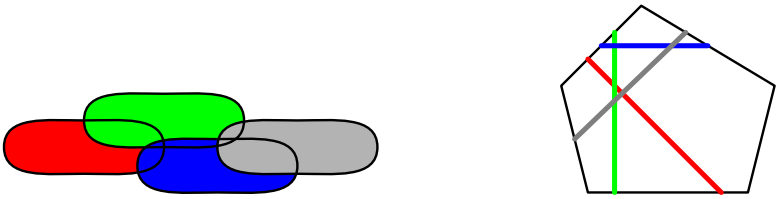


Figure 1.8: On the left, PDISK-representation, on the right representation of a 5-polygon-graph.

Given a representation R of G by polygons (not necessarily convex, not necessarily inscribed into a circle) with $G \subseteq H$, we say that R *can be extended* into a representation S of H if S is obtained from R by adding new corners to existing polygons and by adding new polygons representing the vertices of $V(H) \setminus V(G)$.

Intersection graphs of *homothetic convex polygons* in a plane are defined as follows: Objects in a plane are called homothetic if one can be obtained by shifting and scaling of another. If all objects used for representation of a particular graph are homothetic convex polygons, that graph is an intersection graph of homothetic convex polygons.

Another important class of intersection graphs is the class of *permutation graphs,* for shortness *PER-graphs* [14]. Given a permutation $\pi \in S_n$, the appropriate permutation graph has vertices $v_1, \ldots, v_n$ and an edge $v_i v_j$ is present if $i > j$ and $\pi(i) < \pi(j)$. When we consider a permutation π written below an identity permutation, the intersection representation is obtained by connecting each number in the identity-line with the same number in the permuted line. This class can be generalized in such a way that instead of one permutation we consider k permutations, begin with the identity-permutation and establish a wire connecting all occurrences of one number (by piecewise linear segments). We denote such a class (for a fixed k) as PER_k. When we permit number k (of permutations) to be arbitrary, we obtain the class of FUN-graphs, by full name the function-graphs, intersection graphs of continuous real functions on interval $< 0, 1 >$.

It is important to note that FUN-graphs are exactly co-CO-graphs [22] and that while PER-graphs as well as FUN-graphs can be recognized in polynomial time, for a fixed $k > 2$ it is NP-hard to recognize whether a given graph is a PER_k-graph.

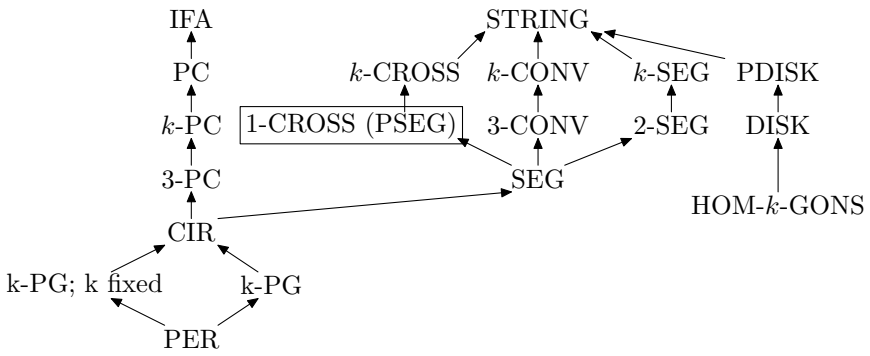


Figure 1.9: Illustration of inclusions among intersection-defined classes. Note that also for each k it is k-PC $\subseteq$ k-CONV and that IFA $\subseteq$ STRING.

Now, please, note that we have defined infinitely many classes that contain class PER and simultaneously they are contained in class FUN. In a similar way we define also infinitely many classes even between other classes:

Between classes of CIR-graphs and PC-graphs, infinitely many classes of k-PC-graphs (intersection graphs of convex k-gons inscribed into a circle) can be sandwiched. Note the difference from *polygon-graphs* introduced in [9] as k-polygon-graphs are graphs representable as intersection graphs of straight line segments inscribed into a k-gon which are sandwiched between PER- and CIR-graphs. Similarly we can generalize the class of SEG-graphs into STRING-graphs in three completely different ways. First of them proceeds in terms of k-CONV-graphs and generalizes class of SEG-graphs into class of CONV-graphs exactly in the same way as when we are generalizing CIR-graphs into PC-graphs by classes of k-PC-graphs. Second generalization is in terms of k-STRING-graphs. We define k-STRING as a string consisting of (at most) k piecewise linear segments. The third way is topological and defines class of pseudo-segments (PSEG) [4] also known as 1-CROSS graphs. 1-CROSS graphs are graphs of curves in a plane such that any pair of segments intersects at most once. Analogously, we define k-CROSS graphs.

Also chordal graphs (CHOR) can be described as intersection graphs. Primarily, they are defined as class of graphs without chord-less cycle of length larger than three or by the existence of perfect elimination scheme [47]. They are also the intersection graphs of subtrees of a tree. They are known to be perfect and it is known that the maximum clique as well as chromatic number (and even coloring) can be found in polynomial time. Obviously, they contain class of interval graphs. A more sophisticated proof is necessary to show that they form a subclass of PC-graphs.

The proof of inclusion of CHOR-graphs in class of PC-graphs is shown through another class: Cactus is a graph defined recursively: Arbitrary cycle C_k is a cactus. A larger cactus is obtained from already existing cactus by picking one its vertex v, adding a new cycle C_l whose (exactly) one vertex is unified with v. Class CACT is a class of cactus-subtrees. Obviously CHOR $\subseteq$ CACT. And if we consider CACT-representation of a graph G, we may consider all cycles of underlying cactus to be drawn outside all other cycles, *i.e.*, boundary of the cactus contains all vertices (of the cactus). If we split each vertex attaching one cycle to the other into a chord, original cactus changes into a (geometrical) circle and if we take convex hull of vertices belonging to individual subtrees, we obtain a PC-representation, which proves that CACT $\subseteq$ PC (and therefore also CHOR $\subseteq$ PC).

On the recognition problem many results were obtained. Among them, at the moment, let us recall following:

- It is NP-hard to determine whether a given graph is a PDISK-graph [34].
- The recognition problem of the PDISK-graphs without K_3 is polynomially solvable [34].
- It is NP-hard to recognize whether a given graph is a STRING-graph (and remains so even for graphs with girth 4) [34].
- It is NP-hard to determine whether a given graph is a SEG- or a STRING-graph. Moreover, no polynomially recognizable class can be sandwiched between these two classes [33].
- Classes PER and FUN can be recognized in polynomial time [14, 22].
- For any $k > 2$, it is NP-complete to recognize PER$_k$-graphs [55].

These theorems motivated our research whose results are presented in following chapters.

1.4 Overview of new results

In Chapter 2 we explore how difficult is it to get some representation of reasonable size (*i.e.*, the smallest possible in some sense) and what size can be enforced by some graphs. We show that it is NP-complete to find an optimal representation for PC-graphs and that it is even hard to decide whether a particular graph can be represented by k-gons inscribed into a circle or not. As well, we show that for graphs on n vertices, there exist

PC-graphs requiring polygons with at least $n - \log n + o(\log n)$ corners for its representation and that this bound is tight (*i.e.*, any PC-graph on n vertices can be represented by polygons with $n - \log n + o(\log n)$ corners). Then we show that it is impossible to use Cartesian coordinate-system to get polynomial certificate proving whether a given graph is a CONV-graphs and the chapter gets concluded by hardness of recognition for some subclasses of subtree-overlap graphs (as the recognition problem for the whole class of subtree-overlap graphs remains still open).

As we have outlined, a lot of intersection-defined classes are defined and being explored and some of them form non-collapsing infinite sequences. Therefore for the recognition problem it is not sufficient to show that one particular class can be (or cannot be) recognized (under particular assumptions), but it is necessary to ask, between which classes we can or cannot find some efficiently recognizable class. First result of this type is due to Kratochvíl in [33] for classes of SEG- and STRING-graphs. This reduction produces either a SEG-graph or not even a STRING-graph. To this problem we devote Chapter 3 where we show that for any fixed convex polygon S no polynomially recognizable class can be sandwiched between the classes of intersection graphs of homothetic convex polygons S and pseudodisk graphs and also that no polynomially recognizable class can be sandwiched between the class of PC-graphs and IFA-graphs.

The fact that the recognition problem becomes polynomial for PDISK graphs when restricted to graphs with girth at least 4 yield a question whether it is possible to recognize efficiently intersection classes when we require graphs to have large girth. So in Chapter 4 we focus on the recognition problem for intersection-defined graphs with large girth. There we show that PC-graphs can be polynomially recognized when restricted to graphs with girth at least 5. Conversely, we show that the recognition problem remains NP-hard for SEG-graphs even for graphs with arbitrarily large girth. The last theorem combines question of this chapter with problem studied in previous chapter, as it also shows that for any fixed k, when restricted to graphs with girth at least k, no polynomially recognizable class can be sandwiched between class of SEG-graphs and PSEG-graphs.

Chapter 2

Complexities of Representations

As mentioned in the introduction, it is easy to decide whether a given graph is a PER-graph or not. The same holds for FUN-graphs. Since algorithms operate on the intersection representation rather than on the graph, we want to find an optimal (in some sense) representation in polynomial time. The size of the representation is measured in several ways. For connected substructures in a tree, the size may be measured in number of leaves of the underlying trees or in maximum number of leaves of individual subtrees. For SEG- and CONV-graphs, the size of a given representation can be measured in size of numbers needed to describe individual endpoints, size of STRING-representation is usually considered as the number of intersections in the representation. Size of PC-representation gets measured in terms of *complicacy*, which gets defined later in this section and for a FUN-representation we consider as its size number of permutations forming a realizer of a corresponding poset, it means, minimum k such that G is a PER_k-graph.

For some classes, the optimum representation can be found in polynomial time, e.g., for INT-graphs this problem is trivial and therefore solvable in linear time by any recognition algorithm (when we consider INT as a subclass of CHOR-graphs).

Example of the converse are FUN-graphs. Despite the fact that FUN-graphs (on n vertices) can be recognized in time $O(n^3)$ through transitive orientation of their complement, it is NP-hard to find their optimum representation [55] (and it remains hard even to get a reasonable approximation if NP $\not\subseteq$ ZPP [25]). Conversely, given a transitive orientation, we can easily get its FUN-representation of size $O(n^2)$ by finding n permutations forming a realizer [41]. More sophisticated constructions can improve multiplicative

constant of the size [26] and, moreover, for all functions $f(n) \in o(n)$ there are posets requiring size of representation $\omega(nf(n))$ [30, 41].

2.1 PC-graphs and Complicacy

In this section, based on [39] (which extends [45]), we pay closer attention to the question of how complicated should be the polygons representing the vertices of a polygon-circle graph with n vertices. One can easily see that n-gons always suffice, which means that polygon-circle graph recognition is definitely in NP. It is conceivable, however, that polygons with less corners would suffice. To be able to precisely formulate this question, we define the *complicacy* of a graph G as the minimum k such that G is the intersection graph of convex k-gons inscribed to a circle, and we denote this invariant by $cmp(G)$ (we set $cmp(G) = \infty$ if G is not a polygon-circle graph). We further define $cmp(n)$ to be the maximum of $cmp(G)$ over all polygon-circle graphs with n vertices. The main result in this direction is the following (here and throughout the section, all logarithms are base 2):

Theorem 1 *We have* $cmp(n) = n - \log n + o(\log n)$.

The lower and upper bounds are proved separately. At the end of this section we consider the computational complexity of determining the complicacy of a graph with the following result:

Theorem 2 *For every fixed finite* $k \geq 3$, *it is NP-complete to decide if* $cmp(G) \leq k$ *holds for an input graph* G.

This result answers an open problem listed at J. Spinrad's web page [51]. The fact that recognition of k-PC-graphs is in NP follows directly from the proof of Lemma 24 and from Theorem 1. In this chapter we focus on the hardness part. Note also that for $k = 2$, $cmp(G) \leq 2$ if and only if G is a circle graph, a polynomially decidable question.

If P is a polygon, then the connected parts obtained from the bounding circle by deleting the corners of P are referred to as the *P-segments*. If two polygons represent nonadjacent vertices, they must be disjoint and hence all corners of one of them lie within the same segment determined by the other one, and vice versa. In the following technical definition we assume that all polygons under consideration are disjoint.

Definition 3 *We say that polygon P* blocks *polygon Q from polygon S if the corners of Q lie in a different P-segment than the corners of S. If a set $\mathcal{S}$ of polygons is such that none of them blocks any other two polygons from each other, we say that the polygons are* positioned around the circle.

See Figure 2.1 for an illustrative example of blocking polygons, and a set of polygons positioned around the circle. Next we make the first simple but useful observation.

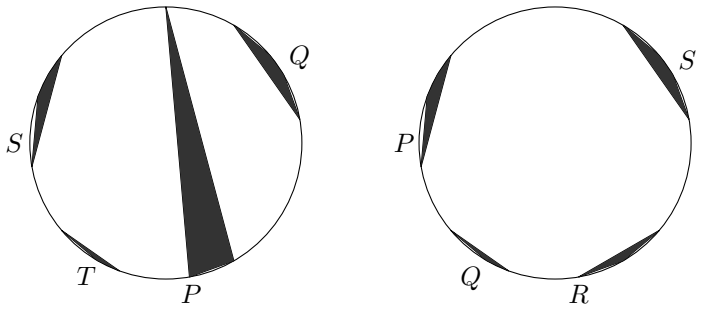


Figure 2.1: In the left, polygon P blocks polygon Q from polygons S and T, in the right, all four polygons are positioned around the circle.

Proposition 4 *In any representation R of the cycle C_{2k} with $2k$ vertices $u_1, u_2, \ldots, u_{2k}$, the polygons $R_{u_{2i}} : i = 1, 2, \ldots, k$ are positioned around the circle. If W'_k is the graph obtained from C_{2k} by adding a vertex v adjacent to $u_{2i}, i = 1, 2, \ldots, k$ (*i.e.*, W'_k is the wheel W_k with each rim edge subdivided), then R_v has at least k corners and $cmp(W'_k) \geq k$ (in fact, the complicacy of W'_k equals k).*

2.1.1 Complicacy of representations – the lower bound

Theorem 5 *For n large enough, we have $cmp(n) \geq n - \log n - 2 \log \log n$.*

Proof. The proof is by constructing graphs of large complicacy by an explicit construction. Suppose n is large enough (how large will follow from the calculations in the proof). Let ℓ be the uniquely defined integer such that

$$1 + 2\ell + \binom{2\ell}{\ell} < n \leq 1 + 2(\ell + 1) + \binom{2\ell + 2}{\ell + 1}.$$

We construct the graph $G = (V, E)$ with vertex set

$$V = \{v\} \cup L \cup P,$$

where vertex v is adjacent to all other vertices, L is a clique of size $2\ell + 2$, and P is an independent set whose vertices are indexed by distinct $(\ell + 1)$-element subsets of L. These indices will determine the adjacencies between

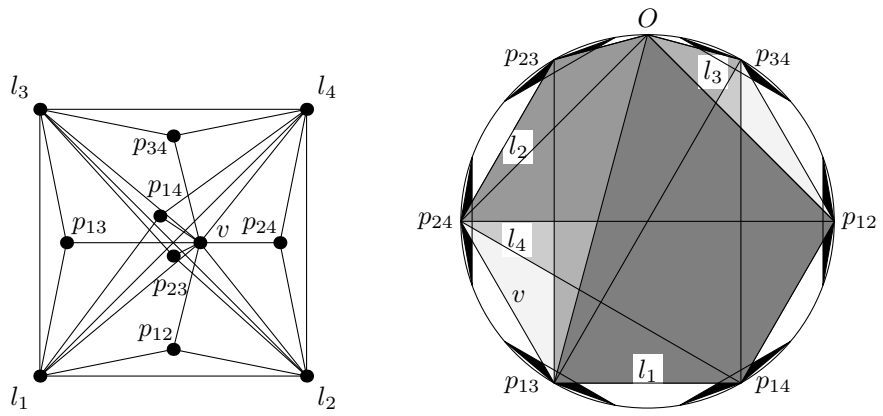


Figure 2.2: An example of the lower bound construction for $\ell = 2$ and $n = 11$. We write simply p_{13} for $p_{\{1,3\}}$ etc.

vertices of P and L as follows. If $p_a \in P$ (with $a \subset L$, $|a| = \ell + 1$), we make p_a adjacent to all $x \in a$ and to no other vertices of L. (Note that G is not determined uniquely, it depends on the choice of the sets used for indexing the vertices from P. Any choice of distinct indices works.)

We claim that G is a polygon-circle graph and that $cmp(G) \geq |P|$. For the first part of the claim, choose a point O on the circle and position the polygons corresponding to the vertices of P around the circle (so that they do not block O from one another). Choose one corner of each of them as its reference point and represent each vertex x of L by the convex hull of the reference points of the vertices of P adjacent to x and of O. By making O a corner of each R_x we guarantee that every two polygons $R_u, R_w, u, w \in L$ intersect (L is a clique). Finally R_v will be the convex hull of O and the reference points of all polygons representing vertices of P. (Note that the auxiliary point O may not be necessary, *e.g.*, in the case when for every two vertices in L, P contains a vertex adjacent to both of them.)

To argue that $cmp(G)$ is large, consider an optimal representation R of G. Note first that for $n > 5$ we have $\ell \geq 1$ and hence $\binom{2\ell}{\ell} \geq 2$. This means that $n > 2\ell + 3$ and indeed G contains all vertices of L.

The key observation is that the polygons $R_p, p \in P$ must be positioned around the circle. For suppose this is not the case, say R_{p_b} blocks R_{p_a} from R_{p_c} for some $a, b, c \subset L$. Since these subsets are different but of equal size, there must be an $x \in a \setminus b$ and a $y \in c \setminus b$. By the definition of G, R_x intersects R_a and R_y intersects R_c, but none of R_x, R_y intersects R_b. But that means that R_b blocks R_x from R_y and these two polygons cannot intersect each other (though x and y belong to the clique L).

To intersect all polygons representing vertices of P, R_v must have at least $|P|$ corners, and hence $cmp(G) \geq |P| = n - 2\ell - 3$. The rest is a simple calculation.

Assume for contradiction that $\ell > \frac{\log n}{2} + \log\log n - \frac{3}{2}$. Then

$$n > \binom{2\ell}{\ell} > \frac{2^{2\ell}}{2\ell+1} > \frac{n\log^2 n}{8(\log n + 2\log\log n - 2)} > n$$

for large enough n (since $\lim_{n\to\infty} \frac{\log^2 n}{8(\log n+2\log\log n-2)} = \infty$), a contradiction. Therefore (for every large enough n), $\ell \leq \frac{\log n}{2} + \log\log n - \frac{3}{2}$ and

$$cmp(G) \geq |P| = n - 2\ell - 3 \geq n - \log n - 2\log\log n$$

as claimed. ♠

2.1.2 Complicacy of representations – the upper bound

Theorem 6 *For every positive constant $c < 1$, there exists an n_0 such that $cmp(n) \leq n - c\log n$ for every $n > n_0$.*

Proof. Let $G = (V, E)$ be a graph on n vertices and let $cmp(G) = k \geq 4$. Consider a polygon-circle representation R of G such that no two polygons share a corner and such that every polygon has at most k corners (this can be always achieved by splitting the corners). Let our representation have the minimum total number of corners among all such representations. Choose a vertex $v \in V$ such that R_v has k corners, and denote its corners $v^1, v^2, \ldots, v^k$ as they appear clockwise around the circle.

