Algorithmica (2027) 89:1
https://doi.org/10.1007/s00453-026-01413-4

Simple Realizability of Abstract Topological Graphs

Giordano Da Lozzo[1] · Walter Didimo[2] · Fabrizio Montecchiani[2] · Miriam Münch[3] · Maurizio Patrignani[1] · Ignaz Rutter[3]

Received: 19 March 2025 / Accepted: 18 October 2026
© The Author(s) 2026

Abstract

An *abstract topological graph* (*AT-graph*) is a pair $A = (G, \mathcal{X})$, where $G = (V, E)$ is a graph and $\mathcal{X} \subseteq \binom{E}{2}$ is a set of pairs of edges of G. A *realization of* A is a drawing Γ_A of G in the plane such that any two edges e_1, e_2 of G cross in Γ_A if and only if $(e_1, e_2) \in \mathcal{X}$; Γ_A is *simple* if any two edges intersect at most once (either at a common endpoint or at a proper crossing). The AT- GRAPH REALIZABILITY (ATR) problem asks whether an input AT-graph admits a realization. The version of this problem that requires a simple realization is called SIMPLE AT- GRAPH REALIZABILITY (SATR). It is a classic result that both ATR and SATR are NP-complete [18, 21]. In this paper, we study the SATR problem from a new structural perspective. More precisely, we consider the size $\lambda(A)$ of the largest connected component of the *crossing graph* of any realization of A, i.e., the graph $\mathcal{C}(A) = (E, \mathcal{X})$. This parameter represents a natural way to measure the level of interplay among edge crossings. First, we prove that SATR is NP-complete when $\lambda(A) \geq 6$. On the positive side, we give an optimal linear-time algorithm that solves SATR when $\lambda(A) \leq 3$ and returns a simple realization if one exists. Our algorithm is based on several ingredients, in particular the reduction to a new embedding problem subject to constraints that require certain pairs of edges to alternate (in the rotation system), and a sequence of transformations that exploit

✉ Miriam Münch
muenchm@fim.uni-passau.de

Giordano Da Lozzo
giordano.dalozzo@uniroma3.it

Walter Didimo
walter.didimo@unipg.it

Fabrizio Montecchiani
fabrizio.montecchiani@unipg.it

Maurizio Patrignani
maurizio.patrignani@uniroma3.it

Ignaz Rutter
rutter@fim.uni-passau.de

[1] Roma Tre University, Rome, Italy

[2] University of Perugia, Perugia, Italy

[3] University of Passau, Passau, Germany

Published online: 02 November 2026

the interplay between alternation constraints and the SPQR-tree and PQ-tree data structures to eventually arrive at a simpler embedding problem that can be solved with standard techniques.

Keywords Abstract topological graphs · SPQR-trees · Synchronized PQ-trees

1 Introduction

A *topological graph* Γ is a geometric representation of a graph $G = (V, E)$ in the plane such that the vertices of G are mapped to distinct points and the edges of G are simple curves connecting the points corresponding to their end-vertices. For simplicity, the geometric representations of the elements of V and E in Γ are called *vertices* and *edges* of Γ, respectively. It is required that: (i) any intersection point of two edges in Γ is either a common endpoint or a point where the two edges properly intersect; (ii) any two edges of Γ have finitely many intersections and no three edges pass through the same crossing point. Additionally, we say that Γ is *simple* if adjacent edges never cross and any two non-adjacent edges cross at most once. The *crossing graph* $\mathcal{C}(\Gamma)$ of a topological graph Γ is a graph whose vertices correspond to the edges of Γ (and hence of G) and two vertices are adjacent if and only if their corresponding edges cross in Γ.

An *abstract topological graph* (*AT-graph*) is a pair $A = (G, \mathcal{X})$ such that $G = (V, E)$ is a graph and $\mathcal{X} \subseteq \binom{E}{2}$ is a set of pairs of edges of G. We say that A is *realizable* if there exists a topological graph Γ_A isomorphic to G such that any two edges e and e' of G cross (possibly multiple times) in Γ_A if and only if $(e, e') \in \mathcal{X}$. The topological graph Γ_A is called a *realization of* A. Note that, by definition, A is realizable if and only if the crossing graph of any realization Γ_A of A is isomorphic to the graph $(E, \mathcal{X})$. Since such a crossing graph only depends on A (i.e., it is the same for any realization of A), we denote it by $\mathcal{C}(A)$.

The AT- GRAPH REALIZABILITY (ATR) problem asks whether an AT-graph $A = (G, \mathcal{X})$ is realizable. The SIMPLE AT- GRAPH REALIZABILITY (SATR) problem is the version of ATR in which the realization of A is required to be simple; if such a realization exists, then A is said to be *simply realizable*. Since the introduction of the concept of AT-graphs [20], establishing the complexity of the ATR (and of the SATR) problem has been widely investigated, also due to its connection with other prominent problems in topological and geometric graph theory. Clearly, if $\mathcal{X} = \emptyset$, both the ATR and the SATR problems are equivalent to testing whether G is planar, which is solvable in linear time [3, 17]. However, for $\mathcal{X} \neq \emptyset$, a seminal paper by Kratochvíl [18] proves that ATR is NP-hard and that this problem is polynomially equivalent to recognizing *string graphs*. We recall that a graph S is a string graph if there exists a system of curves (called *strings*) in the plane whose crossing graph is isomorphic to S. In the same paper, Kratochvíl proves the NP-hardness of the WEAK AT- GRAPH REALIZABILITY (WATR) problem, that is, deciding whether a given AT-graph $A = (G, \mathcal{X})$ admits a realization where a pair of edges may cross only if it belongs to $\mathcal{X}$. He also proves that recognizing string graphs remains polynomial-time reducible to WATR. Subsequent results focused on establishing decision algorithms for WATR; it was first proven

that this problem belongs to NEXP [29] and then to NP [28]. This also implies the NP-completeness of recognizing string graphs (which is polynomial-time reducible to WATR) and of ATR (which is polynomially equivalent to string graph recognition). Concerning the simple realizability setting for AT-graphs, it is known that the SATR problem remains NP-complete, still exploiting the connection with recognizing string graphs [21, 22]. On the positive side, for those AT-graphs $A = (G, \mathcal{X})$ for which G is a complete n-vertex graph, SATR is solvable in polynomial-time, with an $O(n^6)$-time algorithm [23, 24]. Refer to [23] for the complexity of other variants of ATR. Also, a special case of the ATR problem is the popular Simultaneous Embedding with Fixed Edges problem [11], in which multiple graphs on the same vertex set are given and one has to compute a simultaneous drawing of the graphs where common edges are represented by the same curves and no two edges in the same graph cross. In fact, the weak realizability problem is polynomially equivalent to the Simultaneous Graph Embedding with Fixed Edges [11, Theorem 1].

Our contributions. In this paper, we further investigate the complexity of the simple realizability setting, i.e., of the SATR problem. We remark that focusing on simple drawings is a common scenario in topological graph theory, computational geometry, and graph drawing (see, e.g., [8, 12, 15, 30]), because avoiding crossings between adjacent edges, as well as multiple crossings between a pair of non-adjacent edges, is a requirement for minimal edge crossing layouts. Specifically, we study the simple realizability problem for an AT-graph A from a new structural perspective, namely looking at the number of vertices of the largest connected component of the crossing graph $\mathcal{C}(A)$, which we denote by $\lambda(A)$. This parameter is a natural measure of the level of interplay among edge crossings. Namely, SATR is trivial on instances for which $\lambda(A) \leq 2$, that is, instances in which the number of crossings is unbounded but each edge is crossed at most once. On the other hand, the problem becomes immediately nontrivial as soon as $\lambda(A) \geq 3$. Precisely, our results are as follows:

- We prove that Simple AT-graph Realizability is NP-complete already for instances A for which $\lambda(A) = 6$ (which, in fact, implies the hardness for every fixed value of $\lambda(A) \geq 6$, as shown in Corollary 1); see Sect. 3. A consequence of our result is that SATR is paraNP-hard when parameterized by $\lambda(A)$. In particular, unless $\mathsf{P} = \mathsf{NP}$, the problem is neither XP nor FPT with respect to $\lambda(A)$ and, thus, with respect to any graph parameter bounded by $\lambda(A)$, such as the maximum node degree, the treewidth or even the treedepth. Like the results in [18, 21, 22], our hardness proof uses a reduction from 3-Connected Planar 3-SAT. However, the reduction in [18] does not deal with the simplicity of the realization, whereas the reduction in [21, 22] may lead to instances A for which $\lambda(A)$ is greater than six and, actually, is even not bounded by a constant.
- We prove that Simple AT-graph Realizability can be solved efficiently when $\lambda(A) \leq 3$. More precisely, we give an optimal $O(n)$-time testing algorithm, which also finds a simple realization if one exists; see Sect. 4. We remark that the only polynomial-time algorithm previously known in the literature for the SATR problem is restricted to complete graphs and has high complexity [23, 24]. Our algorithm is based on several ingredients, including the reduction to a new embedding problem subject to constraints that require certain pairs of edges to alternate

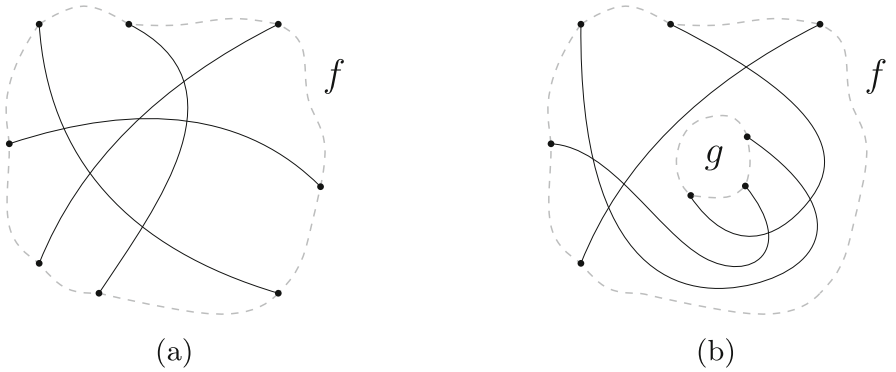


Fig. 1 **a** An untangled 4-crossing; all the endpoints of the four edges belong to face f. **b** A tangled 4-crossing; some endpoints belong to a face g distinct from f

(in the rotation system), and a sequence of transformations that exploit the interplay between alternation constraints and the SPQR-tree [6, 7] and PQ-tree [3, 4] data structures to eventually arrive at a simpler embedding problem that can be solved with standard techniques. We remark that the alternation constraints we encounter in our problem are rather opposite to the more-commonly encountered consecutivity constraints [1, 2, 13, 27] and cannot be handled straightforwardly with PQ-trees.

2 Basic Definitions and Tools

For basic definitions about graphs and their drawings, refer to [5, 9]. We only consider simple realizations and thus we often omit the qualifier "simple" when clear from the context.

Let $A = (G, \mathcal{X})$ be an AT-graph, with $G = (V, E)$, and let Γ_A be a realization of A. A *face* of Γ_A is a region of the plane bounded by maximal uncrossed portions of the edges in E. A set $E' \subseteq E$ of k edges pairwise crossing in Γ_A is a *k-crossing* of A (or, equivalently, of Γ_A). As we focus on simple realizations, we assume that the edges in E' are pairwise non-adjacent in G. For a k-crossing E', denote by $V(E')$ the set of $2k$ endpoints of the k edges in E'. The *arrangement* of E', denoted by $C_{E'}$, is the arrangement of the curves representing the edges of E' in Γ_A. A k-crossing E' is *untangled* if, in the arrangement $C_{E'}$, all $2k$ endpoints in $V(E')$ are incident to a common face (see Fig. 1a); otherwise, it is *tangled* (see Fig. 1b). The next lemma will turn useful in Sect. 4.

Lemma 1 *An AT-graph A with $\lambda(A) \leq 3$ admits a simple realization if and only if it admits a simple realization in which all 3-crossings are untangled.*

Proof The sufficiency is obvious, as any simple realization of an AT-graph in which all 3-crossings are untangled is a specific simple realization. For the necessity, let A be an AT-graph with $\lambda(A) \leq 3$ and let Γ_A be a simple realization of A that contains a tangled

3-crossing E'. Since the edges in E' pairwise cross and $\lambda(A) \leq 3$, the vertices of $\mathcal{C}(A)$ corresponding to the edges in E' induce a maximal connected component of $\mathcal{C}(A)$ that is a 3-cycle (the edges in E' do not cross any edge in $E \setminus E'$). We show how to obtain a new simple realization Γ'_A of A that coincides with Γ_A except for the drawing of one of the edges in E' and such that E' is untangled. Repeating such a transformation for each tangled 3-crossing eventually yields the desired simple realization of A with no tangled 3-crossings.

Consider the arrangement $C_{E'}$ of E' in Γ_A. Since Γ_A is simple and $|E'| = 3$, we have that $C_{E'}$ splits the plane into two faces, one bounded and one unbounded. Let $e_1 = (u_1, v_1)$, $e_2 = (u_2, v_2)$, and $e_3 = (u_3, v_3)$ be the edges in E'. Since E' is tangled and Γ_A is simple, one of such faces, denoted by f, contains an endpoint of two edges of E' and the other face, denoted by h, contains the remaining four endpoints. In fact, any configuration in which exactly one or exactly three endpoints are incident to one of the two faces of the arrangement would require Γ_A to be non-simple. In the remainder, we assume that f is bounded (refer to Fig. 2). This is not a loss of generality, as a stereographic projection can be applied to Γ_A to satisfy this assumption. W.l.o.g., we also assume that the endpoints of the edges of E' incident to f are v_1 and v_2. Denote by G_f (resp. by G_h) the subgraph of G consisting of the vertices and of the edges of G that lie in the interior of f (resp. of h); refer to Fig. 2. Let Q be a closed curve passing through v_1 and v_2 that encloses the drawing of G_f in Γ_A, without intersecting any other vertex or edge; refer to Fig. 2b. Let us follow the drawing of the edge e_2 from u_2 to v_2. Suppose that, as in Fig. 2, at the intersection point p between e_2 and e_1, we see the endpoint u_1 of e_1 to the left; the case in which u_1 is to the right being symmetric. Consider three points a, b, and c along e_1 in Γ_A, encountered in this order going from u_1 to v_1, such that a and b occur before (and arbitrarily close to) p, and c lies after p and arbitrarily close to v_1. To obtain Γ'_A, replace the drawing of e_1 in Γ_A with the union of four curves $\lambda_1, \lambda_3, \lambda_3, \lambda_4$ defined as follows (refer to Fig. 2c):

- λ_1 coincides with the drawing of e_1 in Γ_A between u_1 and a (it only intersects e_3);
- λ_2 starts at a, follows the drawing of e_2 between p and v_2, and then follows clockwise the border of Q between v_2 and v_1 until it reaches c (it intersects no edge);
- λ_3 coincides with the drawing of e_1 between c and b (it only intersects e_2); and
- λ_4 starts at b, follows the drawing of e_2 between p and v_2, and then follows clockwise the border of Q between v_2 and v_1 until it reaches v_1 (it intersects no edge).

Since the drawing of Γ'_A coincides with Γ_A except for the drawing of e_1, and since we have redrawn e_1 in such a way that it only intersects (once) e_2 and e_3, we have that Γ'_A is a simple realization of A. Moreover, when traversing e_2 from u_2 to v_2, we now have that, at the intersection point between e_2 and e_1, we see the endpoint u_1 of e_1 to the right. Therefore, all the endpoints of the edges in E' lie in the same face of the arrangement they form in Γ'_A, that is, E' is untangled in Γ'_A. □

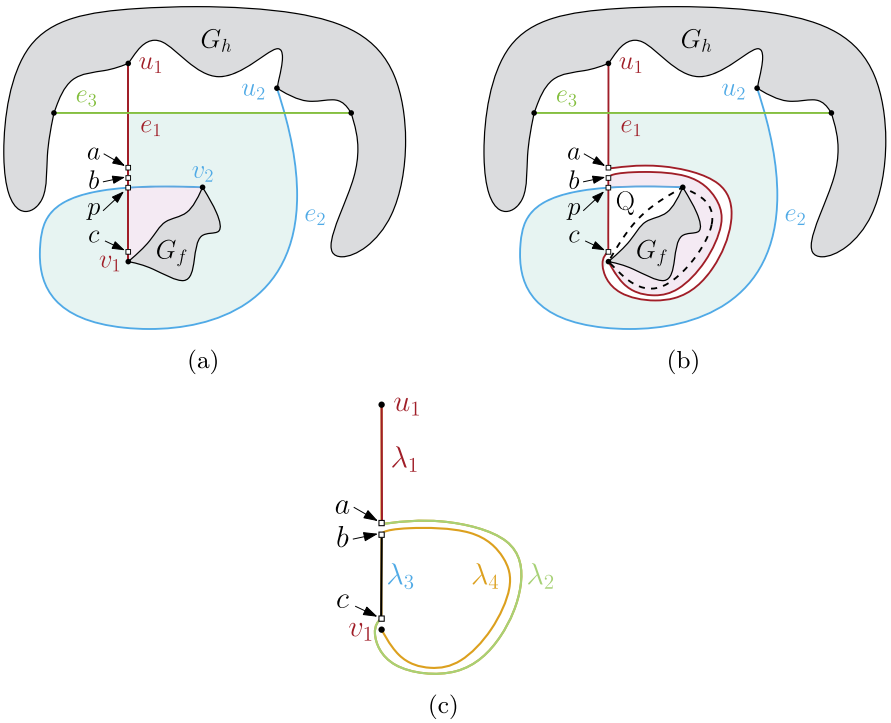


Fig. 2 Illustrations for the proof of Lemma 1. **a** A schematic representation of a simple realization Γ_A of an AT-graph A with a tangled 3-crossing $E' = \{e_1, e_2, e_3\}$. **b** The simple realization Γ'_A obtained from Γ_A, where E' is untangled. **c** The curves forming e_1 in Γ'_A

2.1 Connectivity

Let G be a graph. The graph G is *k-connected* if it contains at least k vertices and the removal of any $k-1$ vertices leaves it connected; 2- and 3-connected graphs are also called *biconnected* and *triconnected*, respectively. A *cut vertex* (resp. *separation pair*) of G is a vertex (resp. a pair of vertices) whose removal disconnects G. Therefore, the graph G is biconnected (resp. triconnected) if it has no cut vertex (resp. no separation pair). A *biconnected component* (or *block*) of G is a maximal (in terms of vertices and edges) biconnected subgraph of G. The *cut components* of a cut vertex v of G are the subgraphs of G induced by v together with the maximal subsets of the vertices of G that are not disconnected by the removal of v.

2.2 PQ-Trees

PQ-trees were first introduced by Booth and Lueker [3, 4]. An *unrooted PQ-tree* over a finite set U is an unrooted tree whose leaves are the elements of U and whose internal nodes are labeled as P- or Q-nodes. A P-node is usually depicted as a circle, a Q-node as a rectangle; see Fig. 3. Consider an unrooted PQ-tree T and let L be the circular

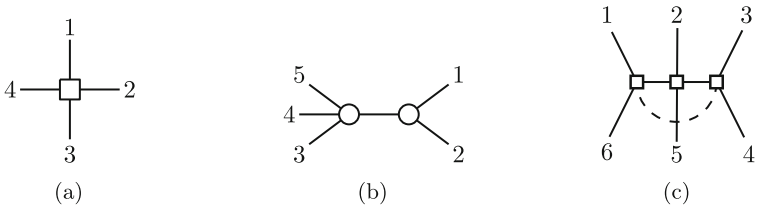


Fig. 3 PQ-trees representing the circular orders **a** (1, 2, 3, 4) and its reverse, **b** of 1, 2, 3, 4, 5, such that 1, 2 are consecutive, **c** (6, 1, 2, 3, 4, 5), (6, 1, 5, 3, 4, 2), (1, 6, 2, 4, 3, 5), (1, 6, 5, 4, 3, 2)

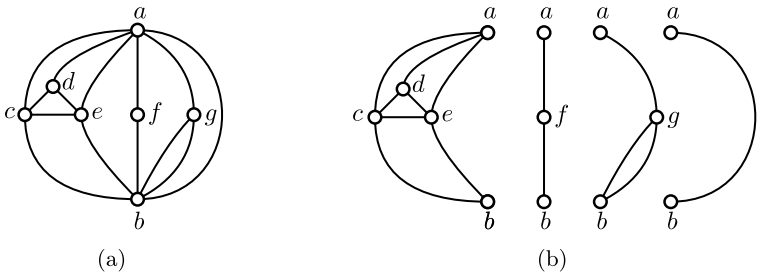


Fig. 4 **a** A graph G. **b** The four split components of G with respect to the split pair $\{a, b\}$.

order of its leaves. Then T represents the set of those circular orders of its leaves that can be obtained from L by (i) arbitrarily permuting the children of arbitrarily many P-nodes and (ii) reversing the order of arbitrarily many Q-nodes. Since all PQ-trees we consider in the paper are unrooted, we will refer to *unrooted PQ-trees* if we write PQ-trees. A *synchronized* PQ-tree contains a pair of Q-nodes that have to be flipped simultaneously; see Fig. 3c.

2.3 SPQR-Trees

Let G be a biconnected graph. A *separation pair* of G is a pair of vertices whose removal disconnects G and a *split pair* of G is a pair of adjacent vertices or a *separation pair*. The *split components* of G *with respect to* a split pair $\{u, v\}$ are defined as follows; see Fig. 4 for an example. If (u, v) is an edge of G, then it is a split component of G with respect to $\{u, v\}$. Also, let $G_1, \ldots, G_k$ be the connected components of $G \setminus \{u, v\}$. The subgraphs of G induced by $V(G_i) \cup \{u, v\}$, minus the edge (u, v), are components of G with respect to $\{u, v\}$, for $i = 1, \ldots, k$.

The SPQR-tree T of a biconnected graph G, introduced by Di Battista and Tamassia [6, 7], represents a decomposition of G into triconnected components along its split pairs. Each node μ of T contains a graph $\mathrm{sk}(\mu)$, called *skeleton of* μ. The edges of $\mathrm{sk}(\mu)$ are either edges of G, which we call *real edges*, or newly introduced edges, called *virtual edges*. The tree T is initialized to a single node μ, whose skeleton, composed only of real edges, is G. Consider a split pair $\{u, v\}$ of the skeleton of some node μ of T, and let $G_1, \ldots, G_k$ be the split components of G with respect to $\{u, v\}$ such that G_1 is not a virtual edge and, if $k = 2$, also G_2 is not a virtual

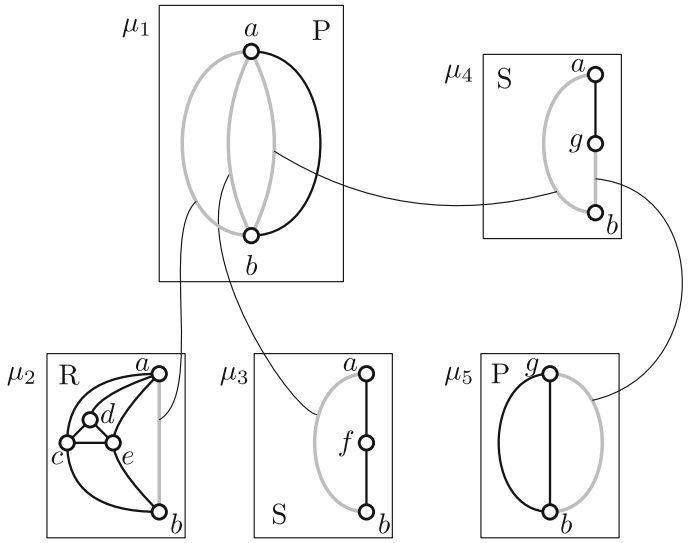


Fig. 5 The SPQR-tree decomposition of the graph G shown in Fig. 4a (virtual edges drawn in gray), where Q-nodes are omitted in favor of allowing real edges (drawn in black) in the skeletons of the other nodes

edge. We introduce a node ν adjacent to μ whose skeleton is the graph $G_1 + e_{\nu,\mu}$, where $e_{\nu,\mu} = (u, v)$ is a virtual edge, and replace $\mathrm{sk}(\mu)$ with the graph $\bigcup_{i\neq 1} G_i + e_{\mu,\nu}$, where $e_{\mu,\nu} = (u, v)$ is a virtual edge. We say that $e_{\nu,\mu}$ is the *twin virtual edge* of $e_{\mu,\nu}$ and vice versa. Applying this replacement iteratively produces a tree with more nodes but smaller skeletons associated with the nodes. Eventually, when no further replacement is possible, the skeletons of the nodes of T are of four types: parallels of at least three virtual edges (P-nodes), parallels of exactly one virtual edge and one real edge (Q-nodes), cycles (S-nodes), and triconnected planar graphs (R-nodes). See Fig. 5 for an example. The two vertices in the skeleton of an P-node μ are called the *poles* of μ.

The *merge* of two adjacent nodes μ and ν in T, replaces μ and ν in T with a new node τ that is adjacent to all the neighbors of μ and ν, and whose skeleton is $\mathrm{sk}(\mu) \cup \mathrm{sk}(\nu) \setminus \{e_{\mu,\nu}, e_{\nu,\mu}\}$, where the end-vertices of $e_{\mu,\nu}$ and $e_{\nu,\mu}$ that correspond to the same vertices of G are identified. By iteratively merging adjacent S-nodes, we eventually obtain the (unique) SPQR-tree data structure as introduced by Di Battista and Tamassia [6, 7], where the skeleton of an S-node is a cycle. The crucial property of this decomposition is that a planar embedding of G uniquely induces a planar embedding of the skeletons of its nodes and that, arbitrarily and independently, choosing planar embeddings for all the skeletons uniquely determines an embedding of G. Observe that the skeletons of S- and Q-nodes have a unique planar embedding, that the skeleton of R-nodes have two planar embeddings (where one is the reverse of the other), and that P-nodes have as many planar embedding as the permutations of their virtual edges. If G has n vertices, then T has $O(n)$ nodes and the total number of virtual edges in the skeletons of the nodes of T is in $O(n)$. From a computational complexity perspective, T can be constructed in $O(n)$ time [14].

3 NP-Completeness for AT-Graphs with $\lambda(A) \geq 6$

In this section, we show that the SATR problem is NP-complete for an AT-graph A even when the largest component of the crossing graph $\mathcal{C}(A)$ has bounded size; specifically, when $\lambda(A) = 6$ (see Theorem 1). We will exploit a reduction from the NP-complete problem 3- CONNECTED PLANAR 3- SAT [19].