Based on this representation R, define A to be the set of vertices x such that R_x has all corners within two consecutive R_v-segments, *i.e.*, such that R_x intersects R_v only in the triangle $v^{i-1}v^iv^{i+1}$, for some i. For $x \in A$, denote this i by $j(x)$ and call it the *index* of x. We can see that for every i, there exists an $x \in A$ of index i. For if such a vertex did not exist for some i, we could reduce the number of corners in the representation by deleting the triangle $v^{i-1}v^iv^{i+1}$ from R_v, contradicting the choice of R.

Now assume that R' minimizes $\sum_{x\in A} j(x)$ among all representations with the same central polygon R_v, with the same total number of corners and with the same set A (when defined as above) as R. For the sake of simplicity we call R' again just R.

For every i, choose one vertex as a representative of the vertices of index i, and call it a_i. As argued above, a_i is well defined for every i. The intersection graph of the polygons $R_{a_i}, i = 1, 2, \ldots, k$ is either a cycle or a disjoint union

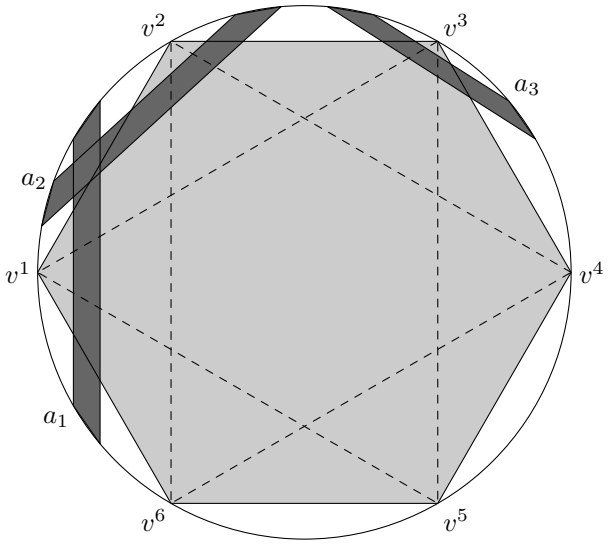


Figure 2.3: Illustration to the choice of representatives of index classes.

of paths, since a_i can only intersect a_{i-1} and/or a_{i+1}. We denote A_1 the set of those a_i that have no neighbors among the other a_h's, and $A_2 = \{a_1, a_2, \ldots, a_k\} \setminus A_1$. We then denote $B = (V \setminus \{v\}) \setminus (A_1 \cup A_2)$ and B_1 the set those vertices of B which are adjacent to at least one vertex of A_1.

Next we claim that every two vertices of A_1 have different sets of neighbors. For suppose that $\{x|xa_i \in E\} = \{x|xa_j \in E\}$ for some $j < i$. Then the polygons representing the neighbors of a_i intersect one side of R_{a_j}, and hence both R_{a_i} and R_{a_j} could be replaced by two parallel chords (digons) placed close enough to this side. This would result in a new representation of the same graph, with the same position of R_v and the same set A, but with a strictly smaller sum of the indices of the vertices of A, contradicting the choice of R. See an illustrative example in Figure 2.4.

The immediate but very important consequence is that

$$|A_1| \leq 2^{|B_1|},$$

since v is a common neighbor of all vertices in A_1, and on the other hand no vertex of A_1 is adjacent to any other vertex of $A_1 \cup A_2$.

Finally we attend to vertices of A_2. Consider the example in Figure 2.5. If R_{a_1} and R_{a_2} intersect, they must intersect in the area bounded by the v^1v^2 side of R_v and the corresponding R_v-segment. Choose a point v^{12} in this segment between the leftmost corner of a_2 and the rightmost corner of a_1. If

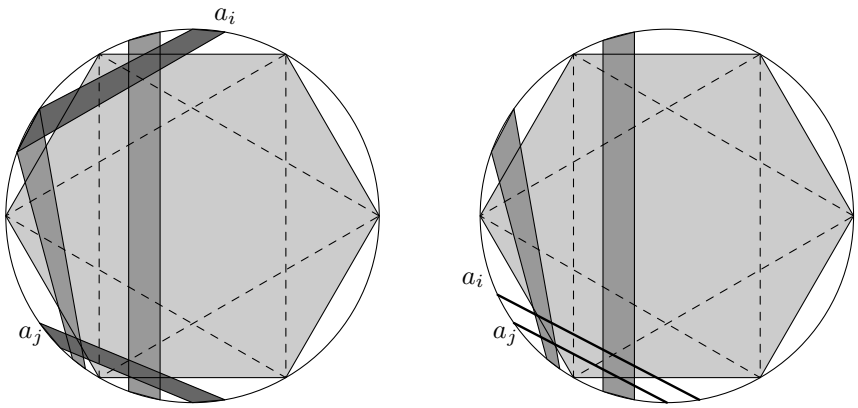


Figure 2.4: Illustration to reducing the sum of indices of the A vertices.

we now replaced the central polygon R_v by the convex hull of $v^{12}, v^3, \ldots, v^k$ (*i.e.*, the corners v^1 and v^2 are replaced by v^{12}), the new R_v (denoted by dashed lines in the figure) would still intersect both R_{a_1} and R_{a_2}, but the total number of corners would decrease. Since this is impossible by the choice of R, the new collection of polygons does not represent G anymore. There are only two possible reasons. Either the new R_v intersects a polygon R_b which is not supposed to be intersected (as in Figure 2.5 bottom left), or the new R_v loses intersection with some R_b that it previously crossed (as in Figure 2.5 bottom right). In the former case, R_b would lie fully within the segment v^1v^2, in the latter one, $b \in A$ and its index is $j(b) = 1$ (or, symmetrically, $j(b) = 2$). In any case, we choose one such b and call it a witness for $(1, 2)$. Similarly, we choose a witness for every $(i, i+1)$ such that $R_{a_i} \cap R_{a_{i+1}} \neq \emptyset$, and denote by B_2 the set of witnesses constructed in this way. By the construction, $B_2 \subset B$. It may happen, though, that one vertex of B_2 is a witness for two pairs (a witness of the latter type and index i could have been chosen both for the pair $(i-1, i)$ and for $(i, i+1)$). Given the fact that a path of t consecutive vertices from A_2 would involve $t-1$ pairs, such a path would give rise to at least $\frac{t-1}{2}$ distinct witnesses, and we obtain the following inequality:

$$|A_2| \leq 3|B_2|.$$

(Note here, that B_1 and B_2 are not necessarily disjoint.)

Now we summarize

$$k = cmp(G) = |A_1| + |A_2| \leq 2^{|B_1|} + 3|B_2| \leq 2^{n-k} + 3(n-k)$$

(the last inequality follows from the fact that $A_1 \cup A_2 \cup B_1 \subset V$ and $A_1 \cup A_2 \cup B_2 \subset V$, while in each case the three summands are pairwise disjoint),

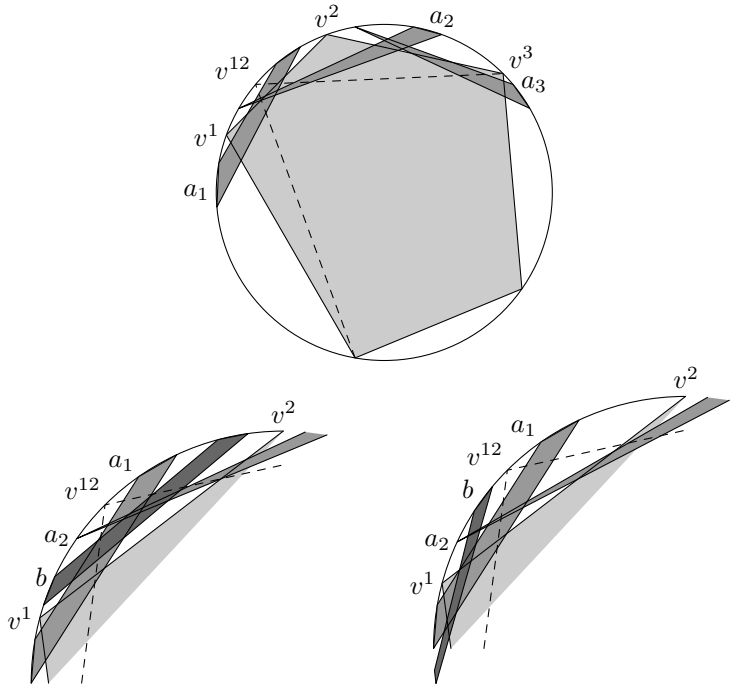


Figure 2.5: Illustration to the construction of B_2, the blocking set for A_2.

and hence

$$2^n \geq 2^k(4k - 3n).$$

For $k > n - c\log n$ $(0 < c < 1)$, we would get

$$2^n > 2^{n-c\log n}(4(n - c\log n) - 3n) = 2^n \frac{n - 4c\log n}{n^c} > 2^n$$

for every $n > n_0$ for some n_0, as $\lim_{n\to\infty} \frac{n}{n^c} = \infty$ and $\lim_{n\to\infty} \frac{4c\log n}{n^c} = 0$. Which is the final long worked for contradiction. ♠

The proof of Theorem 1 is now at hand. Consider the difference $f(n) = cmp(n) - n + \log n$. On one hand, Theorem 5 states that $f(n) \geq -2\log\log n = o(\log n)$. On the other hand, Theorem 6 yields that for every $\varepsilon > 0$, $f(n) < \varepsilon\log n$ for every large enough n (by setting $c = 1 - \varepsilon$), and hence $f(n) = o(\log n)$.

2.1.3 Computational complexity

We first restate Theorem 2 in a slightly stronger form:

Theorem 7 *For every fixed $k \geq 3$, it is NP-complete to decide if $cmp(G) \leq k$ for an input graph G, even if G is promised to have complicacy at most $k+1$.*

Proof. We reduce from the Distinct-3-COL, *i.e.*, following hypergraph coloring problem: Given a set $\mathcal{T}$ of triples over a base set X, decide if the elements of X can be colored by three colors so that every triple $T \in \mathcal{T}$ receives all three colors.

Given such a $\mathcal{T}$, we construct a graph $G_{\mathcal{T}} = (V, E)$ as follows. The vertex set will consist of a cycle of length $2k$ on vertices $u_1, u_2, \ldots, u_{2k}$, a pair of vertices a_x, b_x for every $x \in X$, and a vertex c_T for each triple $T \in \mathcal{T}$. The c vertices form a clique and each of them is adjacent to all u vertices with even subscripts, to all b vertices, and to those a vertices which correspond to x's belonging to the particular triple. The b vertices are further adjacent to their corresponding a vertices and to u_2, u_4, u_6. Formally

$$V = \{u_i | i = 1, 2, \ldots, 2k\} \cup \{a_x, b_x | x \in X\} \cup \{c_T | T \in \mathcal{T}\}$$

$$E = \{u_1u_2, u_2u_3, \ldots, u_{2k}u_1\} \cup \{a_xb_x | x \in X\} \cup \{b_xu_i | x \in X, i = 2, 4, 6\}$$

$$\cup \{b_xb_y | x, y \in X\} \cup \{c_Ta_x | x \in T \in \mathcal{T}\} \cup \{c_Tb_x | T \in \mathcal{T}, x \in X\}$$

$$\cup \{c_Tc_S | T, S \in \mathcal{T}\} \cup \{c_Tu_i | T \in \mathcal{T}, i = 2, 4, 6, \ldots, 2k\}.$$

We claim that $G_{\mathcal{T}}$ is always a PC graph of complicacy at most $k+1$, and its complicacy is (at most) k if and only if $\mathcal{T}$ allows a coloring as desired.

Let R be a PC representation of $G_{\mathcal{T}}$. We first note that all polygons $R_{a_x}, x \in X$ must be positioned around the circle. For if one of them would block another two from each other, some R_x would block some R_y from R_{u_2} and the auxiliary polygon R_{b_y} would not be able to intersect both R_y and R_{u_2}.

As observed in Proposition 4, the polygons $R_{u_{2i}}, i = 1, 2, \ldots, k$ are positioned around the circle, and only these vertices of the cycle are intersected by R_{c_T}'s. Since the polygons representing the b's intersect (only) R_{u_2}, R_{u_4} and R_{u_6}, all the polygons representing the a vertices must lie in the segments determined by R_{u_2}, R_{u_4} and R_{u_6}. So color an element $x \in X$ by color i if R_{a_x} is blocked by R_{u_i} from the other R_{u_j}, $i, j = 2, 4, 6$. We claim that this is a good coloring. For if a triple $T \in \mathcal{T}$ contained two vertices of the same color, say x and y of color 2, the polygon R_{c_T} would need $k-1$ corners for

intersections with $R_{u_i}, i = 4, 6, \ldots, 2k$, plus at least two more corners for intersections with R_x and R_y, that is at least $k + 1$ corners altogether.

On the other hand, given a feasible 3-coloring of X by colors 2,4,6, a representation by k-gons can be achieved by placing the polygons $R_{a_x}, x \in X$ around the circle such that polygons corresponding to vertices of color i are placed within the segment determined by the chord $R_{u_i}, i = 2, 4, 6$. Each c vertex can then be represented by a k-gon with one corner in each segment determined by $R_{u_i}, i = 2, 4, 6, \ldots, 2k$. An illustrative example is in Figure 2.6, where a triangle representing a triple $T = \{x, y, z\}$ is marked. For the sake of simplicity, we only illustrate the case $k = 3$, and the auxiliary polygons R_{b_x} are not shown (drawn with invisible ink).

Along the same lines it is seen that $cmp(G_{\mathcal{T}}) \leq k + 1$ if $(X, \mathcal{T})$ allows a 2-coloring without monochromatic triples. ♠

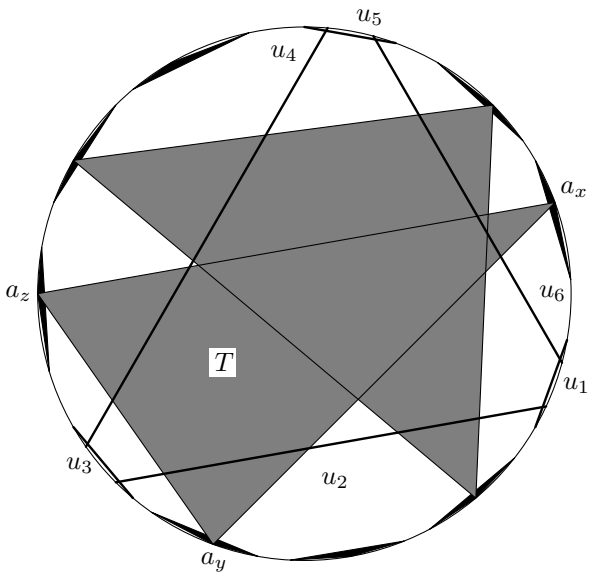


Figure 2.6: Illustration to the NP-completeness proof.

One may ask for what function $f(n)$ of n it is still hard to decide if $cmp(G) \leq f(n)$ for an input polygon-circle graph G on n vertices. As we have seen in previous sections, $f(n)$ must be somewhat smaller than n so that we might expect NP-hardness. By a closer examination of the proof of Theorem 2 we see that we can get close to $\frac{n}{2}$:

Theorem 8 *For every rational $c < \frac{1}{2}$, it is NP-complete to decide if $cmp(G) \leq cn$ for an input graph G on n vertices.*

Proof. Consider the previous construction of $G_{\mathcal{T}}$. If we start the reduction from 3-edge coloring of a graph with $2m$ vertices and $3m$ edges, we get

$$n = |V| = 2k + 2|X| + |\mathcal{T}| = 2k + 8m.$$

Hence setting $n = \frac{8m}{1-2c}$ (if $1-2c > 0$ is rational, we may start with sufficiently many copies of the cubic graph to make n an integer) we get

$$2k = n - 8m = 8m\left(\frac{1}{1-2c} - 1\right) = 2cn$$

and hence our previous proof shows that deciding

$$cmp(G_{\mathcal{T}}) \leq k = cn$$

is NP-complete. ♠

2.2 CONV-graphs and Cartesian Coordinates

In this section we show that it is impossible to use Cartesian coordinate-system as a polynomially-sized certificate for existence of CONV-representation. To do so, we reuse the construction of [37] and modify it. First though, we define some notation and make some basic observations.

Note that a given arrangement of convex polygons can be considered as a planar embedding of a (planar) graph. Corners of individual polygons and intersection-points of their boundaries form vertices, particles of sides of polygons form edges. In this context it makes sense to talk about faces. To each such face we may assign its depth, which refers to the number of individual polygons containing this face. The notion of depth is a useful technical tool for analysis of convex-polygon arrangements [5]. Note that maximum depth is bounded from above by the maximum clique in a graph whose representation we are investigating. Thus given a representation of a cycle (of length larger than three), an appropriate representation consists of faces with depths 0, 1 and 2.

It is an easy observation that while looking for CONV-representation of a circle $C_n (= (\{v_1, v_2, ..., v_n\}, \bigcup_{i=1}^{n-1} \{v_i, v_{i+1}\} \cup \{v_n, v_1\}))$ where $n > 3$, polygons representing vertices of this circle form a Jordan circle in the plane, which allows us to place representants of non-adjacent vertices either inside or outside this topological circle. This circle is unique (unless the representation consists only of segments or is degenerate), but this is not important for us.

We use the Frame-gadget introduced in [37] (see Figure 2.7) and we prove that in any its representation the circle *abcd* has to be represented inside the representation of the circle *A...B...C...D....*

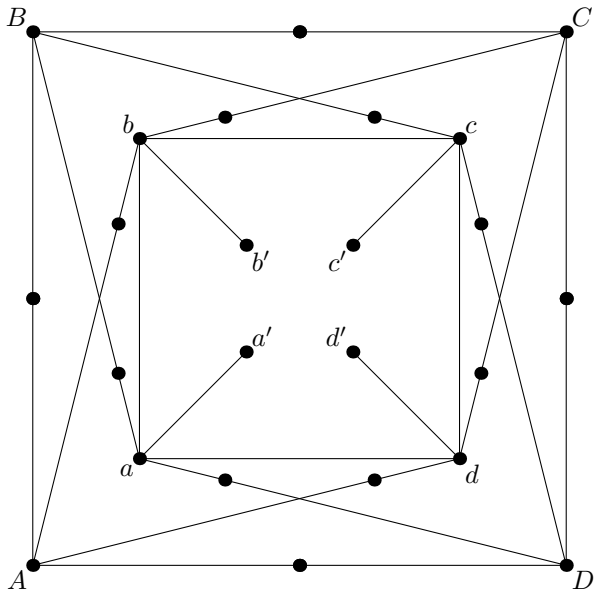


Figure 2.7: The Frame-gadget – the graph with property that in CONV-representation the circle *abcd* has to be inside the representation of *A...B...C...D....* We refer to the circle *abcd* as to *the small circle* while the circle around it containing $ABCD$ we call *the big circle.*

Lemma 9 *While representing a circle C_n with $n \geq 4$ by convex polygons, there appear only two faces not covered by any polygon (*i.e.*, with depth 0). We call them the outer and the inner face.*

Proof. We look at faces of depth 2. As intersecion of two convex polygons is a convex polygon, in a representation of C_n there are exactly n faces of depth 2 and each such face can be indexed by pair of vertices that lie consecutively on C_n. Note that each convex polygon is contained in exactly two consecutive faces of depth 2. Therefore individual faces of depth 2 are connected by faces of depth 1 in a circular way. Thus we get at least two faces of depth 0. As each polygon is contained in *exactly* two faces of depth 2, no other faces of depth 1 connect two faces of depth 2. Now note that faces of depth 1 are always incident to some depth-2 face. So the only way to have more than two faces of depth 0 is that any depth-0 face is adjacent only to depth-1 faces, which are themselves separated by depth-2 faces. But two faces in depth 2 are connected by depth-1 face covered by one (fixed) polygon and this is not possible when this polygon is convex. ♠

Lemma 10 *In the Frame-gadget the circle $\mathcal{D} = A..B..C..D$ has to be represented in the outer face of the representation of the circle $\mathcal{C} = abcd$.*

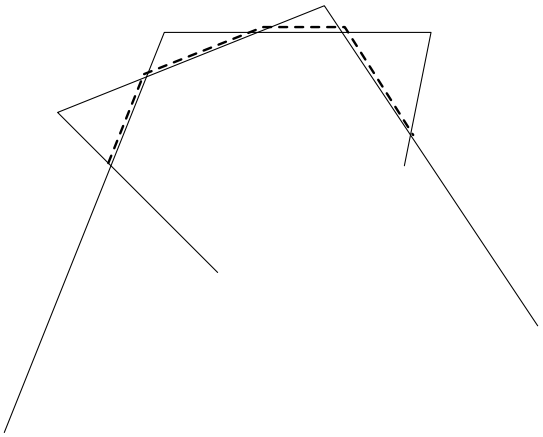


Figure 2.8: Demonstration of the cutting-procedure. When two polygons start alternating on the boundary, we start cutting-off particles of individual polygons along the dashed line up to the point where this alternation ends. In this way we establish an equivalent representation whose outer face forms a polygon without any convex angle, which causes a contradiction.