In a nutshell, given an instance φ of 3- CONNECTED PLANAR 3- SAT, the main idea behind our reduction is to construct an instance $A_\varphi = (G', \mathcal{X}')$ of SATR that is simply realizable if and only if φ is satisfiable. To do so, the instance A_φ will contain a non-self-intersecting triconnected planar subgraph H_φ, called *frame*, that will be used to determine some regions of the plane consisting of some faces of the planar realization of H_φ. A crucial property we exploit is that the faces of H_φ are uniquely determined, up to a flip, since H_φ is triconnected. Specifically, each variable v and each clause c of φ is assigned to a specific face of the frame, and for each occurrence of a variable v into a clause c in φ, the frame contains a face, behaving as *pipe*, that connects the faces to which v and c have been assigned. Also, for each variable or clause, we will augment A_φ with a subgraph that is constrained, by means of the edge-pairs in $\mathcal{X}'$, to lie inside the face of the frame the corresponding variable or clause is assigned to. The subgraph that represents a variable, called *variable gadget*, when constrained to lie within the associated face, admits exactly two distinct simple realizations that correspond to the two possible truth values of its variable. Each of such realizations will transmit a different constraint to all the subgraphs, called *clause gadgets*, corresponding to the clauses the variable occurs in. Such a transmission is encoded in the order in which two paths enter and exit the pipe connecting the variable and the clause faces.

Let φ be a Boolean formula in conjunctive normal form. The *variable-clause graph* G_φ *of* φ is the bipartite (multi-)graph that has a node for each variable and for each clause, and an edge between a variable-node and a clause-node if a positive or negated literal of the variable appears in the clause. If each clause of φ has exactly three literals corresponding to different variables and G_φ is planar and triconnected, then φ is an instance of 3- CONNECTED PLANAR 3- SAT. Observe that in this case G_φ is a simple graph. Our proof exploits several gadgets described hereafter, which are then combined to obtain the desired reduction.

Intuitively, in the instance A_φ of SATR corresponding to φ, we encode truth values into the clockwise or counter-clockwise order in which some edges cross suitable cycles of the subgraphs (called "variable gadgets") representing the variables of φ in A_φ. These edges connect the variable gadgets to the subgraphs (called "clause gadgets") representing those clauses that contain a literal of the corresponding variable. Only if at least one of the literals appearing in a clause gadget encodes a `true` value, the clause gadget admits a simple realization. We start by describing the "frame" of A_φ, that is the part of A_φ that encloses all the gadgets and ensures that they are properly connected. Next, we describe the "split gadget" which, in turn, is used to construct the variable gadget. If a variable v has literals in k clauses, we have k pairs of edges leaving the corresponding variable gadget and entering the k clause gadgets. The clause gadgets always receive three truth values, corresponding to the three literals of the corresponding clause.

Frame. Arbitrarily choose a planar embedding $\mathcal{E}_\varphi$ of G_φ. The *frame* H_φ *of* φ is a 4-regular 3-connected plane graph obtained from $\mathcal{E}_\varphi$ as follows; refer to Fig. 6. For each degree-k variable v_i of φ, the graph H_φ contains a k-cycle formed by the sequence of edges $(e_{v_i,c_i^1}, e_{v_i,c_i^2}, \ldots, e_{v_i,c_i^k})$, which we refer to as the *variable cycle* of v_i, where $c_i^1, \ldots, c_i^k$ are the clause-nodes of G_φ adjacent to v_i in the clockwise order in which they appear around v_i in $\mathcal{E}_\varphi$. For each clause c_j of φ, the graph H_φ contains a 3-cycle formed by the sequence of edges $(e_{c_j,v_j^1}, e_{c_j,v_j^2}, e_{c_j,v_j^3})$, which we refer to as the *clause cycle* of c_j, where v_j^1, v_j^2, v_j^3 are the variable-nodes of G_φ adjacent to c_j in the clockwise order in which they appear around c_j in $\mathcal{E}_\varphi$. For each edge (v_i, c_j) of G_φ, the graph H_φ contains two edges, which we refer to as the *pipe edges* of the edge (v_i, c_j), connecting the endpoints of e_{v_i,c_j} and e_{c_j,v_i} without crossings. We need the following lemma to show that the frame H_φ obtained from $\mathcal{E}_\varphi$ is triconnected.

Lemma 2 *Let G be a triconnected graph, and let u and v be two vertices of G. Also, let e_u and e_v be two edges incident to u and to v, respectively. Then, G contains three vertex disjoint paths connecting u and v that use both e_u and e_v.*

Proof Recall that, by Menger's theorem [26], since G is triconnected, it contains three vertex-disjoint paths P_1, P_2, and P_3 connecting u and v. If such paths use both e_u and e_v, there is nothing to prove. Thus, suppose that at least one of e_u and e_v, w.l.o.g., say e_u, does not appear in any of such paths. We distinguish two cases: **Case 1**: e_v belongs to one of P_1, P_2, and P_3; and **Case 2**: e_v does not belong to any of P_1, P_2, and P_3.

In **Case 1**, assume w.l.o.g. that e_v belongs to P_1. Consider a path P_u between u and v that contains $e_u = (u, u')$. Observe that P_u always exists, since the removal of u from G leaves $G \setminus \{u\}$ connected (in fact, biconnected); P_u can be obtained by the union of e_u with a path between u' and v in $G \setminus \{u\}$. If P_u is disjoint, except at its endpoints, from all other paths, then P_u, P_1, and any of P_2 and P_3 satisfy the requirements of the statement, since such paths are vertex-disjoint and since $e_u \in E(P_u)$ and $e_v \in E(P_1)$. If P_u intersects one of the other paths at an internal vertex, then let x be the first vertex of P_u, encountered when traversing this path from u to v, that belongs to a path P_i, with $i \in \{1, 2, 3\}$. Let P_x be the path obtained by concatenating the subpath of P_u between u and x and the subpath of P_i between x and v. Then, if $i = 1$, the path P_x contains both e_u and e_v, and it is vertex disjoint from P_2 and P_3. Thus, P_x, together with P_2 and P_3 satisfy the requirements of the statement. Otherwise, the path P_x together P_1 and the path P_j with $j \in \{1, 2, 3\} \setminus \{i, 1\}$ satisfy the requirements of the statement.

In **Case 2**, consider again a path P_u between u and v that contains e_u. We show how to modify paths P_1, P_2, and P_3 so that the conditions of **Case 1** apply to P_1, P_2, and P_3 (up to renaming u and v) and we can thus proceed as described above. If P_u is disjoint, except at its endpoints, from all other paths, then we set $P_1 = P_x$. If P_u intersects one of the other paths at an internal vertex, we define the vertex x and the path P_x as above. Let i be the index in $\{1, 2, 3\}$ of the path P_i containing x. Then, we set $P_i = P_x$. Since P_x contains e_u, **Case 1** holds for the updated paths P_1, P_2, and P_3 (up to renaming u and v). This concludes the proof. □

Lemma 3 *The frame H_φ obtained from $\mathcal{E}_\varphi$ is triconnected.*

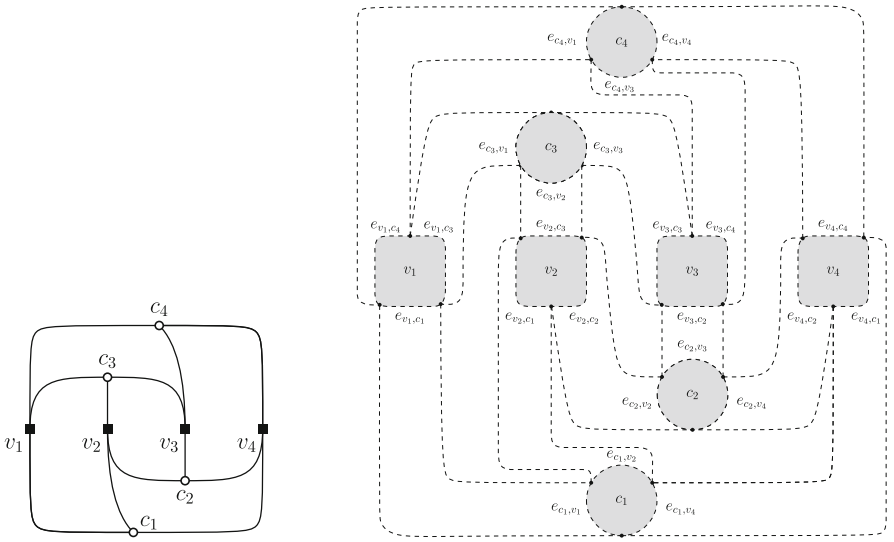


Fig. 6 (Left) A planar 3-connected variable-clause graph G_φ and (Right) the corresponding frame H_φ

Proof Recall that, by construction, each vertex of H_φ belongs either to a variable cycle or to a clause cycle stemming from a vertex of G_φ. Therefore, by Menger's theorem, to prove the triconnectivity of G_φ, we show the existence of three vertex-disjoint paths between any two vertices u and v of H_φ.

Consider first the case in which u and v belong to the same variable or clause cycle C of H_φ. Then there exist already two vertex-disjoint paths between u and v along C. Also, since removing the vertex corresponding to C from G_φ leaves the resulting graph connected, we also have that removing from H_φ the edges of C together with the vertices of C different from u and v leaves the resulting sub-graph of H_φ connected. This implies that there exists in G_φ a vertex-disjoint path between u and v that avoids the edges of C and the remaining vertices of C.

Consider now the case that u and v belong to distinct variable or clause cycles of H_φ, which we denote by C_1 and C_2, respectively. The cycles C_1 and C_2 stem from a node n_1 and a node n_2 of G_φ, respectively. Let e_u be one of the two edges of H_φ incident to u and not belonging to C_1. Analogously, let e_v be one of the two edges of H_φ incident to v and not belonging to C_2. Observe that e_u and e_v are pipe edges of some edge e_1 of G_φ incident to n_1 and of some edge e_2 of G_φ incident to n_2. By carefully choosing e_u and e_v we can enforce that $e_1 \neq e_2$. It follows that e_u and e_v are never the two pipe edges of the same edge of G_φ. By Lemma 2, there exist three vertex-disjoint paths P_1, P_2, and P_3 in G_φ between n_1 and n_2 that use e_1 and e_2. We show that there exist three vertex-disjoint paths in H_φ between u and v. Let P_i, $i \in \{1, 2, 3\}$ be one such path. For each edge e of P_i we select an edge e_H of H_φ such that e_H is a pipe edge of e: If e is incident to n_1 (resp. to n_2), $e_H = e_u$ (resp. $e_H = e_v$); otherwise, e_H is arbitrarily chosen. It is possible to construct a path p_i in H_φ composed of the edges of H_φ defined for P_i, where for each pair of edges e'_H and e''_H originating from two edges of G_φ incident to the same vertex n_j, we add the edges

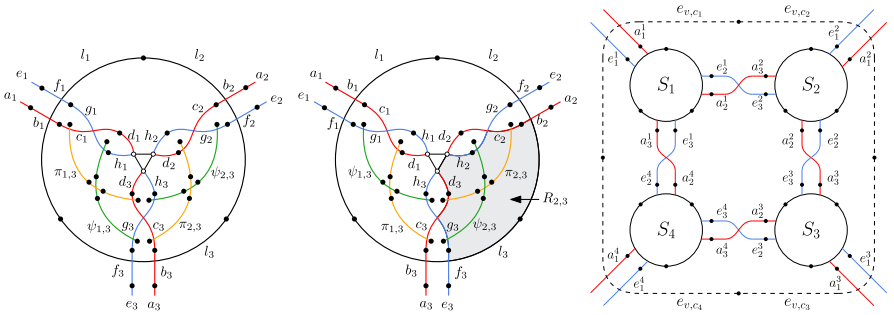


Fig. 7 **a**, **b** The split gadget S: The clockwise circular order of the edges leaving the gadget is either b_1, f_1, b_2, f_2, b_3, f_3 (**a**) or f_1, b_1, f_2, b_2, f_3, b_3 (**b**). **c** The variable gadget $\mathcal{V}_v$. The dashed edges belong to the variable cycle of v in the frame H_φ

of any of the two paths of H_φ connecting the endpoints of e'_H and e''_H along C_j. The construction of p_i fails only in the case in which the two edges e_u and e_v are the two pipe edges of the same edge (n_1, n_2), which is ruled out by the way we choose e_u and e_v. We have obtained three vertex-disjoint paths p_1, p_2, and p_3 in H_φ from vertices of C_1 to vertices of C_2 that do not use edges of C_1 and C_2 and such that u and v are both extremes of some of these paths. By using the edges of C_1 and C_2 it is possible to construct three vertex-disjoint paths in H_φ between u and v. □

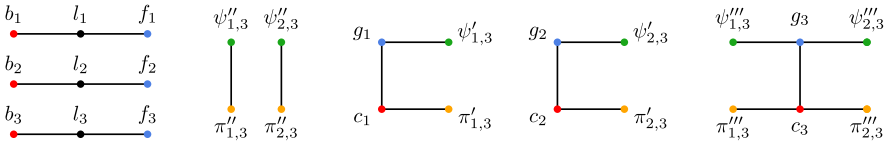
Split gadget. The split gadget S is the AT-graph defined as follows; refer to Fig. 7a and b. The underlying graph of S consists of six connected components: (1) a 3-cycle formed by the edges l_1, l_2, and l_3, which we call *outer cycle* of S; (2) a 3-cycle formed by the vertices v_1, v_2, and v_3 (filled white in Fig. 7a and b) such that, for $i = 1, 2, 3$, the vertex v_i is the endpoint of the two paths formed by the sequence of edges (a_i, b_i, c_i, d_i) (red paths in Figs. 7a and 7b) and (e_i, f_i, g_i, h_i) (blue paths in Fig. 7a and b); (3) four length-3 paths $\pi_{1,3}$, $\pi_{2,3}$, $\psi_{1,3}$, and $\psi_{2,3}$. Given one such length-3 path p, we denote by p', p'', and p''' the three edges of p, where p'' is the middle one, while p' and p''' are the other two (arbitrarily chosen).

The edges of the crossing graph $\mathcal{C}(S)$ of S are defined by the following crossings: (see Fig. 8a): (i) For $j = 1, 2, 3$, edge l_j crosses both b_j and f_j; (ii) For $j = 1, 2$, edge $\pi''_{j,3}$ crosses $\psi''_{j,3}$; (iii) For $j = 1, 2$, edge c_j crosses g_j and $\pi'_{j,3}$, further g_j crosses $\psi'_{j,3}$; finally (iv) edge c_3 crosses g_3, $\pi'''_{1,3}$, and $\pi'''_{2,3}$, further g_3 crosses $\psi'''_{1,3}$ and $\psi'''_{2,3}$.

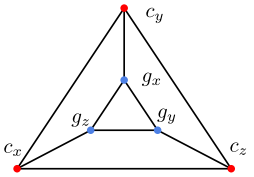
Observe that $\mathcal{C}(S)$ consists of eight *non-trivial connected components* of $\mathcal{C}(S)$, i.e., connected components that are not isolated vertices.

Lemma 4 *In any simple realization of S, the circular (clockwise or counterclockwise) order of the edges crossing the outer cycle of S is either b_1, f_1, b_2, f_2, b_3, f_3 or f_1, b_1, f_2, b_2, f_3, b_3.*

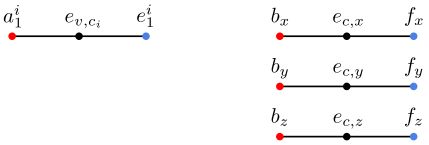
Proof Let C be the outer cycle of S, which consists of the edges l_1, l_2, and l_3. Consider a simple realization Γ of S, such that l_1, l_2, and l_3 are encountered in this order when



(a) Non-trivial connected components of $\mathcal{C}(S)$.



(b) The non-trivial connected component of $\mathcal{C}(\mathcal{Q}_c)$.



(c) The non-trivial connected components of $\mathcal{C}(A_\varphi)$ involving an edge of the frame.

Fig. 8 Illustration of the non-trivial connected components of **a** the crossing graph of the split gadget S and **b** the crossing graph of the clause gadget $\mathcal{Q}_c$. **c** The non-trivial connected components of $\mathcal{C}(A_\varphi)$ involving an edge of the variable cycle (left) and of the clause cycle (right)

clockwise traversing C (refer to Figs. 7a and b); the case in which these edges are encountered in the reverse order in Γ being symmetric. Recall that v_i is the vertex shared by the edges h_i and d_i, for $i = 1, 2, 3$, and let Δ be the triangle induced by v_1, v_2, and v_3 in the underlying graph of S. Observe that, the edges of Δ are not involved in any crossings. Therefore, in Γ, the triangle Δ must lie either in the interior or in the exterior of C. W.l.o.g., we assume that Δ lies in the interior of C, the case in which Δ lies in the exterior of C being symmetric. It follows that, for $i = 1, 2, 3$, the two paths composed of the edges h_i, g_i and the edges d_i, c_i, respectively, lie in the interior of C. This is due to the fact that these paths are incident to v_i and that their edges do not cross the edges of C. This, in turn, implies that the paths $\pi_{i,3}$ and $\psi_{i,3}$, with $i \in \{1, 2\}$, also lie in the interior of C.

Consider the drawing Γ' obtained by removing from Γ the paths $\pi_{i,3}$ and $\psi_{i,3}$, with $i \in \{1, 2\}$. We show two properties of Γ':

Property (a) Δ bounds an internal face of Γ';
Property (b) v_1, v_2, and v_3 appear in this order when clockwise traversing Δ.

To prove Property (a), recall that Δ lies inside C by hypothesis and that its edges cannot be crossed. Moreover, since the edges b_i and f_i must cross the edge l_i, for $i = 1, 2, 3$, the paths (b_i, c_i, d_i) and (f_i, g_i, h_i) must be drawn in Γ' in the exterior of Δ. Therefore, Δ bounds an internal face of Γ'. To prove Property (b), recall that l_1, l_2, and l_3 are encountered in this order when clockwise traversing C by hypothesis and that the edge b_i crosses the edge l_i, for $i = 1, 2, 3$. Therefore, since the paths b_i, c_i, d_i

and b_j, c_j, d_j, with $i \neq j$, do not cross each other, we have that v_1, v_2, and v_3 must appear in this order when clockwise traversing Δ.

Properties (a) and (b) allow us to define the following regions. Denote by $R_{1,2}$ the bounded region of the plane incident to the edge (v_1, v_2) and to the vertex shared by l_1 and l_2. Analogously, denote by $R_{2,3}$ the bounded region of the plane incident to the edge (v_2, v_3) and to the vertex shared by l_2 and l_3 (see the gray region in Fig. 7a and b), and denote by $R_{1,3}$ the bounded region of the plane incident to the edge (v_3, v_1) and to the vertex shared by l_3 and l_1.

To prove the statement it suffices to show that when clockwise traversing C the following sequences of edges may not occur (i) b_2, f_2, f_3, and b_3; (ii) f_2, b_2, b_3, and f_3; (iii) b_1, f_1, f_3, and b_3; (iv) f_1, b_1, b_3, and f_3. Suppose, for a contradiction, that case (i) occurs; the other cases being analogous. Since Γ (and thus Γ') is simple and since c_i crosses g_i, for $i = 2, 3$, we have that the edges $g_3, c_3, d_3, (v_3, v_2), d_2, c_2, g_2$ are encountered in this order when clockwise traversing the boundary of $R_{2,3}$. Consider now the two paths $\pi_{2,3}$ and $\psi_{2,3}$ in Γ. Since the edges $\pi''_{2,3}$ and $\psi''_{2,3}$ cross in Γ, we have that the endpoints of $\pi_{2,3}$ alternate with the endpoints of $\psi_{2,3}$ in an Eulerian clockwise tour of the drawing of $\pi_{2,3} \cup \psi_{2,3}$ in Γ. This, in particular, implies that when clockwise traversing the boundary of $R_{2,3}$ either the edges $\pi'_{2,3}$, $\psi'_{2,3}$, $\pi'''_{2,3}$, and $\psi'''_{2,3}$ are encountered in this order or in the order $\psi'_{2,3}$, $\pi'_{2,3}$, $\psi'''_{2,3}$, and $\pi'''_{2,3}$. Recall that, in Γ, the edge $\pi'_{2,3}$, $\psi'_{2,3}$, $\pi'''_{2,3}$, and $\psi'''_{2,3}$ must cross the edge c_2, g_2, c_3, and g_3, respectively. However, this contradicts the fact that g_3, c_3, c_2, and g_2 appear in this order when clockwise traversing $R_{2,3}$. Thus, the statement follows. □

Variable gadget. For each variable v of degree k in φ and incident to clauses $c_1, c_2, \ldots, c_k$ in G_φ, the *variable gadget* $\mathcal{V}_v$ is an AT-graph defined as follows; refer to Fig. 7c. Assume, w.l.o.g., that $c_1, c_2, \ldots, c_k$ appear in this clockwise order around v in $\mathcal{E}_\varphi$. The underlying graph of $\mathcal{V}_v$ is composed of k split gadgets $S_1, S_2, \ldots, S_k$. For each split gadget S_i, with $i = 1, \ldots, k$, rename the edges a_j and e_j of S_i as a^i_j and e^i_j, respectively, with $j \in \{1, 2, 3\}$. For $i = 1, \ldots, k$, we identify the edges a^i_j and a^{i+1}_{j+1} and the edges e^i_j and e^{i+1}_{j+1}, where $k + 1 = 1$. The crossing graph $\mathcal{C}(\mathcal{V}_v)$ of $\mathcal{V}_v$ consists of all vertices and edges of the crossing graphs of S_i, with $i = 1, \ldots, k$. Moreover, for $i = 1, \ldots, k$, it contains a non-trivial connected component consisting of the single edge (a^i_2, e^i_2), which coincides with (a^{i+1}_3, e^{i+1}_3), $i + 1 = 1$ when $i = k$.

Lemma 5 *In any simple realization of $\mathcal{V}_v$ together with the variable cycle of v in which both a^i_1 and e^i_1 cross e_{v,c_i}, for $i = 1, \ldots, k$, the clockwise circular order of the edges crossing the variable cycle of v in $\mathcal{V}_v$ is either a^1_1, e^1_1, a^2_1, e^2_1,..., a^k_1, e^k_1 or e^1_1, a^1_1, e^2_1, a^2_1,..., e^k_1, a^k_1.*

Proof Let C be the variable cycle of v in H_φ, which consists of the edges e_{v,c_i}, for $i = 1, \ldots, k$. Consider a simple realization Γ of $\mathcal{V}_v \cup C$ in which both a^i_1 and e^i_1 cross e_{v,c_i}, for $i = 1, \ldots, k$, and the edges $(e_{v,c_1}, e_{v,c_2}, \ldots, e_{v,c_k})$ appear in this order when clockwise traversing C (refer to Fig. 7c); the case in which these edges are encountered in the reverse order in Γ being symmetric.

Consider the graph $\mathcal{V}^-_v := \mathcal{V}_v \setminus \bigcup_{i=1}^k \{a^i_1, e^i_1\}$. We have that, for any two edges e and e' of $\mathcal{V}^-_v$, there exists a chain $r_1, \ldots, r_h$ of edges of $\mathcal{V}^-_v$, such that $r_1 = e$, $r_h = e'$,

and, for $i = 1, \ldots, h-1$, it holds that either r_i and r_{i+1} are adjacent or r_i and r_{i+1} cross. This property and the fact that no edge of $\mathcal{V}_v^-$ crosses an edge of C implies that $\mathcal{V}_v^-$ lies either entirely in the interior or entirely in the exterior of C in Γ. W.l.o.g, we assume that $\mathcal{V}_v^-$ lies in the interior of C, the case in which it lies in the exterior of C being symmetric. To prove the statement, it suffices to show that, for $i = 1, \ldots, k-1$, either $e_1^i, a_1^i, e_1^{i+1}, a_1^{i+1}$ appear in this clockwise order around C or such edges appear the clockwise order $a_1^i, e_1^i, a_1^{i+1}, e_1^{i+1}$ around C.

Consider a pair of consecutive split gadget S_i and S_{i+1}, for $i = 1, \ldots, k-1$, and let Γ' be the drawing of the graph $\mathcal{N} := S_i \cup S_{i+1}$ in Γ. By Lemma 4, for $i = 1, \ldots, k$, the edges $e_1^i, a_1^i, e_2^i, a_2^i, e_3^i$, and a_3^i appear in this clockwise order or in the clockwise order $a_1^i, e_1^i, a_2^i, e_2^i, a_3^i$, and e_3^i around the outer cycle of S_i. This leaves us with four possible combinations of orders for the edges exiting $\mathcal{N}$. However, since $e_2^i = e_3^{i+1}$ crosses $a_2^i = a_3^{i+1}$, we have that in a clockwise visit of the external face of Γ' only two combinations are feasible; refer to Fig. 7c. Namely, we either encounter the endpoints of e_1^i, a_1^i, e_1^{i+1}, and a_1^{i+1} in this order or the endpoints of a_1^i, e_1^i, a_1^{i+1}, and e_1^{i+1} in this order. This concludes the proof. □

In the proof of Theorem 1, the two circular orders for the edges $\bigcup_{i=1}^{k} \{a_1^i, e_1^i\}$ of $\mathcal{V}_v$ considered in Lemma 5 will correspond to the two possible truth assignment of the variable v.

Clause gadget. For each clause c in φ, the *clause gadget* $\mathcal{Q}_c$ is the AT-graph, whose construction is inspired by a similar gadget used in [21], defined as follows; see Figs. 9a to 9c. The underlying graph of $\mathcal{Q}_c$ consists of six length-3 paths: For $v \in \{x, y, z\}$, we have a path formed by the edges (a_v, b_v, c_v) (red paths in Figs. 9a to 9c and a path formed by the edges (e_v, f_v, g_v) (blue paths in Figs. 9a–c. The crossing graph $\mathcal{C}(\mathcal{Q}_c)$ of $\mathcal{Q}_c$ consists of one non-trivial connected component (see Fig. 8b) formed by the triangles (c_x, c_y, c_z) and (g_x, g_y, g_z), and the edges (c_x, g_z), (c_y, g_x), and (c_z, g_y).

Lemma 6 *The clause gadget $\mathcal{Q}_c$ admits a simple realization together with the clause cycle of c in which, for $v \in \{x, y, z\}$, both b_v and f_v cross $e_{c,v}$, and in which the edges $e_{c,x}$, $e_{c,y}$, and $e_{c,z}$ appear in this order when traversing clockwise the clause cycle of c if and only if for at least one pair f_v, b_v, with $v \in \{x, y, z\}$, we have that b_v precedes f_v along $e_{c,v}$ when traversing the clause cycle of c clockwise.*

Figures 9a to 9c show that, up to a renaming of the variables x, y, and z, if for at least one pair f_v, b_v, with $v \in \{x, y, z\}$, we have that b_v precedes f_v along $e_{c,v}$ when traversing the clause cycle of c clockwise, then the clause gadget $\mathcal{Q}_c$ admits a simple realization together with the clause cycle c in which, for $v \in \{x, y, z\}$, both b_v and f_v cross $e_{c,v}$, and in which the edges $e_{c,x}$, $e_{c,y}$, and $e_{c,z}$ appear in this order when traversing clockwise the clause cycle of c.

For the opposite direction, consider a simple realization Γ of $\mathcal{Q}_c$ together with the clause cycle c in which, for $v \in \{x, y, z\}$, both b_v and f_v cross $e_{c,v}$, and in which the edges $e_{c,x}$, $l_{c,y}$, and $l_{c,z}$ appear in this order when traversing clockwise the clause cycle of c. Also, assume that, in Γ, there exists no pair f_v, b_v, with $v \in \{x, y, z\}$, such that b_v precedes f_v along $e_{c,v}$ when traversing the clause cycle of c clockwise; refer to Fig. 9d.