Proof. First let us note that $\mathcal{C}$ and $\mathcal{D}$ are non-intersecting circles, each of whose must be represented by convex polygons forming two faces. Thus either they are concentric, or non-concentric. The representation of $\mathcal{D}$ must be in the outer face of the representation of $\mathcal{C}$, since only the outer face (of $\mathcal{C}$) formed by the representation can be used to connect $\mathcal{D}$ to $\mathcal{C}$. This fact gets proven now by contradiction:

Let there exist a representation where the circle $\mathcal{D}$ is represented in the inner face with respect to the representation of $\mathcal{C}$ (let us call it $P_{\mathcal{C}}$). Then we know that on the boundary of the inner face given by $P_{\mathcal{C}}$, polygons have to appear in this ordering: *bacbdcad*.

We know that this face forms a polygon. Now we transform it into an equivalent representation and show that the polygon would not contain any convex angle, although we know that any polygon has to contain at least three convex inner angles. As the face we are acting in is surrounded by the representation of four-cycle, two polygons alternate on the boundary, then one of them disappears and a new one appears (and the new pair alternates again several times).

For each pair A, B we take two points: Where the alternation begins and where it finishes (without loss of generality, let around the surrounding circle appears as the first polygon A and thus polygon B occurres on the boundary after A stops appearing). Between these two points we start this cutting procedure: We follow the polygon A to the first intersection (with B). Then

we cut off the corner of B along the boundary of A to the next intersection. By the next intersection we cut off the corner of A along the boundary of B and so on up to the end of the alternation. After a slight perturbation we obtain an equivalent representation of the four-cycle with the property that the inner face is formed by the polygon with all inner angles non-convex. This follows from the fact that every corner of one polygon we obtain a non-convex angle (as the surrounding polygon has to be convex). If we take an intersection of two polygons then it is either some of the "inner intersections" which after the cutting-procedure is limited to the common border of two convex polygons (and thus we obtain non-convex angles too), or it is the first or last intersection. Then we know that between the first and the last intersection there had to be at least one "middle intersection" and after the cutting procedure the sector between the first and the last intersection is formed by the common border of two convex polygons. Also, the polygon that continues along the border is one of those bordering two and thus as the border is formed (among others) by the border of this polygon (and as this polygon is still convex), we obtain a non-convex angle, too. This is the contradiction (as each convex polygon contains at least three inner angles convex) and we know, that the only possibility how to represent the frame, is to place the representation of *abcd* to the inner face of the representation of the large circle.

Now we easily observe that if we add to the Frame-gadget a vertex X, that is connected with *abcd* by four disjoint paths (of length at least 2, one path to each of *abcd*), the only possibility where to place P_X is in the inner face of the representation of *abcd*. The outer face is used to represent the circle $\mathcal{D}$ and each of P_a, P_b, P_c, P_d is connected to some polygon representing $\mathcal{D}$. ♠

Now we introduce an explicit construction which uses graphs requiring such size in SEG-representation. Then we replace each vertex with a gadget, make these gadgets "specifically adjacent" and show that from any CONV-representation of such a graph we can extract SEG-representation of the original graph. Our gadget is shown in the Figure 2.9. We take a graph that requires in SEG-representation exponential size (when describing by Cartesian system of coordinates), replace each vertex by our gadget and for two adjacent vertices v, v_1 between gadgets W, W_1 place edges from A, a, x, c and C to A_1, a_1, x_1, c_1 and C_1.

Claim 11 *The graph, we have constructed is SEG-graph.*

Proof. The Frame-gadget has a representation with segments. We take this representation and transform it in the following way: By "pressing" P_A

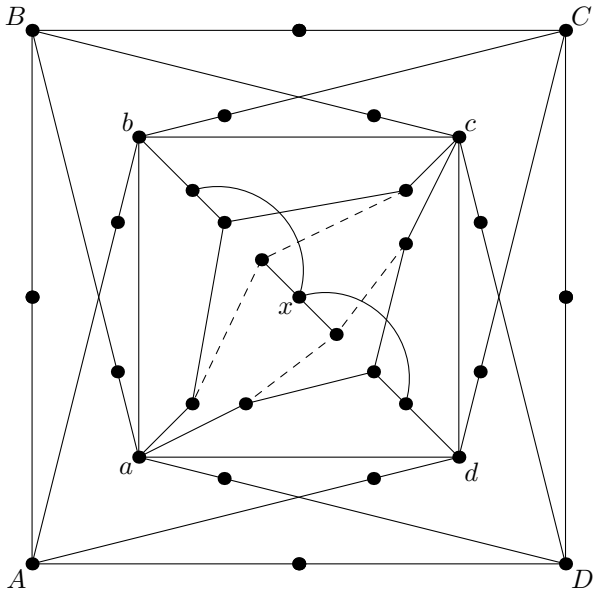


Figure 2.9: Gadget we use for replacing each vertex of the original graph. To dashed paths we refer as to M and N, polygons representing vertices on this path are called (together) P_M and P_N. To the inner face of representation of the circle consisting of path M and vertices a, b and c we refer as to "among P_M", to the inner face of $N \cup \{a, b, c\}$ we refer as to "below P_N".

against P_C we make this representation very thin. Then we pull P_x to become almost as long to pass through almost whole representation of gadget. Then we take SEG-representation of the original graph and replace each segment by our deformed gadget and we obtain SEG-representation of the new graph. ♠

Theorem 12 *There exists a sequence of graphs such that each graph is a CONV-graph and description of this representation requires double-exponential coordinates (with respect to the number of vertices).*

Proof. As we know that P_x has to be represented inside the representation of circle *abcd*, clearly for two neighboring vertices in the original graph our gadgets have to cross in a well described way: the inner face of the representation of *abcd* cannot contain any *a*-, *b*-, *c*- or *d*-vertex of another gadget W'. This is clear, as otherwise we could not intersect by polygon representing x-vertex all polygons we have to. Thus P'_b and P'_d have to lie in the outer face of representation of *abcd*. And thus P'_x has to pass through the representation of *abcd* "from left to right" and we know, that polygons representing path from x to b have to lie in the same outer face. Now for each gadget we take P_x and start the cutting procedure to obtain greedily minimal representation with respect to the complicacy of P_x-polygons. From each P_x we either obtain a segment (and we are done) or a triangle, or a rectangle.

Now we examine only the inner part of the Frame (circle *abcd* and its interior). From Lemma 10 we know that we may consider the circle *abcd* with its interior to form a convex polygon. Now we show that two gadgets representing two neighboring vertices have to cross "across", it means that a path to b passes through (WLOG) P'_a and path to d passes through P'_c. Otherwise we may consider that we are trying to connect P_x with the border of convex polygon replacing representation of circle *abcd*. And both connections are reached (when looking from P'_x) through (without loss of generality) P'_a. Then we know that on the border of the circle there appear always an interlacing pair of consecutive vertices. And then we know that (without loss of generality) P_c has to lie between those two points where paths from P_x reach the border of our replacement of representation of *abcd*-circle (representation of path to P_b and P_d). Thus it is impossible to intersect P_c with P'_c.

We can easily see that there is at least one corner of P_x among P_M and at least one below P_N. Moreover, P_x and P'_x can cross only between P_N and P_M. So, as P'_x crosses P_x across the gadget, we may take from each P_x one corner "among" P_M and one "below" P_N, we take the segment between them and from each CONV-representation of the new graph we obtain SEG-representation of the original graph. And as we took SEG-graphs requiring

exponential size of representation, made a constant blow-up, we constructed sequence of graphs with property that any their CONV-representation has exponential size. ♠

2.3 Subclasses of subtree-filament graphs

This section is based on author's common work with Jessica Enright [10].

As mentioned in the Introduction, subtree-filament graphs are exactly subtree-overlap graphs [11] and we reference them as class SOG. In this section we describe some results which are now being prepared for publication. We define classes of k-SOG graphs as classes of graphs representable as overlap-graphs of subtrees in a tree with (at most) k leaves.

Theorem 13 *The recognition problem of 2-SOG is in P. For any (fixed) $k > 2$ the recognition problem of k-SOG is NP-hard. Moreover, given any fixed tree $\mathcal{T}$ with maximum degree at least 3, it NP-hard to decide whether a given graph can be represented as overlap graph of subtrees of some tree $\mathcal{T}'$ where $\mathcal{T}'$ is obtained from tree $\mathcal{T}$ by subdividing its edges (or equivalently by replacing its edges by arbitrarily long paths).*

The first part of the proof is already known and follows from the fact that 2-SOG are exactly overlap graphs of subpaths in a path. This class is exactly class of interval-overlap graphs and also class of CIR-graphs, which are polynomially recognizable. The hardness part gets represented in three steps:

2.3.1 Tree with three leaves

Lemma 14 *It is NP-hard to recognize 3-SOG.*

Proof. We shall represent a graph by subtrees of three (arbitrarily long) paths connected to one common vertex. To prove that this is an NP-hard problem we reduce 3-CON-3-COL. Before we start, we modify the input graph G' in the following way: Instead of G', we need at least 5 copies of it and we need a 3-connected graph, thus we take two disjoint copies G' and G'', we pick one (arbitrary) edge uv and we make adjacent copy of u in G' with copy of v in G''. Let us denote these two copies as H. Now we take three copies of H and for each vertex v, we place a triangle on its three copies. In this way we obtain a 3-connected graph G consisting of 6 copies of the original graph G'.

First we use the fact that in any subtree-overlap representation of the graph S_1 (see Figure 2.10) either a tree representing s is contained in a tree representing b or vice versa [12]. In the rest of this section we consider without loss of generality the former situation, *i.e.*, s gets represented by a subtree of b (mnemonically, vertex s is *the small*, while b is *the big*). For the graph S_1 we

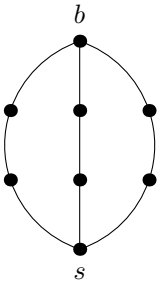


Figure 2.10: The S_1-graph having such a property that when representing it by overlaping subtrees of a tree, either the subtree representing s is a subtree of the tree representing b.

observe that there are only two ways how to represent it as an overlap graph of subdivided star with three leaves. In both cases, both vertices, s and b, be represented over the branching vertex. We always include the S_1-gadget into the graph we are asking for its representability.

Given an instance $G = (V, E)$ of 3-CON-3-COL, for each $v \in V$ we establish two vertices a_v, b_v such that a_v's form an independent set, while b_v's form a clique. Moreover, b_v-vertices are adjacent to, both, b and s. Each a_v is adjacent to corresponding b_v. For each edge $e = \{u, v\}$ we add a vertex c_e. These vertices behave very similarly to b_v. It means, they form a clique, they are adjacent to s and b, for each vertex w there is an edge $c_e b_w$ and each c_e is also adjacent to a_v and a_u (*i.e.*, to a-vertices of its incident vertices).

Coloring of vertices $v \in G$ corresponds to representing individual a_v's onto one wire in the subdivided star. In this way we observe that each 3-colorable graph can be represented (see Figure 2.11). To show the converse, we calculate how many "mistakes" can be done by inappropriate representations of a-vertices. We show that at most edges incident to four vertices can get represented otherwise than with its endvertices on two different paths and as we took 6 copies, at least one copy must be properly colored.

Under these assumptions we can observe:

1. At most one a_v gets represented as a subset (subtree) of the representation of b (for such a pair of vertices we would not be able to represent their c-vertices to overlap b and s and avoiding each other's a-vertex).

2. No representation of any a_v can lie inside the representation of b and avoid a representation of s at the same time (as there are "technical wires" already present and surrounding all three endvertices in the representation of both s and b), so each a-vertex is represented either inside s or outside b.

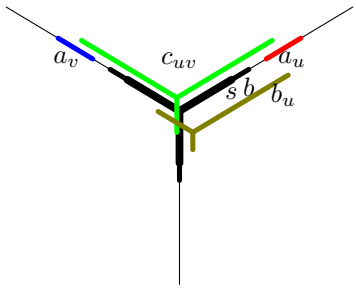


Figure 2.11: Given a 3-coloring of the original graph, for a particular vertex colored by color i, we represent its a-vertex on ith wire. Corresponding b-vertices are represented in the following way: We start inside corresponding a-vertex, continue towards the branching-vertex, there we split and attach two short "legs". Subtrees corresponding to a-vertices placed further away from the branching vertex get shorter "legs". This causes a clique on b-vertices. Each c-vertex (corresponding to individual edge) passes from one wire to another. Note that two c-vertices represented on the same pair of wires need not to overlap so far. This problem is solved in the same way as for b-vertices, *i.e.*, by attaching one short "leg". If we have one tree contained in another, then we attach a new "leg" to each in such a way that the smaller tree gets a longer "leg". Note that generally c-vertices get shorter "leg" than b-vertices to get a complete bipartite graph on b- and c-vertices.

3. Given representations of two a-vertices (a_u, a_v) that are inside the representation of s or outside b, neither contains the other. (We do not say what contains the a-vertex whose representation contains the whole representation of b!)

4. It may happen that for some pair of neighboring vertices a-vertices of both endpoints are represented on the same branch. Such vertices are called *bad guys*. Should bad guys occur on two (or more) branches, say as a_u, a_v on one branch and a_w, a_x on another representing vertices connected by edges uv and wx, then it is impossible to represent b_{uv} and b_{wx} so as to overlap their respective a-vertices and each other simultaneously. Moreover, note that these bad guys always occur in pairs. One member of this pair lies always inside s and other outside b (or at least one containing the whole b; otherwise b_e cannot overlap b (and not even s).

5. Bad-guy-pairs must not be nested (they must overlap each other; otherwise we cannot force respective b_e and b_f to overlap as one must contain the other).

6. Bad guys which are not the two "middle-ones", must have no legally colored neighbor (otherwise respective b_e avoids b_f passing from any vertex closer towards the middle).

7. A graph with minimum degree 3 cannot be represented by "bad guys" only.

   Proof. Either it is K_4 (and we check by brute force that at most two its vertices can be represented on one branch, the other must at least partially lie in the other branches), or it has at least 5 vertices and then we draw a picture showing that the fifth vertex must have two edges to the second vertex (and therefore it is not a graph, but a multigraph). ♠

8. As a corollary of 6 and 7, if each 3-connected component of an instance of 3-COL has minimum degree at least 3, then the whole graph can be represented using at most two bad guys (a single bad guy is impossible). More of them cannot be used, because collapse of the whole 3-connected component is impossible (either it is K_4 – special analysis why it is impossible, or it has at least 5 vertices and then we draw until vertex 5 has two edges to vertex 2). If at least 3 vertices in one 3-connected component collapse into bad guys, from 3-connectivity at least 3 of

them must have legally-colored neighbors violating 6. Now we argue that among all 3-connected components at most 2 bad guys appear: Each 3-connected component contains at least 4 vertices and each bad guy lies in some such component. Therefore each bad guy has at least one legally-colored neighbor and at most two bad-guys are permitted to have such a neighbor.

9. As a corollary of 8, 3 and 1 at most one a-vertex may be represented over the branching vertex (a vertex of degree 3), at most one a-vertex may be represented over the representation of b and at most one pair of a-vertices is represented by bad guys. For all other a-vertices belonging to a particular branch, we give them a color corresponding to this branch. So in this way we may counterfeit coloring of at most 3 vertices (these three vertices in two cases simulate that the vertex has all three colors depending on which color is prohibited and in the third case one monochromatic edge can be represented). But we take (together) 6 copies of the original graph and if the original graph was not 3-colorable, a defect must appear in each of these six copies, while only three of them can be masked by "inappropriate coloring" (*i.e.*, twice an a-vertex represented over the branching vertex (once inside s, once around b) and once bad-guys).

♠

2.3.2 General fixed number of leaves

Lemma 15 *Given any fixed tree $\mathcal{T}$ with maximum degree at least three, it is NP-complete to decide whether G can be represented as a subtree-overlap graph of any $\mathcal{T}'$, where $\mathcal{T}'$ is obtained from $\mathcal{T}$ by subdividing its edges.*

To prove this lemma, we need following terminology:

Definition 16 *In a tree T a vertex v of degree larger than two is called leaf-like if removal of v splits T into at most one tree U with maximum degree larger than two and arbitrarily large set of paths $P_1, \ldots, P_k$.*

Proof. We proceed in a very similar way to the previous proof. Let k be number of leaves of T and l minimum degree of leaf-like vertices. We reduce the k-CON-k-COL problem. Again at first we add copies of a given graph to get at least 5 its copies. Instead of S_1 we establish U_k^l as depicted in the Figure 2.12. This U_k^l can be represented only in such a way that without

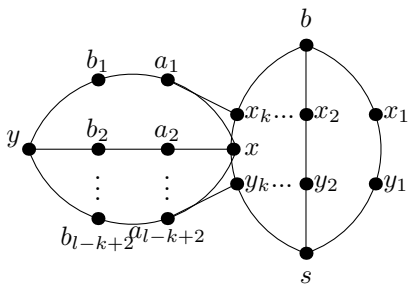


Figure 2.12: Gadget U_k^l consists of $k-1$ paths of length 2 connecting vertices b and s and one path of length 3 connecting these vertices. The middle vertex of the latter path is x and a similar gadget with $l-k+2$ paths is attached to the vertex x: the first vertices of the first and of the last paths are adjacent to x_k and y_k, respectively.

loss of generality, this U_k^l can only be represented in such a way that s gets represented by a subtree of b and s has to be represented over some vertex of degree larger than 2. Similarly, vertex x, which must be represented by a subtree disjoint with subtree representing s, must be represented over some branching vertex. Note that it is necessary to find l disjoint paths from boundary of subtree representing s to subtree representing b and $k-l+2$ disjoint paths from boundary of subtree representing x to the boundary of subtree representing y. Note that this is possible only when the representation of s contains some leaf-like node of degree l, while the representation of x (and as well y) contains all other branching vertices. The rest of the construction is similar to the case of 3 colors and the analysis is also almost the same. It means, again, there can be at most two *bad guys*, a-vertices of at most one vertex (of the original graph) may be represented over a branching node (inside the representation of s), at most one may contain the representation of b and at most one gets represented between the branching-vertex in subtree representing s and subtree representing x. Again no other pair of a-vertices contain each other and coloring of v in the original graph by color i corresponds to a representation of a_v on path from ith leaf. ♠

Corollary 17 *For all $k > 2$, it is NP-hard to recognize k-SOG.*

Proof. Follows from Lemma 15. We perform the same construction, now considering the minimum degree of leaf-like vertices to be 3. By the arguments of Lemma 15 we see that we have to represent the vertex s (we still consider without loss of generality that s gets represented by a subtree of b) over some leaf-like vertex of degree 3. Vertex x then gets represented over all other vertices of degree larger than 3. The rest of the argument then holds. ♠

Chapter 3

Recognition Problem and Sandwiching

Since many intersection classes are defined, instead of asking for the recognition problem for each class separately, it is more useful to show that no polynomially recognizable class can be sandwiched between two particular classes or that polynomial recognition is preserved under certain conditions. Given three classes $A \subseteq B \subseteq C$, we say that the class B is sandwiched between classes A and C. This chapter is dedicated to results on sandwiching.