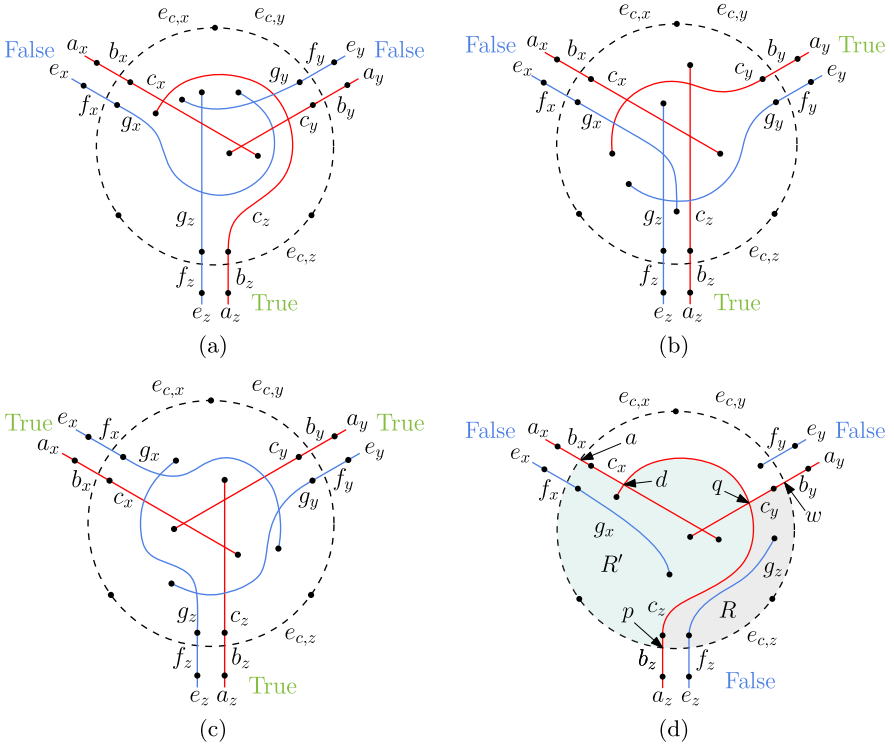


Fig. 9 The clause gadget. **a**–**c** Illustrations for the existence of simple realizations of the clause gadget $\mathcal{Q}_c$ together with the clause cycle of c (dashed edges) when for at least one pair f_v, b_v, with $v \in \{x, y, z\}$, we have that b_v precedes f_v along $e_{c,v}$, while traversing the clause cycle of c clockwise. **d** Partial illustration of an unsatisfied clause

Consider the region R delimited by the following curves: (i) the part of b_z between its intersection with $e_{c,z}$, which we denote by p, and its common endpoint with c_z, (ii) the part of c_z between such an endpoint and the intersection of c_z with c_y, which we denote by q, (iii) the part of c_y between q and its common endpoint with b_y, (iv) the part of b_y between such an endpoint and the intersection of b_y with $e_{c,y}$, which we denote by w, (v) the part of $e_{c,y}$ between w and its common endpoint with $e_{c,z}$, and (vii) the part of $e_{c,z}$ between such an endpoint and p. Observe that the edges whose parts compose the boundary of R, that is, the edges $e_{c,y}$, $e_{c,z}$, b_z, c_z, c_y, and b_y may not be crossed by g_z. Also, since f_z precedes b_z along $e_{c,z}$ when traversing the clause cycle of c clockwise, we have that the common endpoint of f_z and g_z lies in R. It follows that the drawing of g_z in Γ lies in the interior of R.

Consider now the region R' delimited by the following curves: (i) the part of b_z between p and its common endpoint with c_z, (ii) the part of c_z between such an endpoint and the intersection of c_z with c_x, which we denote by d, (iii) the part of c_x between d and its common endpoint with b_x, (iv) the part of b_x between such an endpoint and the intersection of b_x with $e_{c,x}$, which we denote by a, (v) the part of $e_{c,x}$ between a and its common endpoint with $e_{c,z}$, and (vii) the part of $e_{c,z}$ between its

common endpoint with $e_{c,x}$ and p. Observe that the edges whose parts compose the boundary of R', that is, the edges $e_{c,x}$, $e_{c,z}$, b_z, c_z, c_x, and b_x may not be crossed by g_x. Also, since f_x precedes b_x along $e_{c,x}$ when traversing the clause cycle of c clockwise, we have that the common endpoint of f_x and g_x lies in R'. It follows that the drawing of g_x in Γ lies in the interior of R'.

Since, by definition, R and R' are interior disjoint, and since we have shown that, in Γ, the drawing of g_z lies in R and the drawing of g_x lies in R', it follows that g_x and g_z cannot cross in Γ, which contradicts the fact that Γ is a simple realization of $\mathcal{Q}_c$ satisfying the conditions of the statement. □

Based on Lemma 6, we associate the `True` value with a literal of a variable $v \in \{x, y, z\}$ appearing in c when b_v precedes f_v along $e_{c,v}$ while traversing the clause cycle of c clockwise, and `False` otherwise; see Fig. 9. We can finally prove the main result of the section.

Theorem 1 SATR *is* NP*-complete for instances* A *with* $\lambda(A) = 6$.

Proof We give a reduction from the NP-complete problem 3- CONNECTED PLANAR 3- SAT [19]. Let φ be an instance of 3- CONNECTED PLANAR 3- SAT. We construct an instance $A_\varphi = (G', \mathcal{X}')$ of SATR that is simply realizable if and only if φ is satisfiable. We initialize $G' = H_\varphi$ and $\mathcal{X}' = \emptyset$. Then, for each variable v, we extend A_φ to include $\mathcal{V}_v$ as follows: For each clause c_i that contains a literal of v, add to $\mathcal{X}'$ the pair of edges $\{a_1^i, e_{v,c_i}\}$ and $\{e_1^i, e_{v,c_i}\}$, where a_1^i and e_1^i belong to $\mathcal{V}_v$, and e_{v,c_i} belongs to H_φ. Also, for each clause c, we extend A_φ to include $\mathcal{Q}_c$ as follows: For each variable $v \in \{x, y, z\}$ whose literals belong to c, we add to $\mathcal{X}'$ the pair of edges $\{f_v, e_{c,v}\}$ and $\{b_v, e_{c,v}\}$, where f_v and b_v belong to $\mathcal{Q}_c$, and $e_{c,v}$ belongs to H_φ. Finally, for each occurrence of a literal of a variable v to a clause c_i, we identify edges of $\mathcal{V}_v$ with edges of $\mathcal{Q}_v$ as follows: If v appears as a positive (resp. negated) literal in c_i, then we identify the edge a_1^i of $\mathcal{V}_v$ with the edge a_y (resp. e_y) of $\mathcal{Q}_v$ and we identify the edge e_1^i of $\mathcal{V}_v$ with the edge e_y (resp. a_y) of $\mathcal{Q}_v$. Observe that we do not allow the edges a_1^i and e_1^i to cross. Clearly, A_φ can be constructed in polynomial time. The equivalence between A_φ and φ immediately follows from Lemmata 5 and 6, and from the fact that, by Lemma 3, in any simple realization of A_φ, all the variable cycles and all the clause cycles maintain the same circular orientation.

Observe that the size of the largest connected component of the crossing graph $\mathcal{C}(A_\varphi)$ is six; see also Fig. 8. As discussed in the introduction, this implies NP-hardness for any $\lambda(A) = k$, for any $k \geq 6$. In particular, such bound is achieved by a connected component of $\mathcal{C}(S)$, for each split gadget S used to construct $\mathcal{V}_v$, for any variable v, and by a connected component of $\mathcal{C}(\mathcal{Q}_c)$, for any clause c. The component of $\mathcal{C}(S)$ of size six is defined by the edges c_3, g_3, $\pi'''_{1,3}$, $\psi'''_{1,3}$, $\pi'''_{2,3}$, and $\psi'''_{2,3}$ (see Fig. 7a and b). The component of $\mathcal{C}(\mathcal{Q}_c)$ of size six is defined by the edges c_r and g_r, with $r \in \{x, y, z\}$ (see Fig. 9).

Finally, to show that SATR is in NP, observe that one can test whether an instance $A = (G, \mathcal{X})$ admits a simple realization, by considering all possible orders of the crossings of the edges of G that form the pairs in $\mathcal{X}$, run a linear-time planarity testing of the corresponding planarization of G, and return `YES` if and only if at least one such tests succeeds. This concludes the proof that SATR is NP-complete. □

We immediately obtain the following.

Corollary 1 SATR *is* NP*-complete for instances* A *with* $\lambda(A) \geq k$*, for any* $k \geq 6$.

Proof The statement follows by observing that, in the proof of Theorem 1, starting from an instance with $\lambda(A) = k$ and $k \geq 6$, one can iteratively increase $\lambda(A)$ by 1 by introducing a new edge whose endpoints have degree 1 that only crosses an arbitrary edge that corresponds to a vertex of a component of $\mathcal{C}(A)$ of maximum size. This clearly results in an equivalent instance with $\lambda(A) = k + 1$. □

We also remark that the NP-hardness of Theorem 1 and Corollary 1 holds for instances whose crossing graph is planar, and has maximum degree 3 and treewidth 3; see Fig. 8. Finally, since our reduction yields instances whose size is linear in the size of the input (planar) 3-SAT formula, we have the following.

Corollary 2 *Unless Exponential Time Hypothesis (ETH) fails,* SATR *has no* $2^{o(\sqrt{n})}$*-time algorithm, where* n *is the number of vertices of the input AT-graph.*

Proof By the *Sparsification Lemma* [16], it holds that 3- SAT cannot be solved in $2^{o(N+M)}$ time, where N and M are the number of variables and clauses of the formula, respectively, unless the Exponential Time Hypothesis (ETH) fails. On the other hand, there exists a polynomial-time reduction from 3- SAT to PLANAR 3- SAT that transforms 3- SAT instances with $N + M$ variables and clauses into equivalent PLANAR 3- SAT instances with $O((N + M)^2)$ variables and clauses [25]. Moreover, there is a reduction from PLANAR 3- SAT to 3- CONNECTED PLANAR 3- SAT that only requires a linear blow-up in the size of the instance [19]. Altogether, given an instance of 3- SAT with $N + M$ variables and clauses, there is a polynomial-time transformation that maps such instance into an equivalent instance of 3- CONNECTED PLANAR 3- SAT with $O((N + M)^2)$ variables and clauses. Finally, starting from an instance of 3- CONNECTED PLANAR 3- SAT, our reduction constructs a graph whose number of vertices and edges is $n + m \in O(n) = O((N + M)^2)$. Therefore, an algorithm solving SATR in $2^{o(\sqrt{n})}$ time would contradict ETH. □

4 A Linear-Time Algorithm for AT-Graphs A with $\lambda(A) \leq 3$

In this section we show that the problem SATR can be solved in linear-time for AT-graphs A with $\lambda(A) \leq 3$; see Theorem 2. We first give a short high-level overview of the overall strategy but note that proper definitions will only be given later in the detailed description of the algorithm. The first step is to reduce SATR to a constrained embedding problem where each vertex v may be equipped with alternation constraints that restrict the allowed orders of its incident edges around v. Next, we further reduce to the biconnected variant of the embedding problem which leads to new types of alternation constraints. It will turn out that many of these constraints can be transformed into constraints that can be expressed in terms of PQ-trees and are therefore easier to handle. Finally, we show that, when no further such transformations are possible, all the remaining alternation constraints have a simple structure that allows for an efficient test.

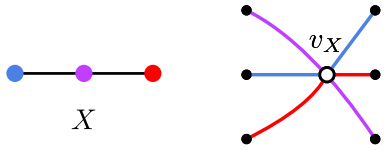


Fig. 10 Illustration of the construction of H

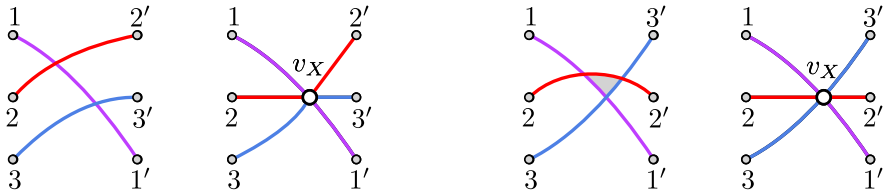


Fig. 11 Illustration of how to obtain a planar drawing of H from a simple realization of A (and vice versa)

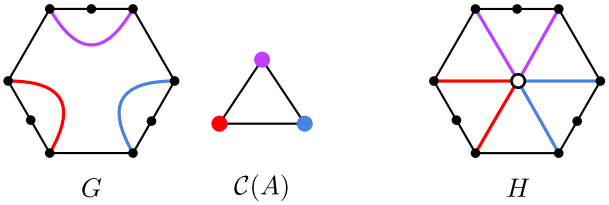


Fig. 12 A non-realizable AT-graph $A = (G, \mathcal{X})$ whose auxiliary graph H is planar

4.1 The Constrained Embedding Problem

We now start with reducing SATR to a constrained embedding problem. Let $A = (G, \mathcal{X})$ be an n-vertex AT-graph such that $\lambda(A) \leq 3$. We construct from G an auxiliary graph H as follows. For each connected component X of $\mathcal{C}(A)$ that is not an isolated vertex, denote by $E(X)$ the set of edges of G corresponding to the vertices of X, and by $V(X)$ the vertices of G that are end-vertices of the edges in $E(X)$. Remove from G the edges in $E(X)$ and add a *crossing vertex* v_X adjacent to all vertices in $V(X)$; see Fig. 10. Since no two crossing vertices are adjacent, the graph H does not depend on the order in which we apply these operations. The edges incident to v_X are partitioned into pairs of edges where two edges (a, v_X) and (b, v_X) form a pair if (a, b) is an edge of G corresponding to a vertex of X. We call (a, v_X) and (b, v_X) the *portions* of (a, b) and say that (a, v_X) and (b, v_X) *stem from* (a, b). Note that since $\lambda(A) \leq 3$, a crossing vertex v_X has either degree 4 or 6. In the first case X is a K_2 and in the latter case X is either an induced 3-path P_3 or a triangle K_3. If $X = K_2$, we color its two vertices red and blue, respectively. If $X = K_3$ we color its three vertices red, blue, and purple, respectively. If $X = P_3$, its vertex of degree 2 is colored purple, whereas we color red and blue the remaining two vertices, respectively. Based on Lemma 1, we observe the following.