3.1 Homothetic polygons in the plane

Proposition 18 *Homothetic convex bodies in a plane in general position (*i.e.*, no pair has an infinite number of common points on their boundaries) form an arrangement of pseudo-disks.*

Proof. Consider two homothetic convex bodies as given in the Theorem. Firstly, we consider the case where one body is smaller. In this case we may use the Banach fixed point theorem for linear mapping which maps the bigger polygon to the smaller one (it can be obtained as a composition of shift and scaling). From Banach theorem we obtain a fixed point (it suffices to know it exists, we do not need an efficient algorithm constructing it). From this fixed point we start drawing both convex bodies in following way on rays (from the fixed point): Either the fixed point is inside (both) bodies, then there appears no intersection. Or the fixed-point is outside (both) bodies. Then we know that our drawing procedure places appropriate points can be considered in such a way that we start emanating a ray from the fixed point in each direction, *i.e.*, we behave like a radar watching its neighborhood. In

this radar-like method we place/reach (on each ray) points of boundaries of our convex bodies in this ordering:

1. We reach the start-point of the smaller body,
2. we reach the start-point of the bigger body or the endpoint of smaller body,
3. the second from the previous point,
4. endpoint of the bigger body.

Let there be at least three rays where the second and the third points (*i.e.*, points 2 and 3 of the "radar" algorithm in previous paragraph) are the same (*i.e.*, the smaller body ends where the bigger body begins). Then these three points either lie on a common line or not. In the former case, in order to preserve both polygons between these three points to be convex, we see that the line between these three points has to be a common border (with infinitely many common points) and we have a contradiction. In the latter case, these three points do not lie on a common line, and thus the border of one of the polygons cannot be convex (the middle point excludes points between the outer two points for one of the bodies). If the fixed point is on the boundary, we start "distributing" the borders of convex bodies with placing the start-point of the smaller and start-point of the larger. Then we only have to place the endpoints which have to be distinct (as one body is smaller, one is larger) and we obtain only one common point of borders (as otherwise we would have again infinitely many common points).

For two convex bodies of the same size we proceed in the same way, except that we use parallel lines instead of rays (in appropriate direction) and the argumentation is the same (which reflects the fact that the "fixed point" lies in the infinity). This completes the proof. ♠

Observation 19 *As the convex polygons are not allowed to intersect only on their boundaries (they are required to properly intersect or not), we may consider that any representation is perturbation resistant,* i.e.*, we may slightly move with any polygon in any direction and obtain (topologically) the same result. Thus we may consider only representation satisfying the assumptions of the previous proposition.*

Theorem 20 *It is NP-complete to decide whether a graph is an intersection graph of homothetic copies of a fixed convex polygons.*

Proof. For triangles the proof can be found in [29].

We use the construction introduced in [27]. It is known that for an unsatisfiable formula the graph (obtained from the construction, see details below) cannot be represented by pseudo-disks (and due to the previous proposition the graphs representable by our homothetic polygons form a subset of graphs representable by pseudo-disks). Thus it suffices to show that the graph obtained from any satisfiable formula can be represented by homothetic polygons in a plane. To do so, we have to introduce following terminology:

Definition 21 *For a convex polygon P we choose some orthogonal basis b_1, b_2 such that all sides of the polygon P are neither parallel to b_1, nor to b_2. Given such a basis b_1, b_2, we consider the smallest axis aligned rectangle containing P (its bounding box) and denote it by $BB(P)$. One can easily see that we may choose a basis such that P touches $BB(P)$ inside $BB(P)$'s edges (not in corners).*

For an arrangement of homothetic convex polygons we may pick such a basis b_1, b_2. As we will not be interested in this basis but only in the bounding boxes, we will not mention that $BB(P)$ depends on the choice of basis. The basis will be fixed so that no corner of $BB(P)$ is an elements of P.

Now let us recall the reduction. Since the reduction is already described in [27], we do so briefly:

We reduce E3-NAE-SAT(4) known to be NP-complete to representability by homothetic polygons of any shape. The graph consists of gadgets for clauses, gadgets for variables and connections between them. The gadget for a variable is C_8 with vertices $c_1, ...c_8$, which can clearly be represented by homothetic polygons in a plane. Each occurrence of the variable is represented by 2 consecutive vertices (c_{2k-1}, c_{2k}). If the variable's occurrence is negated, we swap the labels of c_{2k-1} and c_{2k}. The truth assignment in the representation is determined by the orientation of the polygons $P_{c_1}, ...P_{c_8}$ (which may be either clockwise or anti-clockwise). The connections are represented by a ladder (see Figure 3.1) and crucial for the construction is the fact (proven in [27]) that the ladder cannot distort (*i.e.*, swap vertices "from the left to the right"). The path that begins on the left side, never crosses the path on the right side.

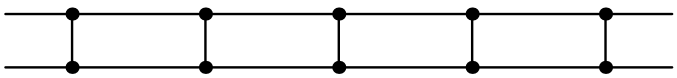


Figure 3.1: The ladder

The ladders representing the first and the second occurrence (of the particular variable) are connected by a "cross-over"-gadget. The third and the fourth occurrence, too. We want to obtain the ladders representing particular occurrences (of the variable) with respect to clockwise orientation for positive and negative variable to appear around the variable-gadget always in the same ordering (*i.e.*, ladder 1, ladder 2, ladder 3, ladder 4). This cross-over ensures this. If the variable is assigned true, we make the respective ladders touch, if the variable is assigned false, we cross them. The crucial fact (proven in [27]) is that it is impossible for ladders to be twisted: There is always a "left" row and a "right" row. The cross-over gadget is depicted in the Figure 3.2.

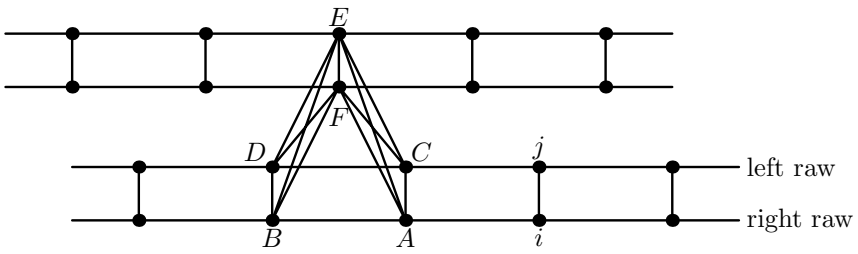


Figure 3.2: Figure shows one cross-over on two ladders. One row in each ladder is called the left one, the other the right one. Vertex labels correspond to the labels on the next figure.

Figure 3.3 shows how two ladders can touch or cross. In following text we use the names from the figure.

More precisely, if we want to cross two ladders, we represent P_A and P_B by polygons of the same size crossing only slightly. We create P_C and P_D to be of the same size. So we obtain a quadruple of polygons of width $(2-\delta)\cdot \text{width}(P_A)$ and height $(1+\varepsilon)\cdot \text{height}(P_A)$. We choose factor μ and create polygons P_E and P_F scaled to P_A by factor $(1+\mu)$ and make them crossing slightly more than are P_A and P_B. Now we are using the fact about bounding boxes that except for small intervals around (four) points where the polygon touches the boundary of its bounding box, there is a small stripe (inside the boundary of the bounding box) disjoint with the polygon.

If we want the ladders to touch only, $P_A ... P_D$ get represented in the same manner. We represent P_E by a polygon obtained in this way: We take P_E as a copy of P_A (placed over P_A). Then we shift it to the left (to avoid intersection with P_i) and to the bottom. Then we scale it slightly to intersect P_D (it can be done by scaling by factor $(1+2\cdot\varepsilon)$. Now we have a proper representation of the whole gadget except P_E. To represent P_E, we take a

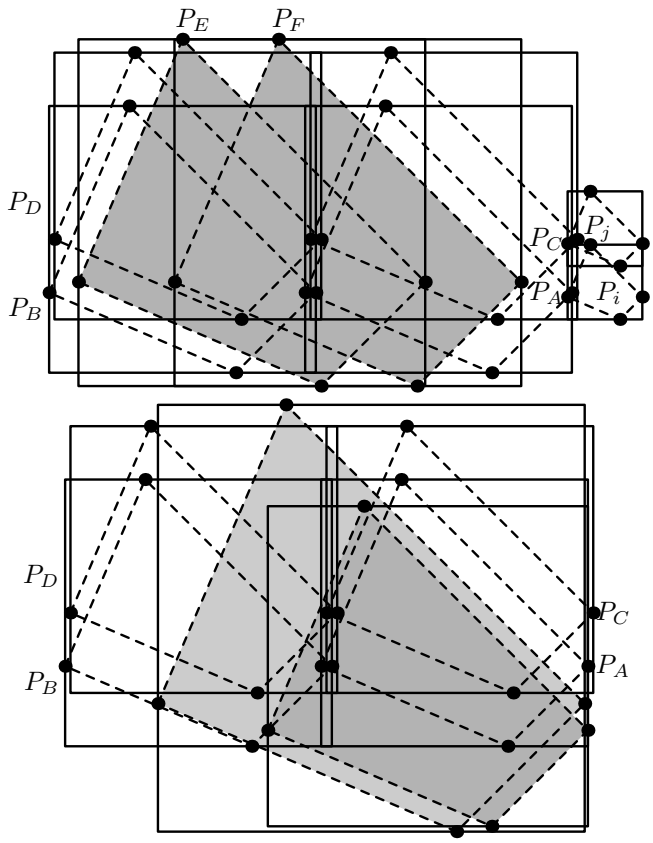


Figure 3.3: The first picture shows how to cross representations of two ladders in cross-over, the second shows how to touch them only.

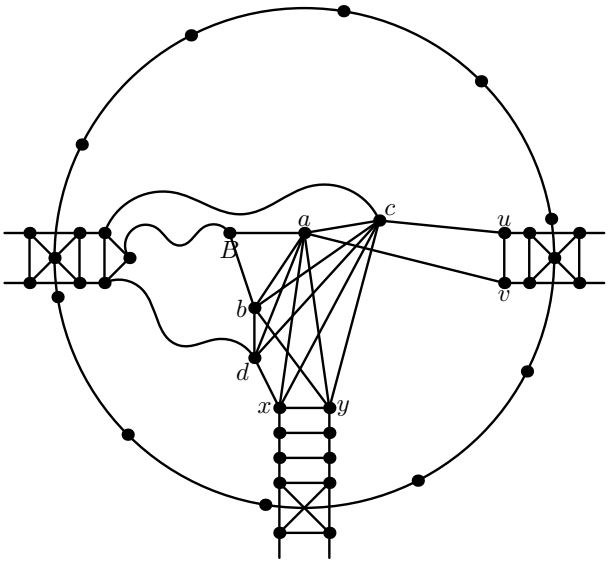


Figure 3.4: The clause gadget, wavy "edges" depict arbitrarily long paths.

copy of P_D (placed over P_D). Then we shift it slightly to the right and start scaling it up to force the right place of intersections with P_D. We surely may stop scaling before the critical factor $(2-\delta)$ is reached as for the factor $2-\delta$ we could place the polygon to intersect $P_A ... P_E$ and moreover the right-most corner could be placed below the (bottom) intersection of P_A with P_i.

Whenever two ladders leading from variables to clauses cross, we represent this crossing by the cross-over, too.

After cross-overs the ladder enters **the clause gadget**. This is represented by a surrounding circle and a structure inside it (see Figure 3.4).

We can easily see that in the gadget problems can occur only when representing vertices *abcdB* and their neighbors. How to do so is depicted in the Figure 3.5 and its description, which completes the proof (it suffices to note that the polygons inside their bounding boxes are convex). Just note that we analyze the situations on positions of polygons P_a, P_c and P_x, P_y.

♠

Corollary 22 *For any shape between the intersection class of corresponding homothetic convex polygons and PDISK no polynomially recognizable class can be sandwiched.*

Proof. The reduction is exactly the same for PDISK as for homothetic convex polygons. Therefore we obtain either a graph of homothetic convex polygons or a graph which is not even PDISK-graph. ♠

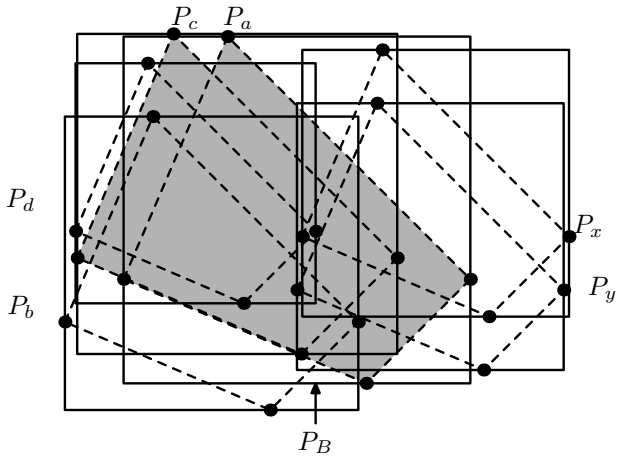
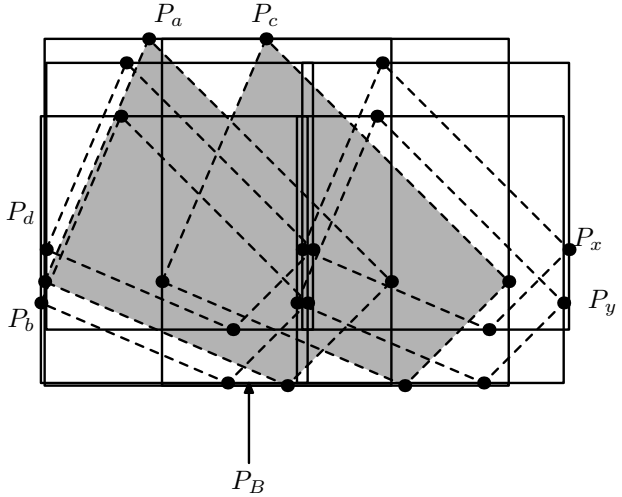
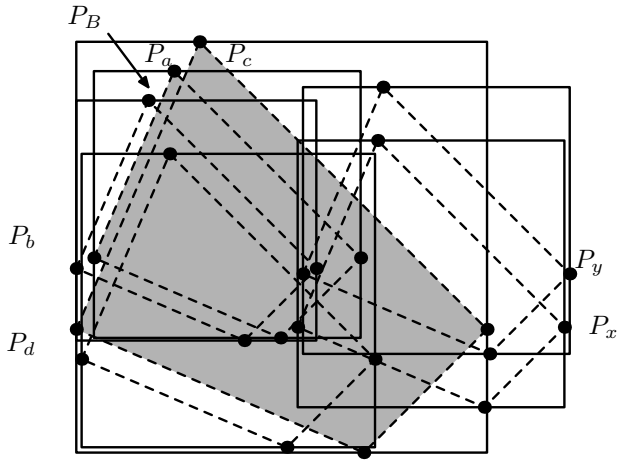


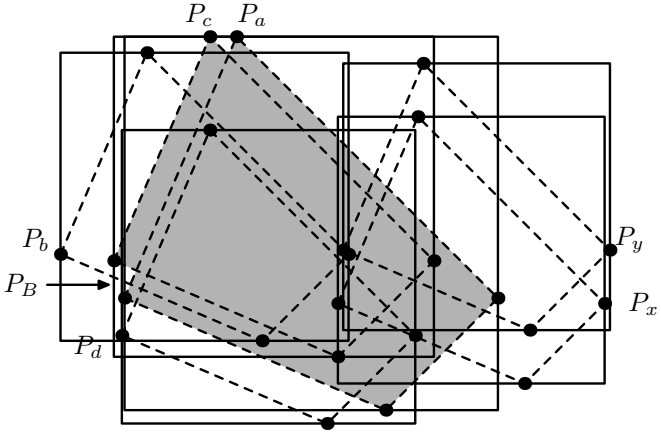
Figure 3.5: Case 1 – False/True: We build this representation as while making cross-over, but P_a has y-coordinate of its left touch-point between y-coordinates of left touch-points of P_d and P_b (and the x-coordinates the same), similarly for the top and bottom touch-point. P_c will be created as a copy of P_a slightly scaled down and shifted to the top and to the left to be from the bottom covered by P_a and to have the left and top touch-point with the same properties as P_c.



Case 2 – True/True is just special case of the cross-over.



Case 3 – Like Case 1, but instead of P_a we start with P_c. Then we create P_a as a copy of P_c, scale P_a down slightly, move slightly upwards and even more slightly to the left to obtain the left touch-point of P_a's x-coordinate still larger than that of P_c, but the top touch-point to be on the same y-coordinate and x-coordinate slightly lower.



Case 4 – False/False: We proceed like in case 1. After we add P_c, we obtain P_a as a copy of P_a slightly shifted to the right and we scale it up to obtain x-coordinate of the left touch-point of P_a less or equal than that of P_c and y-coordinate of the left touch-point of P_a between those of P_b and P_d. Now we start scaling P_b up to cover by it the intersection point of P_a and P_c and we are done.

3.2 Polynomial reduction for PC-graphs

First we show that the recognition problem for PC-graphs is in NP (as for several classes, like segment-graphs or CONV-graphs it is not known whether respective recognition problem is in NP [37]). We use the notion of *alternating sequence*, which was first introduced by Bouchet [3] to recognize circle-graphs:

Claim 23 *G is a PC-graph if and only if it has an alternating representation. Moreover, for a vertex of degree $d \geq 2$, at most d occurrences are sufficient, for vertices of degree at most one two occurrences are sufficient. For a vertex v we use P_v to denote the polygon representing v in a PC-representation.*

Proof. This is a well-known fact. The alternating sequence can be obtained from a PC-representation when we take an arbitrary start-point on the boundary of the circle, go around the circle up to the start-point and write down names of polygons having a corner on the boundary of the circle. For a vertex v we use P_v to denote the polygon representing v in the PC-representation. ♠

Lemma 24 *The recognition problem for PC-graphs is in NP.*

Proof. We guess an alternating sequence (which is polynomially large with respect to the size of the graph) and verify that exactly those pairs forming an edge alternate.

Definition 25 *We say that a polygon P is* visible *from a point x if there exists a ray starting in x that intersects P in a point y such that no point of the segment xy is contained in any other polygon of the representation. We say that a polygon P is visible from a polygon Q if there exists a point $x \in Q$ such that P is visible from x.*

Now we reduce the problem PURE-E3-NAE-SAT to the recognition of PC-graphs. Given formula ϕ, we define a graph G as follows: We start by two non-neighboring vertices e_1 and e_2. Then we order (and number) variables $v_1, v_2, ..., v_n$ of ϕ. Each variable v contributes to the graph by two important vertices v and $\bar{v}$. Technical details of the construction force us to represent each v_i and $\bar{v}_i$ with polygons which lie under P_{e_1} with respect to P_{e_2} or under P_{e_2} with respect to P_{e_1} (*i.e.*, for no i neither $P_{e_1}, P_{e_2}, P_{v_i}$ nor $P_{e_1}, P_{e_2}, P_{\bar{v}_i}$ lie around the circle). When v gets represented in the place under P_{e_1} and $\bar{v}$ under P_{e_2}, v is assigned TRUE, if v is represented under P_{e_2} and $\bar{v}$ under P_{e_1}, v is FALSE, all other possible placements get obstructed (by not yet described constructions). Under P_{e_1} and P_{e_2} (with respect to

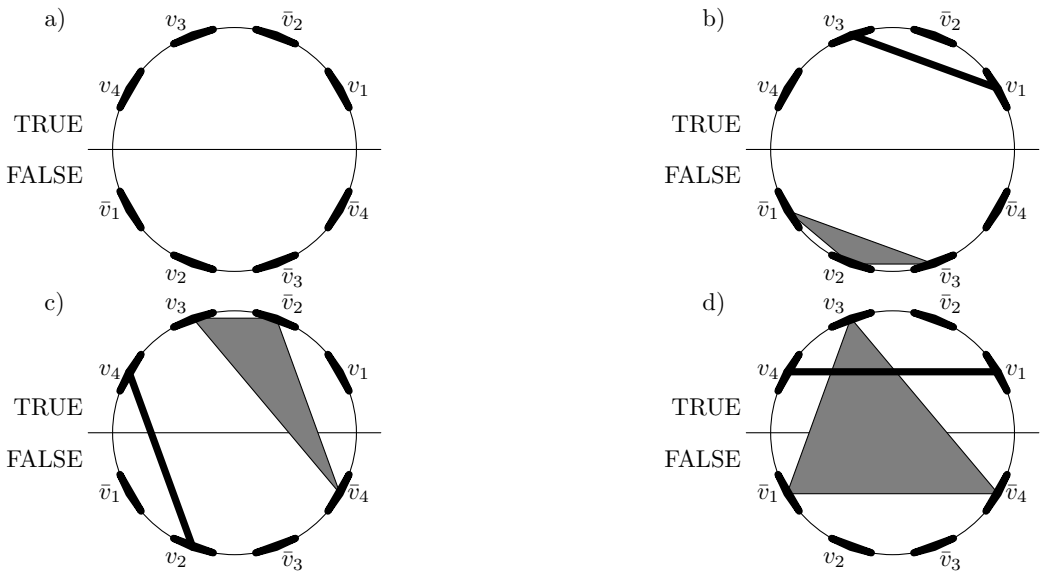


Figure 3.6: Case (a) shows how vertices representing literals should be distributed around the circle, cases (b) and (c) describe a representation of satisfied clauses, case (d) shows an unsatisfied clause – note that in the last case polygons representing the clause must intersect each other. All other possibilities are equivalent to one of those depicted.

each other) vertices v_i and $\bar{v}_i$ get represented, in the ordering of increasing index as depicted on the first picture of Figure 3.6 where instead of P_{e_1} and P_{e_2} we show a line separating them. We represent each clause (v_1, v_2, v_3) by two mutually non-adjacent vertices q_1 and q_2, where q_1 is adjacent to $\bar{v_1}, v_2$ and $\bar{v_3}$, and q_2 to v_1 and v_3.

Note that under assumptions of the last paragraph the "clause"-vertices (q_1, q_2) can be properly represented if and only if neither all three of v_1, v_2, v_3 are assigned TRUE nor (all three) FALSE. These ideas are demonstrated in the Figure 3.6.

Now we describe the technical part of the reduction. We give a description of what should be done for three consecutive variables (with respect to the indices of variables) A, B and C and for a clause Q. Variable B contributes 16 vertices: $a_1, ...a_5, b_1, ...b_5, c_1, ...c_4, r_1, r_2$. These vertices behave as in the Figure 3.7. Variable A contributes 16 vertices with the same labels as for B, but with a prime. These two variables share four vertices in the following way: $a'_1 = a_4, a'_2 = a_5, b'_4 = b_1$ and $b'_5 = b_2$. In the same way C contributes 16 vertices with double-primes, and so: $b''_1 = b_4, b''_2 = b_5, a''_4 = a_1$ and $a''_5 = a_2$. All c- and r- vertices of distinct variables are adjacent. We add a 6-cycle $e_1, x_1, x_2, e_2, x_3, x_4$. We make the vertex e_1 adjacent to all c-, r- and a-vertices

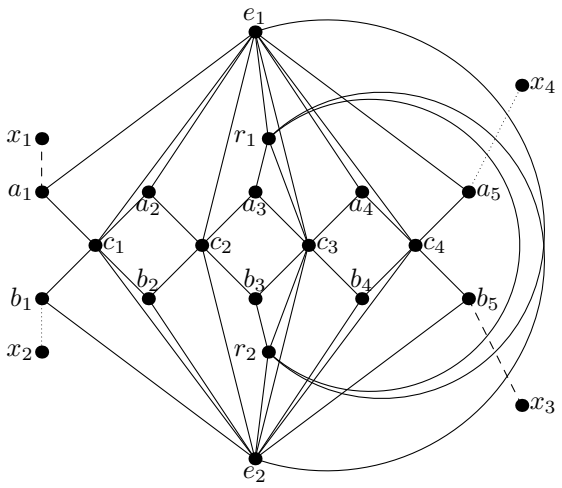


Figure 3.7: Variable gadget. Vertices a_1, a_2, b_4, b_5 and a_5, a_4, b_2, b_1 are shared by gadgets with index larger or smaller by one, respectively. Dashed and dotted edges are valid only for gadgets belonging to the first or the last variable, respectively.

except all a_3's and the vertex e_2 adjacent to all c-, r- and b-vertices except b_3's. When representing variable v_1, we connect its copy of a_1 to x_1 and b_5 to x_4, similarly for the last variable we connect its b_1 to x_2 and a_5 to x_3.

For each clause Q containing literals l_1, l_2, l_3 we add two nonadjacent vertices q_1 and q_2 make both adjacent to all c- and r-vertices, to e_1, e_2 and to all vertices representing the other clauses. Then we make q_1 adjacent with the b_3-vertices of the gadgets representing variables l_1 and l_3 and to the a_3-vertex of the gadget representing l_2 (note that ϕ has only positive occurrences). Finally we connect q_2 with the a_3-vertices of the gadgets representing l_1 and l_3 (not for l_2, it is okay to do so, but more convenient not to).

Note that the a_3- and b_3-vertices of respective variables are exactly the vertices v and $\bar{v}$ introduced in the idea of the reduction. When we prove that a_3's and b_3's appear around the circle in desired ordering (*i.e.*, such that there are two disjoint half-circles in each of which one of a_3 and b_3 corresponding to each variable v_i occur, and do so in order of increasing index of the corresponding v_i), we are done. This gets proven by following lemma with easy consequence that the variable-gadget must be represented as depicted in the Figure 3.8.

Lemma 26 *When representing one fixed variable gadget, polygons representing its $a_1, a_2, a_4, a_5, b_5, b_4, b_2$ and b_1 appear around the circle and, moreover, in this ordering.*

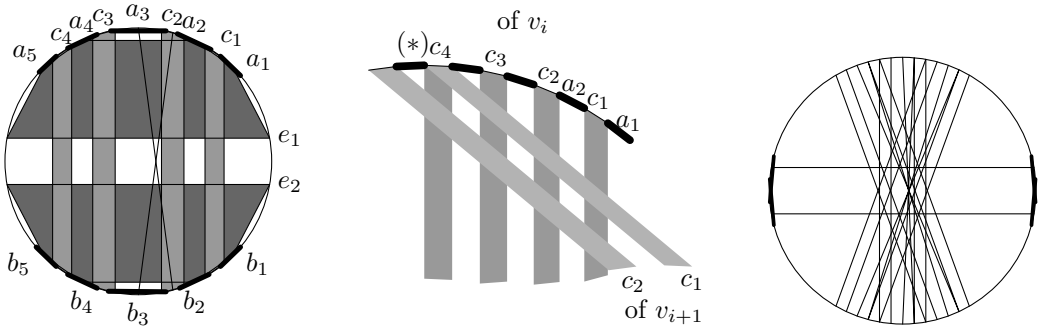


Figure 3.8: The picture on the left describes the unique possibility of how to represent the variable-gadget (up to switching a_3 and b_3), the picture in the middle describes how representants of neighboring variables are represented and share vertices, polygon (*) represents a_5 of v_i and simultaneously a_2 of v_{i+1}, the picture on the right sketches representation of c-vertices belonging to three consecutive variables.

Note that sharing vertices with "neighboring" gadgets propagates this orientation to all other variable-gadgets.

Proof. First of all we observe that all mentioned polygons must lie around the circle, otherwise (if one is blocked by another from the third one, by simple case-analysis we find a path from the first to the third one avoiding neighbors of the second — a contradiction).

Another remarkable observation is that a- and b- vertices must not interlace (*i.e.*, around the circle a- and b- vertices are represented in two disjoint circular-arcs). This is straightforward as otherwise we cannot represent vertices e_1 and e_2.

The rest is simple observation following from the fact that C_4 can be represented only by four-tuple of polygons lying around the circle, thus the subgraph consisting of $a_2, a_3, a_4, b_4, b_3, b_2, c_1, ...c_4$ enforces polygons representing a_2, a_4, b_4 and b_2 in desired way. Polygons representing a_1, a_5, b_5 and b_1 follow from simple case analysis (another placement immediately leads to irrepresentability). ♠

Lemma 27 *The constructed graph without clause-vertices can be represented only in such a way that polygons representing the a_3's and b_3's appear around the circle, and consecutive literals (around the circle) have increasing indices (mod n).*

Proof. Corollary of Lemma 26.

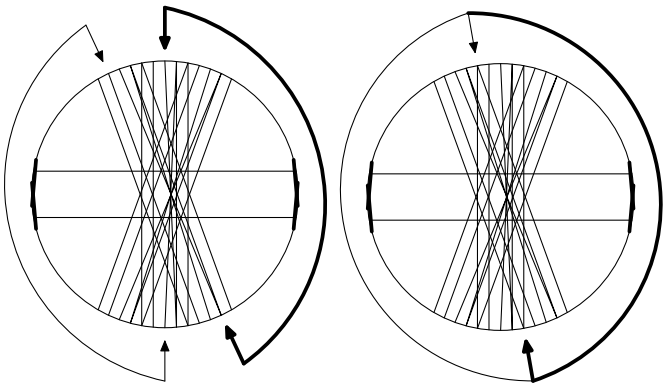


Figure 3.9: Illustration to the proof of Lemma 28. We demonstrate how to extend P_{q_1} and P_{q_2} with respect to representants of c- and r-vertices of particular variables, arrows show some of the corners of q-polygons, arrows connected by a curve belong to one another. First we extend the polygons to cover the area depicted in the left picture, then we extend both polygons to almost touch as depicted in the right picture.

Now we know that an unsatisfiable formula cannot be represented (because polygons representing corresponding vertices must intersect each other although they represent non-adjacent vertices). Thus we have to show:

Lemma 28 *For a satisfiable formula ϕ, it is possible to represent the graph obtained from our construction.*

Proof. After representing the variable-gadgets as depicted in the Figure 3.8 for each clause we add two vertices q_1 (having three neighbors among the a_3's and b_3's) and q_2 (having two such neighbors) in such a way to intersect relevant P_{a_3}'s and P_{b_3}'s and extend them to correctly intersect all auxiliary polygons. We choose one variable v_i of l_1 and l_3 which is the "end-variable" for P_{q_1} (*i.e.*, which is not blocked by P_{q_1} from P_{a_3}'s of l_1 and l_3)[1]. We extend P_{q_1} and P_{q_2} to cover almost half of the circle (we proceed for P_{q_1}, P_{q_2} is similar): P_{q_1} has three corners intersecting relevant P_{a_3} and P_{b_3}'s. We add a corner intersecting P_{c_2} of v_{i-1}. Now except P_{c_3} and P_{c_4} of v_{i-1} and v_i we are intersecting exactly what we should. So we add a corner under P_{c_4} of v_{i-1} with respect to P_{x_1} and P_{x_3} and simultaneously P_{c_1} of v_i and we are done.

[1]Formally when exploring this blocking we consider P_{q_1} restricted not to intersect P_{b_3} of l_1 and l_3 because blocking is defined only for disjoint polygons.

Lemma 28 finishes the proof of the theorem:

Theorem 29 *It is NP-complete to recognize whether a graph G has a PC-representation.*

3.3 Extension for IFA-graphs

Definition 30 *Let G be an interval-filament graph G and R be its representation. For each vertex $v \in V(G)$ we denote by F_v the filament corresponding to v and by I_v the underlying interval (of the filament).*

We are interested in the "topological" position for three mutually non-intersecting filaments. Note that two principal positions are that either from each filament either both other are visible or neither are. This visibility can be defined in the following technical way:

Definition 31 *For a triple of non-intersecting filaments F_a, F_b and F_c we say that F_a is covered by F_b against F_c if either $I_a \subset I_b \subset I_c$ (or the reverse) or $I_a \subset I_b$ and $I_b \cap I_c = \emptyset$ (or the reverse). For a set $\mathcal{F}$ of filaments we say this set lies consecutively on the base-line if F_a is not covered by F_b against F_c for any triple $F_a, F_b, F_c \in \mathcal{F}$.*

Theorem 32 *It is NP-complete to decide whether a given graph is an IFA-graph.*

The theorem is proven in two parts. First it is necessary to show that the recognition problem is in NP. This was already proven in [19]:

Theorem 33 *G is IFA-graph if and only if it is co-(co-INT-mixed)-graph.*

Corollary 34 *The recognition problem of IFA-graphs is in NP.*

Proof. We guess an appropriate INT-graph H (with an interval representation), an appropriate partial-ordering P of vertices and check correctness. ♠

For the reduction we use almost the same graph as for PC-graphs and the same basic ideas. We subdivide each a- and b-vertex into two new vertices for a_i named f_i, g_i and for b_i we name them h_i and j_i (to obtain 6-cycles instead of 4-cycles in the variable-gadget). New variable-gadget is depicted in the Figure 3.10. The e_1 is adjacent to all f_i's and g_i's again, except f_3's and g_3's, and similarly the e_2 is adjacent to all h- and j-vertices. The clause gadget is

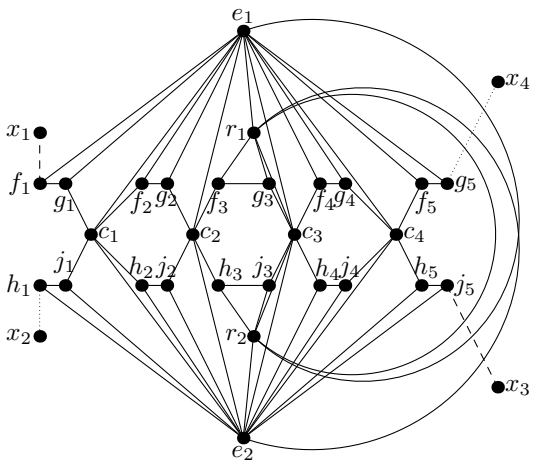


Figure 3.10: The variable-gadget for graphs of interval filaments is similar to the gadget for PC-graphs, only C_4's are subdivided into C_6's. Again dashed lines apply only to the first variable while the dotted to the last.

created similarly as for PC-graphs, both new polygons P_{f_3} and P_{g_3} (or P_{h_3} and P_{j_3}) are intersected instead of P_{a_3}'s (or P_{b_3}'s, respectively). Note that for a satisfiable formula this graph is a PC-graph. Now we show that for an unsatisfiable formula such a graph does not even have an IFA-representation. The argument is similar to the case of PC-graphs, there are just more cases to analyze.

Observation 35 *Filaments representing f- and j- (and similarly g- and h-) vertices lie consecutively on the base-line.*

This observation is almost obvious. If it does not hold, there is a triple F_x, F_y, F_z such that F_x is covered by F_y against P_z. But for each possible triple in G exists a path connecting x with z avoiding neighborhood of y (corresponding to arc-connected curve from F_x to F_z), either we traverse directly inside the variable gadget, or we traverse to the neighboring gadget and use "a bridge" of c_i's. But F_y together with the base-line form a Jordan circle, F_x is (without loss of generality) inside, F_z outside, a contradiction.

We generalize the notion of consecutive occurrences to *circularly consecutive*:

Definition 36 *For a set of disjoint filaments $F_0, ...F_n$ lying consecutively on the base-line we say that this set is* circularly consecutive *if there exists $i \in \{0, ..., n\}$ such that for all $j \neq i$ interval I_j precedes interval I_{j+1} (*i.e.*, its right endpoint is to the left of the left endpoint of I_{j+1}) where $j + 1$ is*

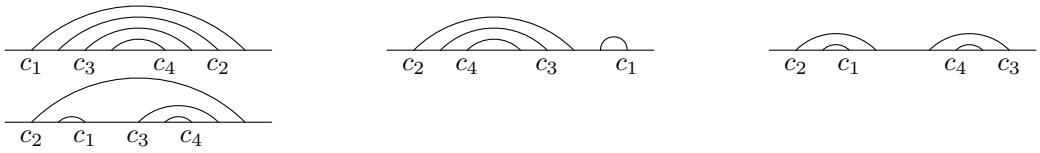


Figure 3.11: These are (up to symmetries) the only possibile representations of vertices c_1, c_2, c_3 and c_4 of the variable gadget.

considered modulo $n+1$ *and* I_i *either contains all other intervals or* I_i *precedes* I_{i+1}.

Lemma 37 *To represent a variable-gadget up to symmetry the c-vertices must occur as depicted in the Figure 3.11.*

Proof. By simple case analysis. ♠

We prove following Lemma similar to Lemma 26:

Lemma 38 *Filaments representing* $f_1, f_2, f_4, f_5, j_5, j_4, j_2, j_1$ *are circularly consecutive.*

Proof. By case-analysis: Again we start observing that relevant f- and j-filaments must not interlace (otherwise e_1 and e_2 irrepresentable) and we check that in none possibility described in the Figure 3.11 this assumption can be violated unless the sub-representation cannot be extended. We start by exploring possible placements of pairs c_i, c_{i+1}. We observe that when $I_{c_i} \subset I_{c_{i+1}}$ endpoints of F_{c_i} are covered by Jordan circle consisting of (possibly parts of) $F_{f_{i+1}}, F_{g_{i+1}}$ and $I_{f_{i+1}} \cup I_{g_{i+1}}$ one and $F_{h_{i+1}}, F_{j_{i+1}}$ and $I_{h_{i+1}}, I_{j_{i+1}}$. For case when $I_{c_i} \cap I_{c_{i+1}} = \emptyset$ this is true for both F_{c_i} and $F_{c_{i+1}}$. This observation proves Lemma for cases 2, 3 and 4 of the Figure 3.11. In case 1 it is necessary to note that I_{f_5} cannot lie between I_{f_4} and I_{j_5} (irrepresentability of e_1, e_2) and the Lemma follows. ♠

This Lemma finishes the proof of NP-hardness of IFA-graphs.