Observation 1 *If A admits a simple realization, then H is planar.*

Proof By Lemma 1, we can consider an embedding of G induced by a simple realization of A in which all 3-crossings are untangled. In this embedding, replace each

crossing with a dummy vertex. For components of $\mathcal{C}(A)$ of size 2, the dummy vertex corresponds to the crossing. For components of $\mathcal{C}(A)$ of size 3, we distinguish whether the component is a path or a 3-cycle; see Fig. 11. In the case of a path, we can contract the edge between the two dummy vertices to obtain the corresponding crossing vertex. For a 3-cycle, by Lemma 1 we can further assume that the triangle enclosed by the edges connecting the three dummy vertices is empty, and we can therefore contract them into a single vertex. □

Observation 1 gives an immediate necessary condition for the realizability of A, which is, however, not sufficient; see Fig. 12. Indeed, a planar embedding of H obtained from contracting edges in a realization of A satisfies an additional property: for each crossing vertex v_X the portions stemming from two distinct edges e, f of G alternate around v_X *if and only* if the two vertices corresponding to e, f in X are adjacent; see Fig. 11. To keep track of this requirement, we equip every crossing vertex v_X with an *alternation constraint* that (i) colors its incident edges with colors r(ed), b(lue), p(urple) so that a portion of an edge in G gets the same color as the corresponding vertex in X, and (ii) specifies which pairs of colors must alternate around v. For a K_2*-constraint* there are no purple edges, and red and blue must alternate. For a K_3*-constraint* all pairs of colors must alternate. For a P_3*-constraint*, red and purple as well as purple and blue must alternate, whereas red and blue must not alternate. Each component X of $\mathcal{C}(A)$ with the coloring described above naturally translates to a constraint for v_X. For $X = K_2$, we obtain a K_2-constraint, for $X = P_3$ we get a P_3-constraint, and for $X = K_3$ we get a K_3-constraint. We say that H is a graph *with alternation constraints*; i.e.; some of the vertices in H have alternation constraints (while others may not). The auxiliary graph H with alternation constraints is *feasible* if it admits a planar embedding that satisfies the alternation constraints of all vertices with such a constraint. Thus, we have the following.

Lemma 7 *An AT-graph $A = (G, \mathcal{X})$ with $\lambda(A) \leq 3$ is simply realizable if and only if the corresponding auxiliary graph H with alternation constraints is feasible.*

Proof Assume that A is simply realizable. By Observation 1 the auxiliary graph H is planar. Recall that in a planar embedding of H obtained from contracting edges in a realization of A for each crossing vertex v_X the portions stemming from two distinct edges e, f of G alternate around v_X if and only if the two vertices corresponding to e, f in X are adjacent; see Fig. 11. Thus H is feasible.

Conversely, assume that H is feasible. To obtain a realization of A from such an embedding of H we do the following; see Fig. 11. First remove all crossing vertices and join all pairs of edges stemming from the same edge e in G to a new curve representing e. This yields a drawing Γ of G in which two edges cross if and only if they are adjacent in $\mathcal{C}(A)$. Note that in general Γ contains triples of edges that cross in a common point. By perturbing such edges in a small disk around their intersection point, we obtain a simple realization of A. □

To find such an embedding, we decompose the graph into biconnected components. It turns out that this may create additional types of alternation constraints that stem from the constraints described above, but do not fall into the category of an existing class

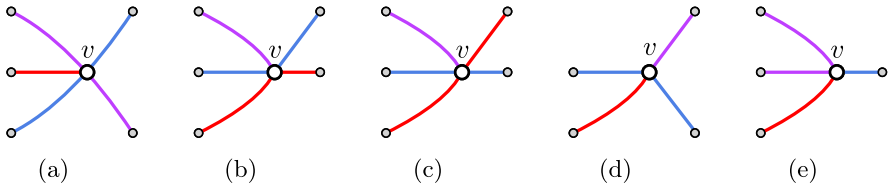


Fig. 13 Circular orders of edges incident to a vertex v satisfying **a** a K_3^{-r}- and a P_3^{-r}-constraint, **b** a P_3^{-p}- but not a K_3^{-p}-constraint, **c** a K_3^{-p}- but not a P_3^{-p}-constraint, **d** a $P_3^{-(p,r)}$- but not a $K_3^{-(p,r)}$-constraint, **e** a $P_3^{-(b,r)}$- and a $K_3^{-(b,r)}$-constraint

of constraints. For the sake of exposition, we introduce these constraints now, even though they will not be part of an instance obtained by the above reduction from SATR.

Let v be a vertex of degree 5 and let c be a color. For a C-constraint ($C \in \{K_3, P_3, K_2\}$) as defined above, we define a corresponding C^{-c}-*constraint* of v, which (i) colors the edges incident to v such that each color occurs at most twice but color c occurs only once and (ii) requires that in the rotation, it is possible to insert an edge of color c so that the original C-constraint is satisfied; see Fig. 13. Observe that a K_2^{-c}-constraint is always satisfied and is thus not needed. Since the colors of a K_3-constraint are entirely symmetric, we may assume without loss of generality that $c = r$ in this case. For P_3-constraints, only red and blue are symmetric, i.e., we may assume without loss of generality that either $c = p$ or $c = r$.

In particular, the K_3^{-r}-constraint and the P_3^{-r}-constraint both require that purple and blue alternate around v, whereas the position of the red edge is arbitrary; see Fig. 13a. Thus the K_3^{-r}-constraint and the P_3^{-r}-constraint are equivalent. For a P_3^{-p}-constraint to be fulfilled, red and blue must not alternate and the purple edge either has to be between the two red edges or between the two blue edges; see Fig. 13b.

Now let v be a vertex of degree 4 and let c, c' be two colors. For a C-constraint, we define a corresponding $C^{-c,c'}$-*constraint* of v, which (i) colors the edges incident to v such that the colors distinct from c and c' occur twice but colors c and c' occur only once if $c \neq c'$, or not at all if $c = c'$, and (ii) requires that in the rotation, it is possible to insert two edges of color c and c', respectively, so that the original C-constraint is satisfied. Since the colors of a K_3-constraint are entirely symmetric, we may assume w.l.o.g. that either $c = c' = r$ or $c = r, c' = b$ in this case. For P_3-constraints, only red and blue are symmetric, we may hence assume without loss of generality that $(c, c') \in \{(r, r), (p, p), (r, p), (r, b)\}$. Observe that a $K_2^{-c,c'}$-constraint is always satisfied and is thus not needed. The same holds for a $K_3^{-r,b}$-constraint, a $P_3^{-r,b}$-constraint and a $P_3^{-r,p}$-constraint; see Fig. 14. Also, note that a $K_3^{-r,r}$-constraint and a $P_3^{-r,r}$-constraint are both equivalent to a K_2-constraint, while a $P_3^{-p,p}$-constraint requires that red and blue do not alternate around v.

Finally for a C-constraint, we define a corresponding $C^{-(c,c')}$-*constraint* of v, which (i) colors the edges incident to v such that the colors distinct from c and c' occur twice but colors c and c' occur only once if $c \neq c'$, or not at all if $c = c'$, and (ii) requires that in the rotation, it is possible to insert an edge of color c and an edge of color c' *consecutively*, so that the C-constraint is satisfied; see Fig. 13d-e for examples.

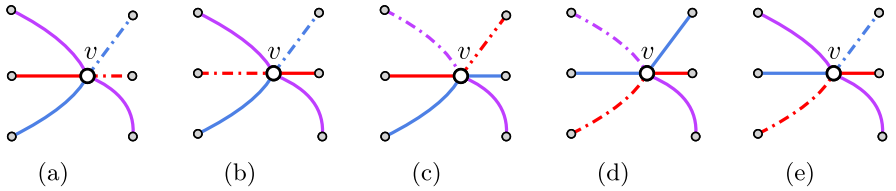


Fig. 14 Circular orders of edges around a vertex v allowing to insert two edges of distinct colors (dashed) so that every color occurs twice and a **a**-**b** K_3-constraint, **c**–**e** P_3-constraint is satisfied

Table 1 An overview of all non-trivial constraints

Constraint	Satisfied if and only if
K_3	all three colors pairwise alternate
P_3	red and purple alternate, and blue and purple alternate, and red and blue do not alternate
K_3^{-r}, P_3^{-r}	blue and purple alternate
P_3^{-p}	red and blue do not alternate
K_2, $K_3^{-r,r}$, $P_3^{-r,r}$, $P_3^{-(r,p)}$	the two blue edges alternate with the two remaining edges
$P_3^{-p,p}$	red and blue do not alternate
$K_3^{-(r,b)}$, $P_3^{-(r,b)}$	red and blue are consecutive

This type of constraints is motivated as follows. Let v be a cut vertex in a graph G. The *cut components* of v in G are the subgraphs of G induced by v together with the maximal subsets of the vertices of G that are not disconnected by the removal of v. Note that the edges belonging to two different cut components cannot alternate around v without resulting in a crossing and observe that a $K_2^{-(c,c')}$-constraint with $c \neq c'$ is always satisfied and is thus not needed. Also note that a $C^{-(c,c)}$-constraint cannot be satisfied, since every C-constraint requires that every color alternates with at least one of the remaining colors. Since the colors of a K_3-constraint are entirely symmetric, we may assume without loss of generality that $c = r, c' = b$ in this case. For P_3-constraints, only red and blue are symmetric, i.e., we may assume without loss of generality that $(c, c') \in \{(r, p), (r, b)\}$. In particular, a $K_3^{-(r,b)}$-constraint and a $P_3^{-(r,b)}$-constraint both require the consecutivity of the two purple edges and are thus equivalent; see Fig. 13e. For a $P_3^{-(r,p)}$-constraint to be fulfilled, the two blue edges must not occur consecutively (see Fig. 13d; i.e., the two blue edges have to alternate with the two remaining edges. Thus a $P_3^{-(p,r)}$-constraint is equivalent to a K_2-constraint. Table 1 gives an overview over all types of constraints. By the above discussion we may assume that only K_3, P_3, K_3^{-r}, P_3^{-p}, K_2, $P_3^{-p,p}$ and $K_3^{-(r,b)}$ constraints occur.

The ALTERNATION- CONSTRAINED PLANARITY (ACP) problem has as input a graph H with alternation constraints and asks whether H is feasible.

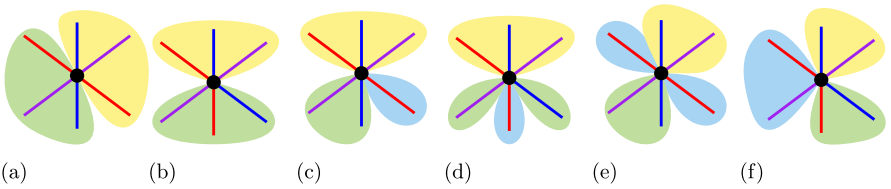


Fig. 15 Examples of suitable merges of cut components satisfying a **a** K_3-constraint in case (3, 3), **b** P_3-constraint in case (3, 3), **c** K_3-constraint in case (3, 2, 1), **d** P_3-constraint in case (3, 2, 1), **e** K_3-constraint in case (2, 2, 2), **f** P_3-constraint in case (2, 2, 2)

By Lemma 7, there is a linear-time reduction from SATR with $\lambda(A) \leq 3$ to ACP. Next, we further reduce ACP to 2- CONNECTED ACP, which is the restriction of ACP to instances for which every connected component of H is 2-connected.

Lemma 8 *There is a linear-time algorithm that either recognizes that an instance H of* ACP *is a no-instance or computes an equivalent instance H' of* 2- CONNECTED ACP.

Proof Our reduction strategy considers one cut vertex at a time and splits the graph at that vertex into a collection of connected components. The reduction consists of applying this cut vertex split until all cut vertices are removed or we find out that H is a no-instance. Consider an instance H of ACP and one of its cut vertices v with cut components $H_1, \ldots, H_l$. In the cut components, let every vertex except v preserve its alternation constraint (if any). Now the goal is to find out which constraints have to be assigned to the copies of v in the cut components such that H is a yes-instance if and only if the union of the H_is is a yes-instance. We denote by $E(v)$ the edges incident to v in H and by $E_i(v)$ the edges incident to v in H_i, for $1 \leq i \leq l$. Without loss of generality assume that $|E_i(v)| \geq |E_j(v)|$ for $1 \leq i < j \leq l$. We encode the distribution of edges from $E(v)$ among the cut components as a *split-vector* $(|E_1(v)|, |E_2(v)|, \ldots, |E_l(v)|)$.

If v has no alternation constraint, H admits a planar embedding that satisfies all alternation constraints if and only if each cut component H_i with $i = 1, \ldots, l$ does and hence all copies of v remain unconstrained. Otherwise v has an alternation constraint C. Observe that this implies $4 \leq |E(v)| \leq 6$ and thus the edges in $E(v)$ are distributed among at least two and at most six cut components. Note that the edges belonging to two different cut components cannot alternate around v without resulting in a crossing. Thus H is a no-instance if $C \in \{K_3, K_3^{-r}, K_2\}$ and there are two cut components containing a pair of edges of the same color from $E(v)$, respectively. If $C = P_3$, the same holds if one cut component contains both purple edges whereas a distinct cut component contains both red or both blue edges. In the following, we assume that the above does not apply. Now we consider cases based on the split-vectors. The first case is generic and covers all split vectors with $|E_1(v)| \leq 3$. Since $4 \leq |E(v)| \leq 6$, the only remaining cases are (5, 1), (4, 2), (4, 1, 1) and (4, 1). We treat each of them separately.