As the construction for IFA-graphs produces either a PC-graph, or a graph that is not an IFA-graph, we proved:

Theorem 39 *No polynomially recognizable class can be sandwiched between PC-graphs and IFA-graphs (*i.e.*,* $\mathcal{C}$ *such that* $PC \subseteq \mathcal{C} \subseteq IFA$*), unless P=NP.*

Proof. By contradiction. We show that existence of such a problem implies P=NP. Consider such a polynomially recognizable class $\mathcal{C}$. Then we may

consider its polynomial recognition algorithm. But when we use the reduction for IFA-graphs (which produces either PC-graph or not IFA-graph), this polynomial algorithm would be able to solve E3-PURE-NAE-SAT, which is NP-hard and therefore P=NP. ♠

Chapter 4

Graphs with large girth

4.1 PC-graphs

Note that in this section we use terminology defined in subsections 2.1 and 3.2. Let us recall that a graph $G = (\{v_1, \ldots, v_n\}, E)$ is a PC-graph if and only if there exists an alternating sequence S of letters $v_1, \ldots, v_n$ such that $v_i v_j \in E(G)$ if and only if v_i and v_j alternate in S.

In general, visibility of polygons is representation dependent and two representations with the same alternating sequence may differ in visibilities from one point (see Fig. 4.1). This is not the case for K_3-free graphs:

Lemma 40 *Let S be an alternating sequence containing one occurrence of a special symbol x (that corresponds to a 1-gon in any respective PC-representation) and let R and R' be two PC-representations with alternating sequence S. If S (and thus R and R') represent a K_3-free graph, any polygon that is visible*

Figure 4.1: An example of two representations of the same graph with the same alternating sequence. In the first one, the polygon a is visible from x, but not in the second one.

from x in R is also visible from x in R', and vice versa.

Proof. By contradiction. Consider two representations R and R' of a K_3-free graph G. Let R and R' provide the same alternating sequence and let there be a point x and a polygon A such that in R the polygon A is visible from x and in R' not (note that by a point x we refer to a place between two consecutive symbols in respective alternating sequences). Consider individual polygons that block visibility in R'. All of them must intersect with A (otherwise such a polygon blocks visibility in R, too). Consider the last polygon B that has some corner (without loss of generality) "to the right" from x (see Figure 3). We know that B intersects with A (it must have all corners on the bounding circle and must not cover A against point x, as it would do so in R). Then the next one C must have at least one corner "to the left" from x, moreover, it intersects with B (because on the "envelope" it is the next one) and by the same argumentation as for B it intersects with A. Thus we have a triangle in G and a contradiction with the assumption that G is K_3-free.

We conclude this introductory subsection by making two observations regarding the connectivity of minimal non-PC-graphs.

Lemma 41 *A graph G is a PC-graph if and only if all its connected components are PC-graphs.*

Proof. We take alternating representations of individual components and place them one after another.

Lemma 42 *A graph G is a PC-graph if and only if all its biconnected components are PC-graphs.*

Proof. Again we use alternating representations. Let H and J be induced subgraphs of G sharing an articulation m. We rotate the alternating representations of H and J in such a way to obtain one occurrence of m at the beginning of the sequence representing J and one at the end of sequence representing H. We concatenate the representation of J after the representation of H. An inductive application of this argument proves the lemma.

4.1.1 Decompositions

In this section we introduce technical notions and lemmas leading to the recognition algorithm.

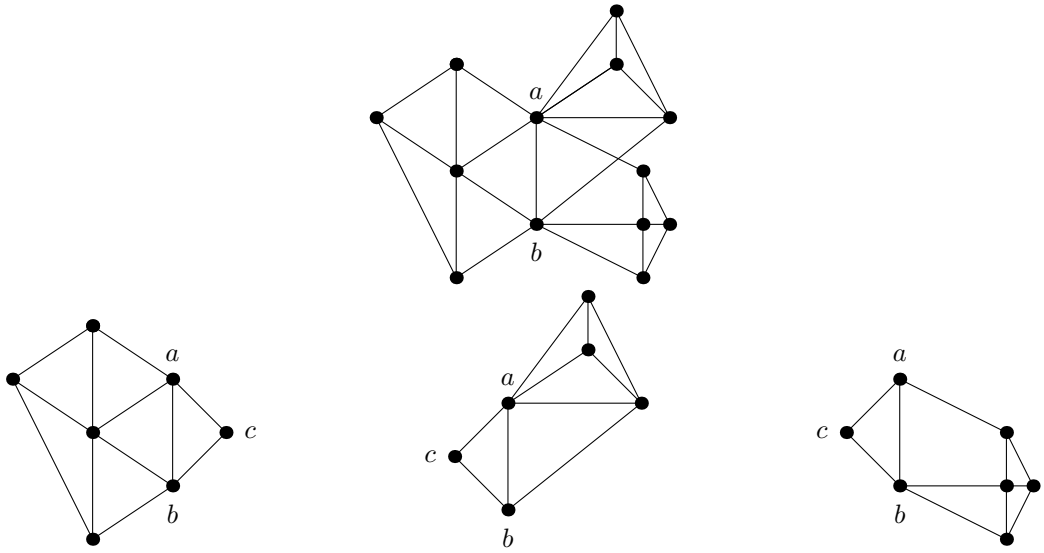


Figure 4.2: An example of the pair-cutting decomposition. At the top of the figure is the original graph with vertices a and b forming a cutting edge. Below are depicted the three elements of its pair-cutting decomposition based on ab. Note that the left one and the right one are pc-primes, but the middle one is not

Definition 43 *Let $G = (V, E)$ be a graph. An edge ab whose endpoints form a cut in G is called a* cutting edge. *Let $C_1, C_2, ..., C_k$ be the connected components of $G[V \setminus \{a, b\}]$ for a cutting edge ab. The* pair-cutting decomposition of G based on *ab is the collection of graphs $G_i^{ab} = (C_i \cup \{a, b, c\}, E(G[C_i \cup \{a, b\}] \cup \{ac, bc\}$, where c is a new extra vertex, $i = 1, 2, \ldots, k$.*

A graph is called pc-prime *(pair-cutting-prime) if it has no nontrivial pair-cutting decomposition (*i.e.*, if every cutting edge divides the graph into one connected component and a single vertex).*

Proposition 44 *A biconnected graph G with a cutting edge ab is a PC-graph if and only if all the graphs in the pair-cutting decomposition based on ab are PC-graphs.*

Proof. We start with the alternating representations $S_1, ...S_k$ of the graphs $G_1^{ab}, ...G_k^{ab}$ in the pair-cutting decomposition. Without loss of generality we may assume that each of them begins with the newly added vertex c. Let $\tilde{S}_i$ be obtained from S_i by removing all occurrences of c and adding an occurrence of a to the beginning and an occurrence of b to the end of the sequence. Then the concatenation $\tilde{S}_1 \ldots \tilde{S}_k$ is an alternating representation of G.

Obviously a and b alternate. By our transformation we could not lose any alternation among vertices of G. Thus it suffices to check that we did not add new ones. For $i \neq j$, any symbol from $\tilde{S}_i$ distinct from a and b cannot alternate with any symbol from $\tilde{S}_j$ (distinct from a and b). Suppose that by replacing the first occurrence of c by a we have created a new alternation of a and $x \neq b$ in $\tilde{S}_i$. Thus in S_i all occurrences of a lie between two occurrences of x. But a and c alternated in S_i, thus between these two occurrences of x there is an occurrence of c and thus x and c alternated. Hence $x = b$ (c alternated only with a and b), which is a contradiction.

The converse is clear from the fact that the class of PC-graphs is closed under taking induced minors and the fact that each G_i^{ab} is an induced minor of G. Indeed, for some $j \neq i$, contract all vertices of C_j into a single vertex c and contract all vertices of all remaining C_l's for $l \neq i, j$ into the vertex a. Since G was biconnected and vertices a and b formed a cut, c is adjacent exactly to a and b.

The previous proposition could be a corner stone of a recognition algorithm for PC-graphs. However, presently we are only able to decide on graphs of large girth, and then the reduction is inconvenient because it creates short cycles. For this sake we introduce the following adjustment. In each element of the pair-cutting decomposition $G[V(C_i) \cup \{a, b\}]$ we mark the edge ab "red" instead of adding the triangle abc. We will talk about the *red-edged decomposition* and each element of this decomposition will be called a *red-edged graph*. It is not true anymore that a biconnected graph is in PC if and only if each red-edged component of the decomposition is a PC-graph. However, we can still control how to glue representations together if G has no short cycles.

In the sense of Definition 43 we denote by G^e the graph obtained from G by adding a new vertex adjacent to the end-vertices of e. Similarly, $G^{e_1,\dots e_k}$ is the graph obtained by adding k new vertices, each connected to the endpoints of one of the edges e_i.

Observation 45 *Let R be an alternating representation of a graph G . If this representation can be extended to a representation of G^{ab}, it can be done so by adding the subsequence $cabc$ between some two consecutive symbols in R.*

Proof. Consider an alternating representation S of G^{ab} which is an extension of R. First we remove all new occurrences of symbols distinct from a, b and c (if any). Still we have a representation of G^{ab} which extends R. Now we

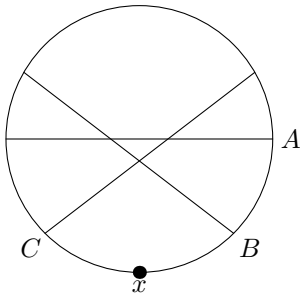


Figure 4.3: Polygon B has one corner to the right from x, polygon C to the left.

reduce the occurrences of c's. We greedily try to remove them one by one. For each of them, if c still alternates with both a and b, the occurrence is removed. Note that after we process all occurrences of c, exactly two of them remain in the representation. (Since c alternates with a, b, at least two occurrences must remain. A simple case analysis shows that among more than two occurrences of c at least one is superfluous.) Since in this step we removed only occurrences of c, no alternations were lost.

Now we replace the rightmost occurrence of c with $cabc$ and denote this alternating sequence $\tilde{S}$. We show that no new alternations appear. This is obvious for alternations of any pair of symbols distinct from a and b. Suppose a alternates with some $x \neq b$ (for alternation of b and x, the argument is analogous). We know that $\tilde{S}$ contains a subsequence $...c...a...cabc...$ and to create a new alternation of x with a by adding $cabc$, the newly added a must appear between two occurrences of x. But then these occurrences of x surround the rightmost occurrence of c and thus c alternates with x in S. Hence $x = b$, a contradiction. Finally, we remove the leftmost occurrence of c and all new occurrences of a and b (with respect to R) except of $cabc$ and we obtain the desired extension of R. Indeed, c alternates with both a and b since both a and b occurred in R.

Proposition 46 *Let G be a connected red-edged graph of girth at least 4 with $Q = \{e_1, ...e_k\}$ being the set of its red edges. Let R be a PC-representation of G. If for any $e \in Q$, the representation R can be extended into a representation of G^e, then R can be extended into a representation of $G^{e_1,...e_k}$.*

Proof. We prove the statement by induction on the number of red edges. Of course, the statement is obvious if G has just one red edge. Otherwise, consider $e_1 = ab$.

Using Observation 45 we extend R to R' by adding $cabc$ to represent G^{ab}, but then (in order to eliminate the triangle abc) we replace the newly added $cabc$ by $c_1ac_2c_1bc_2$. In this way we obtain a representation R^* of the graph $G^* = (V(G) \cup \{c_1, c_2\}, E(G) \cup \{ac_1, c_1c_2, c_2b\})$, and clearly R^* extends R.

The key observation (proved below) is that for any other red edge uv, R^* can be extended to a representation of $(G^*)^{uv}$. Since G^* is K_3-free and has smaller number of red edges than G, the induction hypothesis implies that R^* can be extended to a representation $\tilde{R}$ of $(G^*)^{e_2,\dots e_k}$. The desired representation of $G^{e_1,\dots e_k}$ extending R is now obtained by renaming $c_1 = c_2 = c$, which corresponds to contraction of the edge c_1c_2 into vertex c.

Key observation: Let R be a representation of a connected K_3-free graph G with red edges ab and uv. If R can be extended to R' representing G^{ab} and to R'' representing G^{uv}, then R can be extended to $\tilde{R}$ representing $G^{ab,uv}$.

Without loss of generality consider that R' is obtained by adding $cabc$ to the beginning of R. Note that since each subsequence $cabc$ or $wuvw$ can be added separately, problems may be caused only by unwanted alternations between some symbol of a, b and some of u, v. We analyze two cases. Either R'' is also created by placing $wuvw$ to the beginning of R, or not.

Case 1: R'' is $wuvwR$. In this case we without loss of generality consider that a and u do not alternate in R (a cannot alternate with both u and v as G is K_3-free) and that all occurrences of a are preceded by all occurrences of u (either this holds or we reverse the sequence, if none of these sequences are good, R contains $...u...a...u...$ or $...a...u...a...$ and R' or R'' would not represent G^{ab} or G^{uv}, respectively).

Let $\tilde{R} = cabcwuvwR$. By the assumptions on occurrences of a and u in R, a and u do not alternate in $\tilde{R}$. Suppose a and v do not alternate in R, we show that they do not alternate in $\tilde{R}$ either (and similar arguments apply to alternations of b and u, and also of b and v). Since v alternates with u in R, v occurs at least once before the last occurrence of u and hence all occurrences of v lie before the first occurrence of a (otherwise again R' or R'' do not represent G^{ab} or G^{uv}, resp.). Hence v and a do not alternate in $\tilde{R}$.

Case 2: Let $R' = cabcR$ and let R_1, R_2 be nonempty sequences such that $R'' = R_1wuvwR_2$ (and hence $R = R_1R_2$). Let $\tilde{R} = cabcR_1wuvwR_2$ (*i.e.*, the smallest common extension of R' and R''). We continue by contradiction. Suppose without loss of generality that a and u alternate in $\tilde{R}$ but not in R (and hence neither in R' and R''). We claim then either R_1 or R_2 contains neither u nor a. At least one of R_1 and R_2 contains both a and u, and assume without loss of generality that both symbols appear in R_2. Again we analyze two cases:

a) Both a and u appear in both R_1 and R_2. To avoid alternation in R' and R'', no a may appear after the first u in R_1, thus in R_1 all a's lie before all u's. The same argumentation implies that in R_2, all a's lie after the last u. So

$$\tilde{R} = cabc...a...a...u...u...wuvw...u...u...a...a....$$

and hence u and a do not alternate in $\tilde{R}$.

b) The sequence R_1 only one of the symbols, say a (for u the argument is similar). We know that the new alternation in $\tilde{R}$ uses the first occurrence of a (in $cabc$) and the first occurrence of u (in $wuvw$), otherwise a and u would alternate in R' or R''. So $\tilde{R}$ contains a subsequence $caR_1wu...a...u...$ (since there are no u's in R_1). But we know that R_1 contains at least one occurrence of a and hence a and u alternate in R''.

Therefore R_1 contains no a's and u's.

Similarly as we argued before, all occurrences of a must preceded all occurrences of a in R_2 (because a and u alternate in $\tilde{R}$ but not in R' nor in R''). Thus $\tilde{R}$ contains subsequence

$$cabcR_1wuvw...a...u...$$

Because the graph G is connected, at least one symbol x occurring in R_1 also occurs in R_2 (otherwise the symbols in R_1 would form connected components).

Depending on the position of occurrences of x in R_2, x alternates with a or u in $\tilde{R}$ and hence also in R (if $x \neq b, v$, then this follows from the fact that x alternates with the same symbols in $\tilde{R}$ as in R, and if $x = b$ or $x = v$ then x alternates with a or with u, respectively, by default).

If x alternates with u in R, we know that at least one its occurrence is after the last occurrence of a in R_2 (because all u's are after the last a) and thus x alternates with a in R' and therefore in R, too. Analogously, when x alternates with a in R, it alternates with u as well. Thus x alternates with both a and u in R and hence $\tilde{R}$ contains a subsequence

$$cabc...x...wuvw...x...a...u...x...$$

The rightmost a and u refer to the last a and to the first u in R_2, without loss of generality let $x \neq v$. We know that at least one occurrence of v is after the first u and hence after the last a in R_2 (otherwise u and v would not alternate in R). Therefore $\tilde{R}$ contains a subsequence

$$cabc...x...wuvw...x...a...u...v$$

and v alternates with x in R'' and thus in R, and we have a triangle xuv in G, a contradiction.

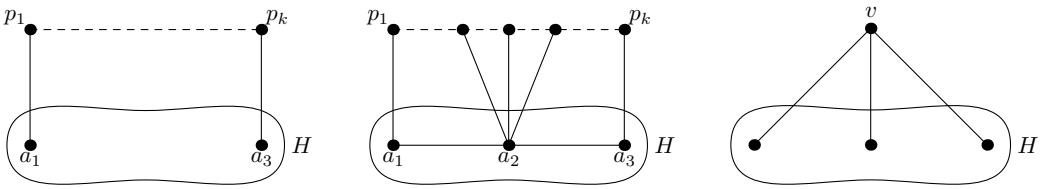


Figure 4.4: The first two pictures show pseudoears of type I and II, the third one is an example of a vertex we operate with in our version of ear-decomposition lemma. Solid lines denote edges, dashed lines represent paths.

4.1.2 Pseudoears

A well-known characterization of biconnected graphs is given by the *ear-decomposition lemma*:

Lemma 47 *A biconnected graph can be constructed from any of its cycles by consecutive addition of paths (with endpoints in the already constructed subgraph, so called* ears*) and/or single edges.*

However, we need a version where the constructed graph is an induced subgraph, and therefore we need to avoid adding edges. Thus we define the technical notion of a pseudoear and introduce a special version of this lemma for K_3-free graphs.

Definition 48 *Let H be an induced subgraph of a graph G, and let $U = a_1 \dots a_3$ be an induced path in H of length at least 2. A* pseudoear attached along *U is an induced path $P = p_1 ... p_k$ in $G \setminus H$ of length at least 1 such that either*

I. H has length at least 3 and $E(P, H) = \{p_1a_1, p_ka_3\}$, or

II. $U = a_1a_2a_3$ has length 2 and $E(p_1, H) = \{p_1a_1\}$, $E(p_k, H) = \{p_ka_3\}$, and $E(\{p_2, \dots, p_{k-1}\}, H) \subseteq \{p_ia_2 | i = 2, \dots, k-1\}$.

Lemma 49 *Any biconnected K_3-free graph without cutting edges can be constructed from any of its induced cycles by consecutive addition of pseudoears and/or single vertices adjacent to at least two vertices of the so far constructed subgraph. On the other hand, any K_3-free graph constructed in this way is biconnected and has no cutting edges.*

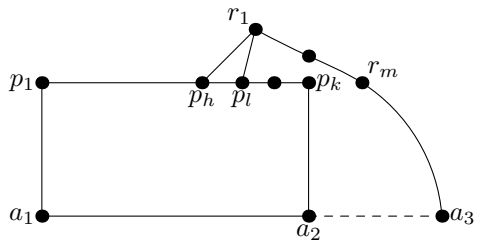


Figure 4.5: Illustration to the proof of Lemma 49.

Proof. Since pseudoears are special types of ears and addition of single vertices can be composed from addition of one ear and several edges, the biconnectedness of the constructed graph follows from the classical ear-decomposition lemma. The nonexistence of cutting edges follows from the assumed K_3-freeness. Hence the key part of the lemma is its first (constructive) part and we proceed with the proof of this one.

Suppose the statement is not true and consider a smallest counterexample G. Let H be a largest (induced) subgraph of G that can be created by adding pseudoears and/or single vertices as stated in the lemma. Then each vertex in $G \setminus H$ has at most one neighbor in G, otherwise we have a contradiction with the maximality of H. Let us consider a shortest path $P = p_1p_2...p_k$ on vertices of $G \setminus H$ such that this path has exactly two distinct neighbors a_1 and a_2 in H (such a path exists since G is biconnected, *e.g.*, each shortest ear from the classical ear-decomposition has this property). Obviously, p_1a_1 and p_ka_2 are the only edges between P and H. If a_1a_2 is not an edge in G, $p_1...p_k$ is a pseudoear (attached along any induced path $a_1...a_2$) and H was not the largest possible subgraph, a contradiction. If a_1a_2 do form an edge in G, we know (because G has no cutting edge) that there exists a path R connecting P with $H \setminus \{a_1, a_2\}$. Consider a shortest possible $R = p_lr_1...r_ma_3$. Observe that a_3 is not adjacent to both a_1 and a_2 (G is K_3-free), but r_1 can be adjacent to several vertices of P. Without loss of generality assume $a_3a_1 \notin E(G)$.

Since H is biconnected, a_2 lies on a path $a_1a_2...a_3$. Because of the minimality of R, all neighbors of $r_1, ..., r_{m-1}$ in H are among $\{a_1, a_2\}$ and a_3 is the only neighbor of r_m in H. Suppose first that $a_3a_2 \notin E(G)$, *i.e.*, the path $a_1a_2 \dots a_3$ has length at least three. If any r_i is adjacent to a_2, we take the largest such i and note that the path $r_ir_{i+1}...r_m$ is a pseudoear of type I attached along $a_2...a_3$. If this is not the case, but some r_i is adjacent to a_1, we take the largest such i and note that the path $r_ir_{i+1}...r_m$ is a pseudoear of type I attached along $a_1a_2...a_3$. Otherwise, let h be the smallest index such

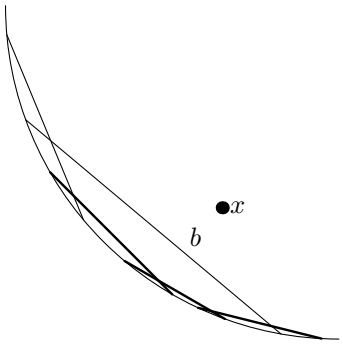


Figure 4.6: Demonstration of a jammed polygon – b is jammed with respect to x.

that $r_1p_h \in E_G$ (clearly $h \leq l$), then $p_1...p_hr_1...r_m$ is a pseudoear of type I attached along $a_1a_2...a_3$. Each of these cases contradicts the assumption of maximality of H.

If $a_2a_3 \in E(G)$, the argumentation is similar, only we aim at pseudoears of type II. If any r_i is adjacent to a_1, we take the largest such i and note that the path $r_ir_{i+1}...r_m$ is a pseudoear of type II attached along $a_1a_2a_3$. Otherwise, let h be the smallest index such that $r_1p_h \in E_G$ (clearly $h \leq l$), then $p_1...p_hr_1...r_m$ is a pseudoear of type II attached along $a_1a_2a_3$.

Each of these cases contradicts the assumption of maximality of H.

4.1.3 Jammed polygons

Definition 50 *A polygon P in a PC-representation is* jammed *with respect to a point x outside P (and inside the bounding circle or on its boundary) if any straight ray starting in x which intersects P also intersects some other polygon of the representation.*

Lemma 51 *Let $P = b...c$ be an induced path of length at least 2 in a biconnected PC-graph G. Then for all PC-representations of G, all polygons representing the inner vertices of P are jammed with respect to any point x which lies on the bounding circle and from which all polygons representing the vertices of P are visible.*

Proof. Consider a given PC-representation R of G. Let x be the point on a bounding circle from which all polygons representing the vertices of P are visible. Then all polygons representing the vertices of P lie around the circle (otherwise no such point x exists) and moreover, the situation looks

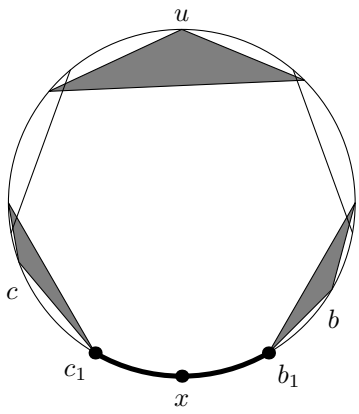


Figure 4.7: An illustration to the proof of Lemma 51.

as depicted in Figure 4.7, *i.e.*, if b_1 and c_1 are the corners of P_b and P_c (resp.) closest to x, then all polygons representing P lie outside the arc b_1xc_1. Consider $u \in P \setminus \{b, c\}$. As G is biconnected, $G' = G \setminus \{u\}$ is connected. Consider a shortest path $Q = b...c$ in G'. The representation of G' obtained from R by removing P_u contains a representation of a path Q connecting P_b and P_c.

By the x-envelope of R_Q (representation R restricted to Q) we mean the set of points $z \in \bigcup_{y \in Q} P_y$ such that the open segment xz does not intersect any polygon of the representation. The x-envelope of R_Q is a Jordan curve going from b_1 to c_1. Since P_u lies outside the arc b_1xc_1, any ray from x intersecting P_u has to intersect the x-envelope of R_Q and hence also some polygon representing a vertex from Q. Thus P_u is jammed with respect to x.

A *sector* is the arc of the bounding circle between any two consecutive corners. Analogously, in an alternating sequence it refers to the place between two consecutive symbols.

Observation 52 *If a polygon is jammed with respect to one point in a sector, then it is jammed with respect to all points of this sector.* ♠

Lemma 53 *If a polygon P_v is jammed with respect to some point in a sector xy in a representation R of a biconnected graph G and if G' is obtained from G by pending a vertex u of degree one on v, then in every representation of G' which extends R and has at least one corner of P_u in the sector xy, all corners of P_u are in the sector xy and P_v must get a new corner there.* ♠

Lemma 54 *Every biconnected pc-prime graph with girth at least 5 has at most 1 minimal PC-representation.*

Proof. By induction on the number of pseudoears added in the construction of the graph. The first step is obvious: C_k with $k \geq 5$, can be represented only by bigons lying around the bounding circle, and such representation is up to symmetries, unique.

Now consider a graph G obtained from H by adding a pseudoear or a single vertex. Consider a minimal PC-representation of G. When we remove polygons representing vertices of the pseudoear (or the single vertex), from the induction hypothesis we obtain an extension of the unique minimal PC-representation of H. We take this minimal representation R, and investigate how to extend it into a representation of G:

1. Adding a pseudoear of type I. The pseudoear, say $p_1, ...p_k$, consists of at least two vertices adjacent to exactly two vertices a_1 and a_3 in H having (in H) distance at least 3. We need a sector, into which both P_{a_1} and P_{a_3} can be extended (*i.e.*, to add a corner) without adding unwanted intersections. Such a sector is uniquely determined, as otherwise H is not biconnected. (Indeed, if both P_{a_1} and P_{a_3} can be extended into two different sectors, let a_{11} and a_{12} be two possible new corners of P_{a_1}. The same points can be used by P_{a_3}. Thus we know that the segment $a_{11}a_{12}$ intersects only polygons representing common neighbors of a_1 and a_3. Since H is C_4-free, there is at most one such common neighbor and H is not biconnected.)

   Let the unique sector be (in terms of alternating sequence) $...xy...$ and let $...xa_1a_3y...$ be an extension of R to a representation of H (note that exactly one of $...xa_1a_3y...$ and $xa_3a_1y...$ is a valid representation of H). We first extend the representation to

   $$...xp_2p_1p_3p_2p_4p_3...p_kp_{k-1}y...$$

   which is the only minimal way to represent the path $p_1...p_k$. If $x \neq a_1$, change the subsequence $...xp_2p_1...$ to $...xp_1a_1p_2p_1...$, otherwise (*i.e.*, when $x = a_1$) change it to $...p_1xp_2p_1...$ and analogously for y and a_3. It is not difficult to see that this is the unique minimal extension of R. In the first case apply Lemma 51 to (any) xy-path in G. Since a_1 (or a_3) is jammed in R, it is necessary to extend it into the xy sector by Lemma 53. In the second case again the choice is unique, otherwise again the graph H was not biconnected.

2. Adding a pseudoear of type II. Again we find a unique sector into which a_1, a_2 and a_3 can be extended. At the first sight it might appear to be sufficient to find two sectors, one that can extend a_1 and a_2, second one that can extend a_2 and a_3, but in this case when representing some p_i, we split H into non-trivial blocks. Because of Lemma 51 P_{a_2} is jammed. We behave like in the first case, only those p_i's adjacent to a_2 will be interlaced by a_2. Each p_i is necessary to intersect by a_2 separately and it is necessary to extend a_2 (because P_{a_2} is jammed, because it is necessary to represent all p_i's around the circle and because H is K_3-free).

3. Adding a single vertex v. For each triple of its neighbors we check whether they lie around the circle. If a is covered by b against c, we may omit intersection of P_v with b as it will be intersected as a side-effect of intersecting a and c. Now we have set of polygons lying around the circle that should be intersected by P_v.

   Now we restrict H to neighbors of v represented around the circle C and non-neighbors of v, set N. If I is representable, no non-neighbor of v covers one its neighbor against another. Given one correct corner of v and one polygon P_c (for $c \in C$) there are three possibilities how to ensure intersection of P_v and P_c. Either we immediately cross P_c by P_v or we extend P_c and cross it by P_v together with some other P_d (for $d \in C$) or we extend P_v and add a corner of P_v to intersect only this polygon. For each polygon there is possible only one of these three possibilities. In all cases, wherever we represent the new corner of P_c, we can represent the new corner of P_v in the same place. Also note that in the first case, wherever we represent the new corner of P_v, we can represent the corner of P_c in the same place. Assume that there are two possibilities for representing the respective corner of P_c in the second and in the third case and two possibilities for representing P_v in the first case. Then we connect these two endpoints by a segment and we know that this segment intersects only P_c and thus divides the representation into two blocks which contradicts the biconnectivity of H (this segment intersects only polygons in the intersection of neighborhood of c and neighborhood of v (which is empty) and from the graph H only polygon representing c. Note that the proof is invariant under choice of "the correct" corner of P_v. Thus for the algorithm we start by brute force choosing a polygon representing fixed $c_1 \in C$, try all possibilities for "the correct" placement of one corner of P_v. This step does not rely on the ordering of neighbors, thus we add two symbols to the alternating sequence at most n times in total, which is implementable in $O(n^5)$.

4.1.4 Algorithms

Algorithm 1:

Input: A biconnected graph G
Output: Decision whether G has a PC-representation.
Auxiliary variables: $\mathcal{H}$ – set of graphs, $\mathcal{S}$ – set of representations, initially empty

Add G to set $\mathcal{H}$.
`while` exists $F \in \mathcal{H}$ with a cutting-edge ab `do`
    remove F from $\mathcal{H}$ and replace it by the elements of the red-edged decomposition of F with respect to ab.
`done`
`forall` $F \in \mathcal{H}$ `do`
    `if` Algorithm2(F)='false' `then` return 'false'
    `else`
        add the representation of F provided by
        Algorithm2(F) into $\mathcal{S}$
`done`
`forall` $R \in \mathcal{S}$ `do`
    `for` $e \in$ red_edges_of(graph_of(R)) `do`
        `if` Algorithm3(R,e)='false' `then` return 'false';
return 'true';

The correctness of Algorithm 1 follows from Lemmas 41, 42 and Propositions 44, 46. Now we present the algorithm for finding a representation of a pc-prime graph as the rest is either brute-force or obvious (and was described above).

In the Algorithm 2 we call an arc of a bounding circle between two corners of polygon as a *sector*.

Algorithm 2:

Input: pc-prime graph G.
Output: PC-representation or `false`.
Auxiliary variables: Graph H, initially empty.

`while` ($|V(H)| < |V(G)|$) `do`
    find a pseudoear or a single vertex P attachable to H;
    by brute force represent it;
    `if` the representation is impossible `then` return false;
    add P to H;
`done`

The algorithm for finding pseudoears (or single vertices) implements the (constructive) proof of Lemma 49: finding a suitable representation follows the proof of Lemma 54. Note that looking for a pseudoear or a single vertex is (naively) possible in $O(n^2)$, and representing a pseudoear is possible in $O(n^4)$ (looking for feasible sector, trying different placements of p_1 and p_k). Note that we are adding pseudoear (or a single vertex) only linearly many times. Thus the algorithm is polynomial.

Algorithm 3:

Input: Alternating representation $\mathcal{R}$ of a graph G and a red edge $ab = e$ in G (c is not a vertex of G)
Output: Decision whether G_e has PC-representation.

1. By brute force try to add *cabc* between all consecutive pairs of symbols into alternating representation and check whether we obtain correct representation. If we at least once succeed, `return representation;`

2. `return` 'false'

The algorithm is just brute force, but polynomial. The correctness is obvious from Observation 45.

Our algorithm works only for graphs of girth at least 5. Most of claims is proven for K_3-free graphs, but for graphs of girth 4 we have no equivalent of Lemma 54. Moreover, we can construct graphs of girth 4 with exponentially many minimal representations. Such a graph is depicted in Figure 4.8. Thus the recognition problem of K_3-free PC-graphs remains open.

4.2 SEG- and PSEG-graphs

In this section meet two questions. The recognition problem for graphs with large girth with the sandwiching problem.

Theorem 55 *For all k given a graph G with girth at least k, it is NP-hard to decide whether G is a SEG-graph. The same holds for PSEG-graphs.*

Proof. We use construction similar to [33]. This time we reduce P3CON-E3-SAT(4). For a formula Φ and an appropriate planar embedding of corresponding graph $G(\Phi)$, we construct a graph H with girth at least k such that whenever H is a SEG-graph, Φ is satisfiable while non-satisfiable Φ produces not even PSEG-graph.

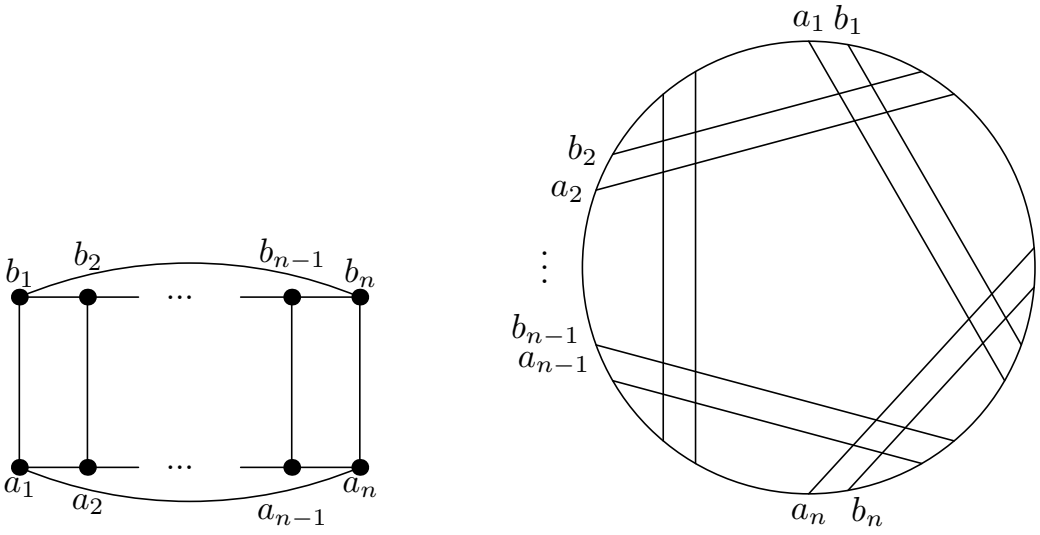


Figure 4.8: Construction of graphs with girth 4 having exponentially many minimal PC-representations. Note that pairs a_i, b_i can be switched in the representation.

We modify the graph $G(\Phi)$ in the following way: Each vertex corresponding to a variable we replace by a circle of length $\max\{16, k\}$. Each edge of $G(\Phi)$ will be replaced with pair of long-enough paths containing a gadget called cross-over. Each vertex corresponding to a clause gets replaced by a clause gadget. The assignment to variables we obtain similarly to a section 3.1 from the orientation of the circle representing particular (variable) vertex. The 'orientation' of pair of paths from the vertex gadget to a clause gadget describes whether respective occurrence in a clause is positive or negative. By 'orientation' of a pair of paths we mean that one path is "to the left" to the other. This is a standard trick that appears, *e.g.*, also in [33].

From each variable gadget (forming a cycle) $v_1, d_1, v_2, d_2 \ldots, v_8, d_8 \ldots$ for each occurrence of this variable we place two paths. If the occurrence is positive, then the first path starts in v_{2i-1}, the second in v_{2i}. If the occurrence is negated, we change this ordering (the first path starts in v_{2i}, the second in v_{2i-1}). Vertices denoted by d_i are dummy vertices forcing the SEG-representation to place the whole segment v_i before (resp. after) v_j when running on the boundary of segments representing the variable gadget. For the sake of simplicity, one path will be called the 'left' path, one the 'right' path. As there is given ordering in which these pairs of paths must start from the variable gadget, and as the assignment is given by the orientation of the cycle (variable gadget), *i.e.*, when passing around the cycle having the interior by the left-hand-side, vertices appear in ordering v_1, v_2, v_3 or v_3, v_2, v_1

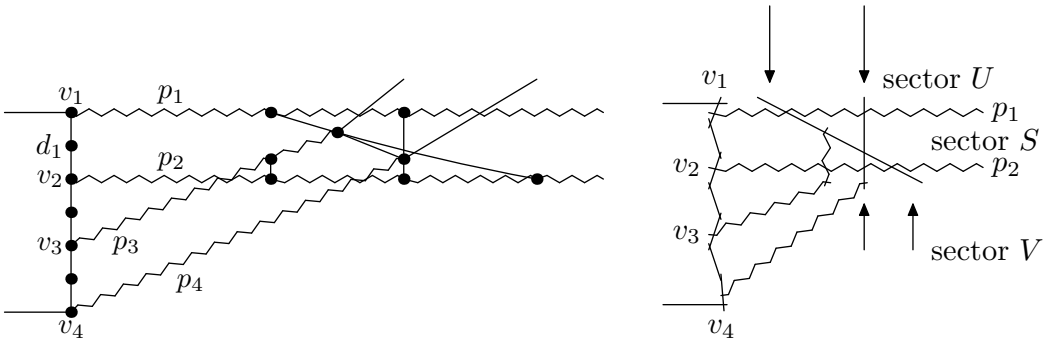


Figure 4.9: Cross-over gadget and its representation starting from the variable gadget. Broken line denotes sequence of arbitrarily many segments (depending on required girth). Arrows in the representation offer two possibilities where to continue.

on this circle. Thus it is necessary to ensure that respective occurrences of the pairs of paths will appear in the correct ordering. We do so again by placing the **cross-over** gadget between the first and the second and then between the third and the fourth pair of the paths.

This cross-over allows two pairs of paths either to cross, or to touch only. Up to now, the cross-over gadgets that were used, were independent on many factors: In particular it was not necessary to know which path from the pair is on the left and which on the right. Two pairs simply either crossed, or touched only. These gadgets allowed us to solve non-planar versions of the problem SAT-problem, but unfortunately, they were based on C_4 occurrences. Our new cross-over needs to know which path is "on the left" and which one is "on the right", but this is OK when reducing P3CON3SAT(4).

The new cross-over gadget is depicted in the Figure 4.9. It is possible to either cross two paths or touch only. For the following claim it is necessary to say that the clause-gadget will be surrounded by a circle. Both paths representing one occurrence of one variable are adjacent to a common vertex of this circle surrounding the clause-gadget.