Case: $|E_1(v)| \leq 3$. Note that if $|E_i(v)| = 1$ for every $i > 1$, then we can always arrange the cut components around v such that C is satisfied. Now assume that at least two of the cut components contain more than one edge from $E(v)$; i.e., $E_2(v) \geq 2$. We

are allowed to (i) mirror the order of the edges in a cut component around v and to (ii) insert the ordered edges in a cut component consecutively into the order of the edges in a different cut component. It is easy to see that if there are only two cut components H_1, H_2, we get all cyclic orders of edges around v where no pair of edges of H_1 alternates with a pair of edges from H_2; see Fig. 15a-b for examples. Next observe that the same holds if there are more cut components and all but H_1, H_2 contain exactly one edge; see Fig. 15c-d for examples. It remains to consider the split-vector (2, 2, 2); i.e., v has degree 6 and $C \in \{K_3, P_3\}$. In this case, we get all cyclic orders of edges around v where no pair of edges of a cut component alternates with a pair of edges from a different cut component. Consider $C = P_3$. Recall that we assume that no cut component contains both purple edges whereas a distinct cut component contains both red or both blue edges. If one component contains both red (or both blue) edges, then the remaining components both contain one blue and one purple edge. Since red has to alternate with purple but must not alternate with blue, C cannot be satisfied without alternating cut components; i.e., H is a no-instance. In all remaining cases with split-vector (2, 2, 2), we can always arrange the cut components around v such that C is satisfied (see Fig. 15e for an example), since we assume that there are no two cut components containing a pair of edges of the same color from $E(v)$, respectively. In all positive cases, it suffices to leave each copy of v unconstrained. In the following, we consider the remaining split-vectors with $|E_1(v)| \geq 4$.

Case: (5, 1). Let $C \in \{K_3, P_3\}$ be the constraint of v and let c be the color of the edge of $E(v)$ in H_2. To merge embeddings of H_1 and H_2 to a planar embedding of H such that the C-constraint is satisfied, it is necessary that the embedding of H_1 allows to insert an edge of color c such that the C-constraint is satisfied. Thus it is necessary that the order of edges around v in H_1 satisfies a C^{-c}-constraint. Note that if the C^{-c}-constraint is satisfied, it is guaranteed that the embeddings of H_1 and H_2 can be merged such that the original C-constraint is satisfied. Thus it is necessary and sufficient to equip the copy of v in H_1 with a C^{-c}-constraint whereas the copy of v in H_2 remains unconstrained.

Case: (4, 2). Let $C \in \{K_3, P_3\}$ be the constraint of v and let c, c' be the colors of the edges of $E(v)$ in H_2. To merge embeddings of H_1 and H_2 to a planar embedding of H such that the C-constraint is satisfied, it is necessary that the embedding of H_1 allows to insert two edges of color c, c' consecutively, such that the C-constraint is satisfied. Thus, it is necessary that the order of edges around v in H_1 satisfies a $C^{-(c,c')}$-constraint. Note that if the $C^{-(c,c')}$-constraint is satisfied, the embeddings of H_1 and H_2 can always be merged such that the original C-constraint is satisfied. Thus it is necessary and sufficient to equip the copy of v in H_1 with a $C^{-(c,c')}$-constraint whereas the copy of v in H_2 remains unconstrained.

Case: (4, 1, 1). Let $C \in \{K_3, P_3\}$ be the constraint of v and let c, c' be the colors of the edges of $E(v)$ in H_2 and H_3, respectively. To merge embeddings of H_1 and H_2 to a planar embedding of H such that the C-constraint is satisfied, it is necessary that the embedding of H_1 allows to insert two edges of color c, c', such that the C-constraint is satisfied. Thus it is necessary that the order of edges around v in H_1 satisfies a $C^{-c,c'}$-constraint. Note that if the $C^{-c,c'}$-constraint is satisfied, it is guaranteed that the embeddings of H_1 and H_2 can be merged such that the original C-constraint

is satisfied. Thus it is necessary and sufficient to equip the copy of v in H_1 with a $C^{-c,c'}$-constraint whereas the copies of v in H_2 and H_3 remain unconstrained.

Case: (4, 1). Let C^{-c} with $C \in \{K_3, P_3\}$ be the constraint of v and let c' be the color of the edge in $E_2(v)$. To merge embeddings of H_1 and H_2 to a planar embedding of H that satisfies the C^{-c}-constraint, it is necessary that the embedding of H_1 allows to insert an edge of color c, such that the $C^{-c,c'}$-constraint is satisfied. Thus it is necessary that the order of edges around v in H_1 satisfies a $C^{-c,c'}$-constraint. Note that if the $C^{-c,c'}$-constraint is satisfied, the embeddings of H_1 and H_2 can always be merged such that the original C^{-c}-constraint is satisfied. Thus it is necessary and sufficient to equip the copy of v in H_1 with a $C^{-c,c'}$-constraint whereas the copy of v in H_2 remains unconstrained.
We may assume that after a linear-time preprocessing every edge in H is labeled with the block it belongs to. Then for a cut vertex v a split as described above takes $O(\deg(v))$-time. When no cut vertex is left, we return the obtained constrained blocks. □

4.2 Algorithm for the Embedding Problem

We define a more general problem General Alternation- Constrained Planarity (GACP) whose input is a graph H where vertices of degree 4, 5, or 6 may be equipped with an alternation constraint or with a (synchronized) PQ-tree (*but not both*). The question is whether H admits a planar embedding such that all alternation constraints are satisfied (i.e., H is feasible) and the order of edges around a vertex with a PQ-tree B is compatible with B. The 2- connected GACP problem is the restriction of GACP to input graphs whose connected components are 2-connected. Clearly, every instance of (2- connected) ACP is an instance of (2- connected) GACP. For our purpose, however, it will turn out that PQ-tree constraints are easier to handle. Thus, given an instance of ACP we aim to construct an equivalent instance of GACP, where as many alternation constraints as possible are replaced by PQ-trees. In particular, alternation constraints of degree-4 vertices can be replaced by the PQ-trees shown in Fig. 16.

Lemma 9 *Given an instance of* 2- connected GACP *containing a degree-*4 *vertex* v *with alternation constraint* C*, we obtain an equivalent instance by removing* C *and equipping* v *with a suitable (synchronized) PQ-tree.*

Proof Let C be the constraint of v. We distinguish cases based on C.
Case: C *is a* K_2*-constraint*. The constraint is fulfilled if and only if red and blue alternate, which can be represented by a single Q-node as shown in Fig. 16a.
Case: C *is a* $P_3^{-p,p}$*-constraint*. The constraint is fulfilled if and only if red and blue do not alternate. Hence it is necessary and sufficient that the two red edges appear consecutively, which can be enforced by the PQ-tree in Fig. 16b.
Case: C *is a* $K_3^{-(r,b)}$*-constraint*. The constraint is fulfilled if and only if a red and a blue edge can be inserted consecutively, such that all colors pairwise alternate. This is the case if and only if the two purple edges appear consecutively, which can be enforced by the PQ-tree in Fig. 16c. □

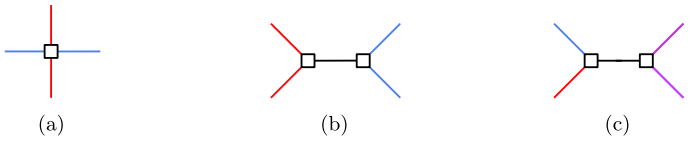


Fig. 16 The PQ-trees representing alternation constraints of degree-4 vertices. **a** K_2-constraint, **b** $P_3^{-p,p}$-constraint, and **c** $K_3^{-(r,b)}$-constraint

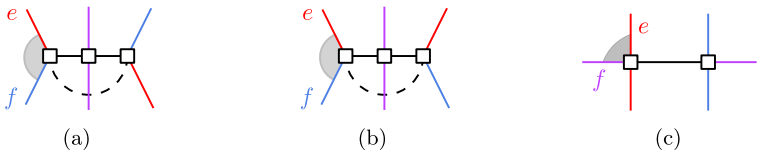


Fig. 17 The (synchronized) PQ-trees for degree-6 vertices with consecutive edge pair and **a** K_3-constraint, **b**, **c** P_3-constraint

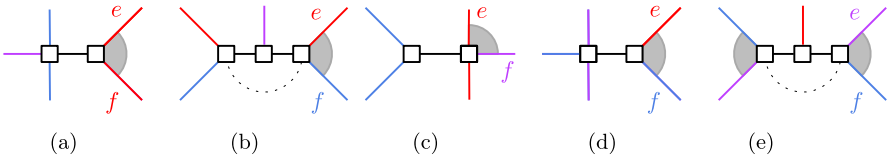


Fig. 18 The (synchronized) PQ-trees for degree-5 vertices with consecutive edge pair and **a**–**c** P_3^{-p}-constraint, **d**–**e** K_3^{-r}-constraint

Hence we may assume from now on that no vertex with an alternation constraint in H has degree less or equal to 4; i.e., all these vertices have degree 5 or 6.

Let v be a vertex with alternation constraints. We call two edges e, f incident to v a *consecutive edge pair*, if they are consecutive (around v) in *every* planar embedding of H that satisfies all constraints. We show that, in most cases, an alternation constraint at a vertex incident to a consecutive edge pair can be replaced by a PQ-tree.

Lemma 10 *Let H be an instance of* 2- CONNECTED GACP *and let v be a vertex in H with alternation constraint C incident to a consecutive edge pair. Then H is either a no-instance, or we obtain an equivalent instance H' by removing C and equipping v with a suitable (synchronized) PQ-tree if*

(i) $C \neq K_3^{-r}$ *or*
(ii) $C = K_3^{-r}$ *and one of the consecutive edges is red or*
(iii) $C = K_3^{-r}$ *and v is incident to two distinct consecutive edge-pairs.*

Proof Let e, f be a consecutive edge pair. Note that if $\deg(v) = 6$, we may assume that e, f have distinct colors, since otherwise H is a no-instance as every color has to alternate with at least one other color in order to satisfy a K_3- or P_3-constraint.
Case: C is a K_3-constraint. Without loss of generality assume that e and f are red and blue, respectively. Then the orders of edges around v in which e, f appear consecutively and the alternation constraint is satisfied, can be represented by the synchronized PQ-tree shown in Fig. 17a.

Case: C is a P_3-constraint. First consider the case that e and f are red and blue. Then the synchronized PQ-tree shown in Fig. 17b represents all allowed orders around v. Otherwise, if e is purple we may assume that f is red and label v with the PQ-tree shown in Fig. 17c.
Case: C is a P_3^{-p}-constraint. If e and f have the same color, say red, we label v with the PQ-tree shown in Fig. 18a. If e and f are red and blue, respectively, we label v with the synchronized PQ-tree shown in Fig. 18b. Finally, if e is purple, then f has a different color, say red. Then we label v with the PQ-tree shown in Fig. 18c.
Case: C is a K_3^{-r}-constraint and one of the consecutive edges is red. In this case e and f must be of distinct colors and we may assume without loss of generality that e is red and f is blue. Then we label v with the PQ-tree shown in Fig. 18d.
Case: C is a K_3^{-r}-constraint and v is incident to two distinct consecutive edge pairs. If the red edge is contained in a consecutive pair, we are in the previous case. Hence assume this is not the case. Note that H is a no-instance if a consecutive edge pair contains two edges of the same color. Hence we may assume that both consecutive edge pairs $\{e, f\}$ and $\{e', f'\}$ contain a blue and a purple edge. If the two consecutive pairs are disjoint, then we label v with the synchronized PQ-tree shown in Fig. 18e. Otherwise, we may assume that $f = f'$. But then the two pairs enforce that $\{e, e', f\}$ is consecutive, and consequently we obtain that the remaining two edges, one of which is red, also form a consecutive pair. We can thus apply the previous case. □

The overall strategy of the remaining section consists of three steps. In Step 1 we identify consecutive edge pairs in H that allow us to replace alternation constraints by PQ-trees with the help of the SPQR-tree of H. Then we show how to compute an equivalent instance that does not contain P_3- or K_3^{-r}-constraints. Finally, we end up with an instance whose alternation constraints are all K_3-constraints and every vertex with such a constraint appears in the skeletons of exactly two P-nodes and one S-node in the SPQR-tree.

In Step 2, we handle such constraints by considering them on a more global scale. We show that they form cyclic structures, where either the constraints cannot be satisfied or can be dropped and satisfied irrespective of the remaining solution. Eventually, we arrive at an instance with only (synchronized) PQ-trees as constraints, which we solve with standard techniques in Step 3.

Step 1: Consecutive Edge Pairs

For the rest of this section, let H be an instance of 2- CONNECTED GACP and let T be the SPQR-tree of H. We begin with Step 1 and identify consecutive edge pairs. Let μ be a node of T and let v be a vertex of its skeleton incident to the virtual edges $e_1, \ldots, e_k$. Then the *distribution vector* $(d_1, \ldots, d_k)$ of v, with $d_i \geq d_{i+1}$ for every $1 \leq i < k$, contains for each virtual edge e_i the number d_i of edges from $E(v)$ contained in e_i.