Claim 56 *Cross-over is a SEG-graph. Moreover, if this gadget appears in a graph obtained by our construction, segments x and y have to intersect inside the second pair of paths (like shown in the Figure 4.9).*

Proof. The first part (SEG-representation) is obvious from the Figure. Moreover (pseudo)segments x and y have to intersect each other. They cannot intersect outside S, as the original graph G was 3-connected, sectors U and V are divided by Jordan circle. Because of neighbors of vertex represented

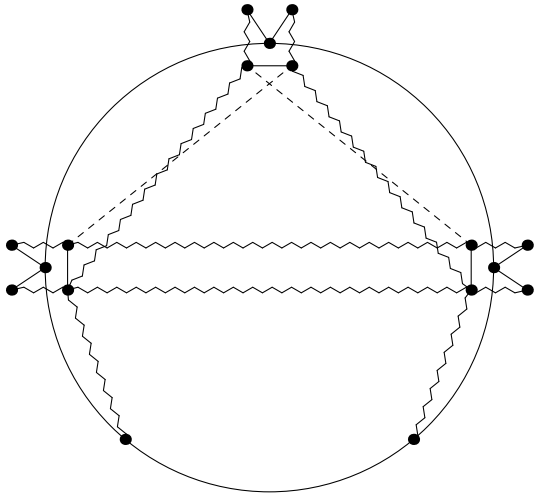


Figure 4.10: Clause gadget

by x, it is impossible to pass from sector U to V otherwise than we show. Now x and y must intersect inside S as y divides sector S into left and right sector. Without loss of generality the left sector creates another Jordan circle that must be crossed by x and must be crossed by y.

Corollary 57 *The left-right orientation of the paths does not change within cross-over.*

Remark: Note that the argumentation so far performed for both, variable- and cross-over-gadget works also for CONV-graphs.

Now we present the clause-gadget. It is depicted in the Figure 4.10. This gadget can be represented by (pseudo)-segments whenever at least one variable is evaluated by true.

Lemma 58 *The clause-gadget has a (pseudo)segment representation if and only if at least one variable is evaluated to TRUE.*

Proof. We proceed using the Figure 4.11. When the top literal is *true,* we represented the paths denoted by dashed lines inside the circle in the gadget. Even if the "top" literal is *false*, at least one dashed line can always be represented inside the circle. Moreover under those circumstances by any *true* "bottom" literal corresponding dashed path can be represented outside the inner cycle.

Theorem 59 *For any fixed k between class of SEG- and PSEG-graphs restricted to graphs with girth at least k no polynomially recognizable class can be sandwiched, unless P=NP.*

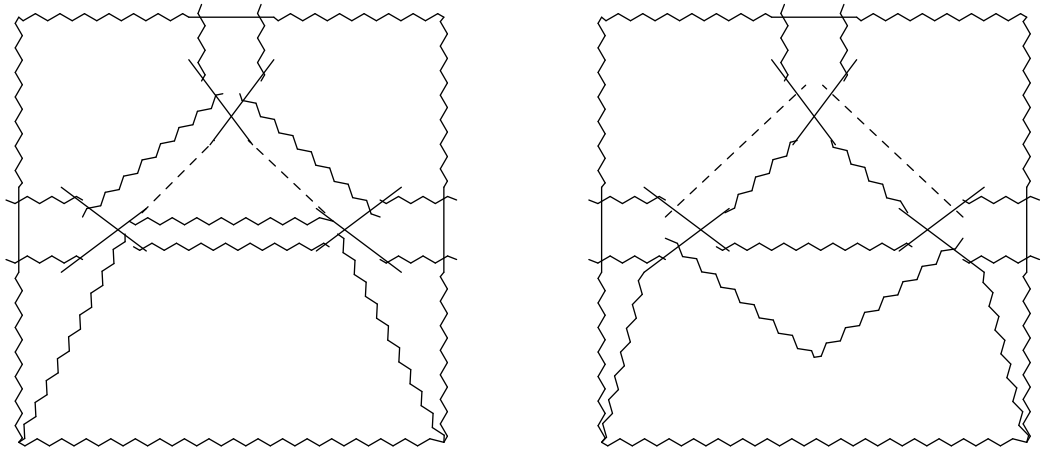


Figure 4.11: Representation of the clause. On the left the top literal evaluated to *true*, on the right the bottom two literals *true*.

Proof. Clear from the fact that satisfiable formulae formed a SEG-graph, while non-satisfiable ones could not be represented even by pseudosegments. If it was possible to sandwich some polynomially recognizable class between SEG and PSEG, an appropriate polynomial recognition algorithm would provide a solution of an NP-complete problem in polynomial time. ♠

Chapter 5

Conclusion and open problems

Despite being explored for years, classes of graphs with geometrical representations provide many open problems that are hard in general graphs. The community is interested in tractability of problems that are NP-hard in general, as well as in even faster algorithms for already polynomially solvable problems. Some of the interesting currently open problems are:

- *Does each planar graph have an intersection representation by homothetic triangles in a plane? (Kaufmann, Kratochvíl, Lehmann, Subramanian in [29])* The article [29] gives several interesting results for graph-optimization problems for this class, on the other hand it shows that the recognition problem is NP-hard and effective algorithm finding such a representation for planar graphs would be helpful. It is known [7] that each planar graph is a contact graph of triangles (triangles do not intersect, each pair shares at most one point), but triangles in the cited article are not homothetic. In [1], it is shown that series-parallel graphs are intersection graphs of homothetic triangles. There is also a conjecture of Felsner and Kratochvíl that each 4-connected planar triangulation has a contact representation by homothetic triangles. A proof of this conjecture would answer the question.

- *Can the problem of maximum clique for segment graphs be solved in polynomial time? (Kratochvíl and Nešetřil in [38])* This question was asked by Kratochvíl and Nešetřil when they proved that this problem can be polynomially solved when segments use only fixed number of directions [38] (such a class is called k-DIR, segment graphs form union of these classes). This problem and state of current knowledge is nicely described in [2].

- *Can subtree-overlap graphs be recognized (represented) in polynomial time?* This question is opened since 2000 when Gavril [18] introduced a polynomial-time algorithm for maximum clique and independent set. Class of subtree-overlap graphs is exactly the class of subtree-filaments [11] and it generalizes many other intersection classes (*e.g.*, class of polygon-circle graphs, circle graphs, circular-arc graphs). Gavril's algorithm for maximum clique and independent set operates with a representation. Recently we proved that the recognition problem for polygon-circle graphs and interval-filament graphs is NP-complete, therefore importance of this question increases, although it is expected to be NP-complete (the converse would be surprising).

- *What is the complexity of recognition of CONV- and string graphs with girth (*i.e.*, length of the shortest cycle) at least 5? (Kratochvíl and Pergel in [40])* For several classes (polygon-circle graphs [40], or pseudodisk graphs – intersection graph of pseudodisks in a plane) it is known that the recognition problem becomes polynomially solvable when we are given a graph without short cycles. Therefore it was believed that the recognition problem gets easier for graphs with high girth. On the other hand we have shown that for segment and pseudosegment graphs this problem remains NP-hard even for graphs of arbitrarily large girth.

- *What is the complexity of recognition of cactus subtree graphs?* A cactus is a graph obtained from a circle by consecutively sticking circles to a vertex of an already constructed cactus. This class obviously generalizes class of chordal and circular-arc graphs. Conversely, it is generalized by class of polygon-circle graphs. As we have shown that polygon-circle graphs are hard to be recognized and as for them some interesting polynomial-time algorithms are known [18], it is important to ask whether cactus-subtree graphs can be recognized efficiently. This class was mentioned by Gavril in several his texts [19, 20, 18].

- *What is the complexity of recognition of bicircular-arc graphs? (Felsner, private communication)* Bicircular-arc graphs are intersection graphs representable by convex objects consisting of convex hull of two circular arcs (on one fixed circle). This question was posed by Stefan Felsner after polygon-circle graphs appeared to be NP-complete. This class generalizes polynomially recognizable class of trapezoid graphs [15] and is generalized by class of polygon-circle graphs. At first this class seemed to behave like 3-PC-graphs, whose recognition is NP-complete but after some attempts it appeared to be much weaker in some sense. This

problem is another attempt to find largest possible polynomially recognizable class for Gavril's algorithms on maximum weighted clique and maximum weighted independent set, after polygon-circle graphs appeared to be hard (as well as graphs of interval filaments). The number of two circular arcs is maximum possible. Our reduction [39] with a slight modification shows hardness of the recognition problem also for intersection graphs of objects obtained as convex hulls of three (or more) circular-arcs (on a common circle).

- *Is the recognition problem in NP for class of segment graphs and for class of intersection graphs of convex polygons? (Kratochvíl and Matoušek in [37])* Both classes are known to be NP-hard to recognize [33], moreover, in [37], it was shown for segment graphs that the Cartesian coordinates of endpoints may lead to exponentially large description. We have extended this result even for intersection graphs of convex polygons in a plane. For string graphs (that were considered to be harder than segment graphs) the recognition problem was shown to be in NP [48]. For graphs of segments and convex polygons in a plane this problem remains open. The recognition problem is only known to be in PSPACE [37].

- *Sandwiching problem:* As many intersection-defined classes are being explored, it does not suffice to decide whether one particular class can be polynomially recognized. More efficient approach is to detect classes between whom are the "recognition gaps", *i.e.*, where no polynomially recognizable class can be sandwiched. For the first time such a gap was introduced by Kratochvíl [33]. Our reduction makes a similar gap between PC-graphs and graphs of interval filaments [46]. Still it is not known, for example, what is the smallest function f such that given a PC-graph we can decide in polynomial time whether it can be represented by (at most) k-gons (inscribed into a circle), or whether at least $f(k)+1$-gons are needed. Our result only shows that f cannot be identity.

- *Are segment graphs near-perfect? Are graphs of interval filaments near-perfect?* The former is a well-known question of Paul Erdös. Interval graphs are known to be perfect, polygon-circle graphs are known to be near-perfect [32]. For segment graphs this problem remains open. To anwser the near-perfectness question would be interesting also for graphs of interval filaments and for string graphs as for these classes this question was posed by Kratochvíl. We have found an intersection-

defined class which is not near-perfect, but the representing sets consisted of two arc-connected sets in a plane.

- *What is the most efficient FUN-representation that can be found in polynomial time?* We have already mentioned the fact that it is hard to find an optimum k such that a given graph is a PER_k-graph. It is known that each graph with n vertices is a $\mathrm{PER}_{\frac{n}{2}}$-graph, as a counterpart, we cannot approximate this k with better factor than $\Omega(\sqrt{n})$ (unless $NP \subseteq ZPP$). The problem is to determine better bounds and another reasonable question is to establish an inapproximation result without using the assumption that $NP \subseteq ZPP$. Note that this problem is also well-studied, as it is exactly the poset-dimension problem.

Bibliography

[1] M. Badent, C. Binucci, E. Di Giacomo, W. Didimo, S. Felsner, F. Giordano, J. Kratochvíl, P. Palladino, M. Patrignani, and F. Trotta. Homothetic triangle contact representations of planar graphs. In P. Bose, editor, *CCCG*, pages 233 – 236. Carleton University, Ottawa, Canada, 2007.

[2] J. Bang-Jensen, B. Reed, M. Schacht, R. Šámal, B. Toft, and U. Wagner. *Topics in Discrete Mathematics, Dedicated to Jarik Nešetřil on the Occasion of his 60th birthday*, volume 26 of *Algorithms and Combinatorics*, chapter On six problems posed by Jarik Nešetřil, pages 613 – 627. Springer, Berlin, M. Klazar, J. Kratochvíl, M. Loebl, J. Matoušek, R. Thomas and Pavel Valtr edition, 2006.

[3] A. Bouchet. Circle graph obstructions. *Journal of Combinatorial Theory, Series B*, 60(1):107 – 144, 1994.

[4] C. Dangelmayr and S. Felsner. Chordal graphs as intersection graphs of pseudosegments. In M. Kaufmann and D. Wagner, editors, *Graph Drawing*, volume 4372 of *Lecture Notes in Computer Science*, pages 208 – 219. Springer, 2007.

[5] C. Dangelmayr and M. Pergel. On the complicacy of intersection graphs of polygons. in preparation, 2008.

[6] H. de Fraysseix. A characterization of circle graphs. *Eur. J. Comb.*, 5:223 – 238, 1984.

[7] H. de Fraysseix, P. O. de Mendez, and P. Rosenstiehl. On triangle contact graphs. *Combinatorics, Probability & Computing*, 3:233 – 246, 1994.

[8] G. Ehrlich, S. Even, and R. E. Tarjan. Intersection graphs of curves in the plane. *Journal of Combinatorial Theory, Ser. B*, 21(1):8 – 20, 1976.

[9] E. S. Elmallah and L. K. Stewart. Polygon graph recognition. *Journal of Algorithms*, 26(1):101 – 140, 1998.

[10] J. Enright and M. Pergel. On some subclasses of subtree-overlap graphs. in preparation, 2008.

[11] J. Enright and L. Stewart. Subtree filament graphs are subtree overlap graphs. *Information Procesing Letters*, 104(6):228 – 232, 2007.

[12] J. A. Enright. Subtree overlap graphs – towards recognition. Msc. thesis, University of Alberta, 2006.

[13] E. M. Eschen and J. P. Spinrad. An $O(n)$ algorithm for circular-arc graph recognition. In *SODA*, pages 128 – 137, 1993.

[14] S. Even, A. Pnueli, and A. Lempel. Permutation graphs and transitive graphs. *Journal of Association for Comp. Machines*, 19(3):400 – 410, 1972.

[15] S. Felsner, R. Müller, and L. Wernisch. Trapezoid graphs and generalizations, geometry and algorithms. *Discrete Appl. Math.*, 74(1):13 – 32, 1997.

[16] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[17] M. R. Garey, D. S. Johnson, G. L. Miller, and C. H. Papadimitriou. The complexity of coloring circular arcs and chords. *SIAM Journal of Algebraic and Discrete Methods*, 1(2):216 – 227, 1980.

[18] F. Gavril. Algorithms on circular-arc graphs. *Networks*, 4:357 – 369, 1974.

[19] F. Gavril. Intersection graphs of helly families of subtrees. *Discrete Appl. Math.*, 66(1):45 – 56, 1996.

[20] F. Gavril. Maximum weight independent sets and cliques in intersection graphs of filaments. *Information Processing Letters*, 73(5 – 6):181 – 188, Mar. 2000.

[21] P. C. Gilmore and A. J. Hoffman. A characterization of comparability graphs and of interval graphs. *Canadian Journal of Mathematics*, 16:539 – 548, 1964.

[22] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics, Vol 57)*. North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, 2004.

[23] D. Goncalves, P. Ochem, and J. Chalopin. personal communication. 2006.

[24] A. Gyarfás. Problems from the world surrounding perfect graphs. *Zastosow. Mat.*, 19(3 – 4):413 – 441, 1987.

[25] R. Hedge and K. Jain. The hardness of approximating poset dimension. *Electronic Notes in Discrete Mathematics*, 29:435 – 443, 2007.

[26] T. Hiraguchi. On the dimension of orders. *Sci. Rep. Kanazawa Univ.*, 4(4):1 – 20, 1955.

[27] P. Hliněný and J. Kratochvíl. Representing graphs by disks and balls (a survey of recognition-complexity results). *Discrete Mathematics*, 229(1 – 3):101 – 124, 2001.

[28] I. Holyer. The NP-completeness of edge-coloring. *SIAM J. Comput.*, 10(4):718 – 720, 1981.

[29] M. Kaufmann, J. Kratochvíl, K. A. Lehmann, and A. R. Subramanian. Max-tolerance graphs as intersection graphs: cliques, cycles, and recognition. In *SODA*, pages 832 – 841. ACM Press, 2006.

[30] D. J. Kleitman and B. L. Rothschild. Asymptotic enumeration of partial orders on a finite set. *Trans. Amer. Math. Society*, 205:205 – 220, 1975.

[31] M. Koebe. On a new class of intersection graphs. In *Proceedings of the Fourth Czechoslovak Symposium on Combinatorics Prachatice*, pages 141 – 143, 1990.

[32] A. V. Kostochka and J. Kratochvíl. Covering and coloring polygon-circle graphs. *Discrete Math.*, 163(1 – 3):299 – 305, 1997.

[33] J. Kratochvíl. String graphs. II. recognizing string graphs is NP-hard. *Journal of Combinatorial Theory, Series B*, 52(1):67 – 78, 1991.

[34] J. Kratochvíl. Intersection graphs of noncrossing arc-connected sets in the plane. In S. North, editor, *Symposium on Graph Drawing, GD '96, Berkeley, California,, September 18 – 20, 1996*, pages 268 – 270. Springer, 1997.

[35] J. Kratochvíl, A. Lubiw, and J. Nešetřil. Noncrossing subgraphs in topological layouts. *SIAM J. Discrete Math.*, 4(2):223 – 244, 1991.

[36] J. Kratochvíl and J. Matoušek. String graphs requiring exponential representations. *J. Comb. Theory, Ser. B*, 53(1):1 – 4, 1991.

[37] J. Kratochvíl and J. Matoušek. Intersection graphs of segments. *J. Comb. Theory, Ser. B*, 62(2):289 – 315, 1994.

[38] J. Kratochvíl and J. Nešetřil. INDEPENDENT SET and CLIQUE problems in intersection defined classes of graphs. *Comment. Math. Univ. Carolin.*, 31:85 – 93, 1990.

[39] J. Kratochvíl and M. Pergel. Two results on intersection graphs of polygons. In G. Liotta, editor, *Graph Drawing*, volume 2912 of *Lecture Notes in Computer Science*, pages 59 – 70. Springer, 2003.

[40] J. Kratochvíl and M. Pergel. Geometric intersection graphs: Do short cycles help? In G. Lin, editor, *COCOON*, volume 4598 of *Lecture Notes in Computer Science*, pages 118 – 128. Springer, 2007.

[41] M. Massow and P. Allen. personal communication. 2008.

[42] J. Matoušek and J. Nešetřil. *Invitation to Discrete Mathematics*. Oxford University Press, Oxford, U.K., 1998.

[43] T. A. McKee and F. R. McMorris. *Topics on Intersection Graphs*. SIAM, 1999.

[44] J. Pach and G. Tóth. Recognizing string graphs is decidable. *Proceedings 9th International Symposium GD 2001, Vienna 2001, Lecture Notes in Computer Science*, 2265:247 – 260, 2002.

[45] M. Pergel. Algoritmy na průnikových grafech. Master's thesis, Charles University in Prague, Faculty of Mathematics and Physics, 2003.

[46] M. Pergel. Recognition of polygon-circle graphs and graphs of interval filaments is NP-complete. In A. Brandstädt, D. Kratsch, and H. Müller, editors, *WG*, volume 4769 of *Lecture Notes in Computer Science*, pages 238 – 247. Springer, 2007.

[47] D. J. Rose, R. E. Tarjan, and G. S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5(2):266 – 283, 1976.

[48] M. Schaefer, E. Sedgwick, and D. Štefankovič. Recognizing string graphs in NP. *J. Comput. Syst. Sci.*, 67(2):365 – 380, 2003.

[49] M. Schaefer and D. Stefankovic. Decidability of string graphs. In *ACM Symposium on Theory of Computing*, pages 241 – 246, 2001.

[50] J. Spinrad. Recognition of circle graphs. *J. Algorithms*, 16(2):264 – 282, 1994.

[51] J. P. Spinrad. `http://www.vuse.vanderbilt.edu/~spin/open.html`. 1995.

[52] J. P. Spinrad. *Efficient Graph Representations*. American Mathematical Society, 2003.

[53] A. C. Tucker. An efficient test for circular-arc graphs. *SIAM J. Comput.*, 9(1):1 – 24, 1980.

[54] W. Unger. On the k-colouring of circle-graphs. In R. Cori and M. Wirsing, editors, *STACS*, volume 294 of *Lecture Notes in Computer Science*, pages 61 – 72. Springer, 1988.

[55] M. Yannakakis. The complexity of the partial order dimension problem. *SIAM J. Algebraic Discrete Methods*, 3(3):351 – 358, 1982.