Lemma 11 *Let H be an instance of* 2- CONNECTED GACP *and let T be the SPQR-tree of H. A vertex v with alternation constraint C in H is incident to a consecutive edge pair if v appears in a skeleton of T that*

(i) has a virtual edge that contains exactly two edges from $E(v)$,
(ii) has a virtual edge that contains all but two edges from $E(v)$, or
(iii) is an R-node.

Moreover, a vertex v of degree 5 is incident to two consecutive edge pairs if v appears in a skeleton of T that

(iv) has two virtual edges, each of which contains exactly two edges from $E(v)$, or
(v) is an R-node that has at least four virtual edges incident to v.

Proof It is easy to see that in cases (i) and (ii) there is a consecutive edge pair. For (iii) consider a vertex v in the skeleton of an R-node and assume that neither (i) nor (ii) applies. Recall that v has degree 5 or 6 and that the vertices in the skeleton of an R-node have degree at least 3; i.e., the edges of $E(v)$ are distributed among at least three virtual edges. Thus it remains to consider the distribution vectors (1, 1, 1, 1, 1, 1), (1, 1, 1, 1, 1), (3, 1, 1, 1). Recall that the order of the virtual edges around a vertex in the skeleton of an R-node is fixed up to reversal. In all three cases, there are two edges f, g appearing in two virtual edges e_f, e_g, respectively, such that e_f, e_g both contain exactly one edge from $E(v)$ and appear consecutively in every planar embedding of the skeleton of the R-node. Thus, $\{f, g\}$ is a consecutive edge pair.

It is easy to see that in case (iv) there are two consecutive edge pairs. For (v) consider a vertex v in the skeleton of an R-node μ and assume that (iv) does not apply. Thus it remains to consider the distribution vectors (2, 1, 1, 1) and (1, 1, 1, 1, 1). Recall that the order of the virtual edges around a vertex in the skeleton of an R-node is fixed up to reversal. In both cases there are three virtual edges e, e', e'', each of which contains exactly one edge from $E(v)$ that appear consecutively around v with e' between e and e'' in every planar embedding of the skeleton of v. Let f, g, h be the corresponding edges in $E(v)$ such that $f \in e, g \in e', h \in e''$. Then $\{f, g\}$ and $\{g, h\}$ are two consecutive edge pairs. □

Since we immediately replace alternation constraints by PQ-trees whenever we find consecutive edge pairs, we assume from now on that no vertex with alternation constraint satisfies one of the conditions of Lemma 11. We will show later that this implies that every vertex that has an alternation constraint shows up in the skeleton of a P-node. Thus, in the following we focus on vertices v with alternation constraints that show up in the skeleton of a P-node and show how to get rid of their alternation constraints. By the assumption that none of these vertices satisfies the conditions of Lemma 11, we can exclude distribution vectors that contain two 2s as well as distribution vectors with sum 6 that contain a 2. It thus remains to consider vertices that appear in P-nodes with the distribution vectors (1, 1, 1, 1, 1, 1), (1, 1, 1, 1, 1), (3, 1, 1, 1), (3, 1, 1) and (2, 1, 1, 1). For each of these cases we have to distinguish three subcases, as the second pole u of the P-node is either unconstrained, has a PQ-constraint, or also has an alternation-constraint. In the following we show that in most cases we can compute an equivalent instance with fewer alternation constraints in constant time and thus may assume that these cases do not occur; see Fig. 19 for an overview. We start by showing that in the case of a (1, 1, 1, 1, 1, 1) or a (1, 1, 1, 1, 1) distribution vector we can get rid of the alternation

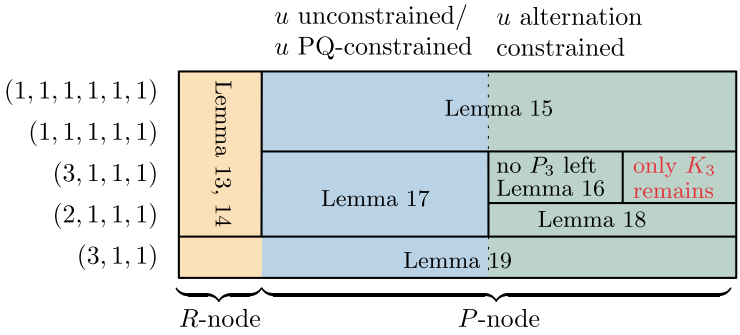


Fig. 19 Overview of which cases are handled by which lemma

constraint C of v as it is either always possible to reorder the children of μ according to C within a realization of H without C, or H without C (and thus H) is not realizable.

Lemma 12 *Let v be a vertex in H with alternation constraint C that appears in a P-node μ in T with distribution vector $(1, 1, 1, 1, 1, 1)$ or $(1, 1, 1, 1, 1)$. There is a constant-time algorithm that either recognizes that H is a no-instance, or computes an equivalent instance H' with fewer alternation constraints by removing C and possibly other constraints.*

Proof Let u be the pole of μ distinct from v. If u is unconstrained let H' be the instance we obtain by removing C from H. Since every solution for H' remains a valid solution if we reorder the children of μ according to C, H and H' are equivalent.

Now assume that u also has an alternation constraint C'. Recall that u and v have either degree 5 or 6. If $\deg(u) \neq \deg(v)$, say $\deg(u) = 6, \deg(v) = 5$, then there is a virtual edge in the skeleton of μ that contains exactly two edges e, f from $E(u)$; a contradiction to the assumption that no degree-6 vertex with alternation constraint is incident to a consecutive edge pair. Thus $\deg(u) = \deg(v)$ and we check whether one of the constantly many orderings of the virtual edges satisfies both C and C'. In the negative case, H is a no-instance; in the positive case, let H' be the instance we obtain by removing both C and C' from H. Clearly, every solution of H is a solution for H'. Note that every solution for H' remains valid if we reorder the children of μ according to one of the orders that satisfy both C and C'. Thus H and H' are equivalent.

It remains to consider the case that u has a PQ-tree B. We may assume without loss of generality that B contains the consecutivities induced by the virtual edges in the skeleton of μ. If $\deg(u) = \deg(v)$, we check whether one of the constantly many orderings of the virtual edges satisfies C and is compatible with B. In this case, similarly to the discussion above, we get an equivalent instance H' by removing C. Otherwise, $\deg(v) = 5$ and $\deg(u) = 6$, i.e., v has distribution vector $(1, 1, 1, 1, 1)$ in μ, whereas u has distribution vector $(2, 1, 1, 1, 1)$ in μ. Let e, f be the two edges from $E(u)$ contained in the same virtual edge e_1. Then, similar to the case above, we check if one of the constantly many orderings of the virtual edges satisfies C and is compatible with B. In the negative case, H is a no-instance. In the positive case we get an equivalent instance H' by removing C from H, as every solution for H' remains

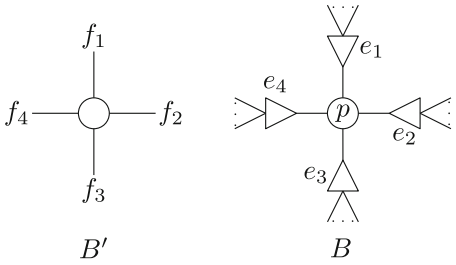


Fig. 20 Illustrations of the PQ-trees used in the proof of Lemma 14. Triangles represent subtrees of B

valid if we reorder the children of μ according to one of the orders that satisfies both C and B and, if necessary, flip the embedding of the subgraph represented by e_1. □

Similar techniques allow us to get rid of P_3-constraints.

Lemma 13 *Let v be a vertex in H with P_3-constraint C that appears in a P-node μ in T with distribution vector* $(3, 1, 1, 1)$. *Then there is a constant-time algorithm that either recognizes that H is a no-instance, or computes an equivalent instance H' with fewer alternation constraints by replacing C by a suitable (synchronized) PQ-tree.*

Proof Note that H is a no-instance if e_1 contains either both or none of the purple edges from $E(v)$. Hence assume that e_1 contains exactly one purple edge f from $E(v)$. If e_1 contains a red edge g and a blue edge h from $E(v)$, then no planar embedding of H with f between g and h around v satisfies C, since then purple cannot alternate with both blue and red such that red and blue do not alternate. Thus g, h are a consecutive edge pair and C can be replaced by a suitable (synchronized) PQ-tree by Lemma 10. Otherwise, assume without loss of generality that e_1 contains one purple and two red edges from $E(v)$. In this case, in any planar embedding of H that satisfies C, the purple edge f has to appear between the two red edges. Thus f forms a consecutive edge pair with each of the red edges and thus again by Lemma 10, C can be replaced by a suitable (synchronized) PQ-tree. □

Next we show that if only one pole of a P-node has an alternation constraint, then we can also get rid of this alternation constraint in case of a $(3, 1, 1, 1)$ or a $(2, 1, 1, 1)$ distribution.

Lemma 14 *Let v be a vertex in H with alternation constraint $C \in \{K_3, K_3^{-r}\}$ that appears in a P-node μ in T with distribution vector* $(3, 1, 1, 1)$ *or* $(2, 1, 1, 1)$. *Let u be the pole of μ distinct from v. If u is unconstrained or has a (synchronized) PQ-constraint, there is a constant-time algorithm that either recognizes that H is a no-instance, or computes an equivalent instance H' with fewer alternation constraints by either removing C or replacing C with a suitable (synchronized) PQ-tree.*

Proof Observe that in case of a $(3, 1, 1, 1)$ distribution vector, $C = K_3$ and the three edges from $E(v)$ in e_1 must have pairwise distinct colors; otherwise H is a no-instance. If the distribution vector of v is $(2, 1, 1, 1)$, $C = K_3^{-r}$ and the two edges from $E(v)$ in e_1 must be purple and blue; otherwise H is a no-instance or the alternation constraint can be replaced by a PQ-tree by Lemma 10. If u is unconstrained, let H' be the instance

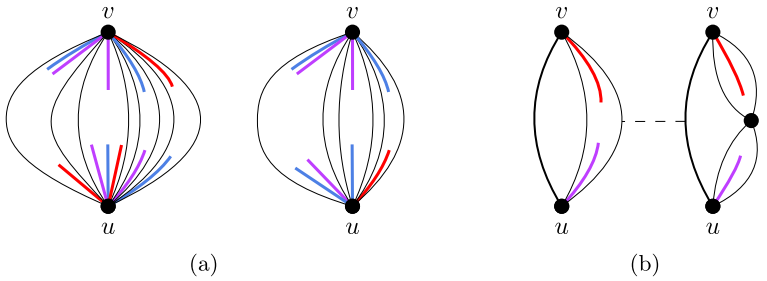


Fig. 21 Illustrations of operations used in the proof of Lemma 15

we obtain from H by removing C. Clearly, every solution of H is also a solution for H'. Note that every solution for H' remains a valid solution if we reorder the children of μ according to C. Thus, H is a yes-instance if and only if H' is a yes-instance.

Now assume that u has a (synchronized) PQ-constraint and let B be the (synchronized) PQ-tree of u. Note that we may assume without loss of generality that B contains the consecutivities induced by the virtual edges in the skeleton of μ. Let e_1, e_2, e_3, e_4 be the four virtual edges incident to u in the skeleton of μ and let f_1, f_2, f_3, f_4 be four edges from $E(u)$ such that for every $1 \leq i \leq 4$, f_i is contained in e_i. We set $F = \{f_1, f_2, f_3, f_4\}$ and denote the restriction of B to F by B'. Observe that B' has four leaves and thus contains at most two non-leaf nodes. Moreover, note that in case B' is synchronized, there is an equivalent PQ-tree containing exactly one non-leaf node of type Q, thus we may assume that B' is not a synchronized PQ-tree. If B' is trivial, i.e., it contains exactly one non-leaf node that is of type P, then we let H' be the instance we obtain from H by removing C. Clearly, every solution of H is also a solution of H'. Conversely, note that the structure of B' implies that in B there is a P-node p of degree 4 such that for every virtual edge e_i in the skeleton of μ, there is an edge in B incident to p whose removal yields two subtrees B_1, B_2 such that the leaf-set of B_1 corresponds exactly to the edges from $E(u)$ in e_i; see Fig. 20. Hence the embeddings of the subgraphs represented by the virtual edges in the skeleton of μ are completely independent. In other words, every solution of H' remains a valid solution after reordering the children of μ according to C, since B is compatible with each of these orders. Thus H is a yes-instance if and only if H' is a yes-instance.

Otherwise, there is an edge pair f_i, f_j in F that is consecutive in every order represented by B'. Note that also $F \setminus \{f_i, f_j\}$ is consecutive in every order represented by B', and that this is also a consecutive a pair of edges, since B' has only four leaves. Hence we may assume without loss of generality that $i, j \neq 1$. In particular, we get that two of the virtual edges containing exactly one edge from $E(v)$ have to appear consecutively in any planar embedding of H that satisfies all constraints. This yields a consecutive edge pair around v. Note that in case $C = K_3^{-r}$ the two edges of $E(v)$ in e_1 are a second consecutive edge pair. Thus, independent of its type, C can be replaced by a suitable (synchronized) PQ-tree by Lemma 10. □

In case of a (2, 1, 1, 1) distribution in a P-node μ with two alternation constrained poles, we get rid of both alternation constraints in constant time.



Fig. 22 The PQ-trees used in the proof of Lemma 15

Lemma 15 *Let v be a vertex in H with K_3^{-r}-constraint C that appears in a P-node μ in T with distribution vector $(2, 1, 1, 1)$. Let u be the pole of μ distinct from v. If u also has an alternation constraint C', then we can compute an equivalent instance H' with fewer alternation constraints in constant time.*

Proof Since u has an alternation constraint, it has either degree 5 or 6. Let e_1^v be the virtual edge in μ that contains more than one edge from $E(v)$ and let e_1^u be the virtual edge in μ that contains more than one edge from $E(u)$. First assume $e_1^v \neq e_1^u$. Let H' be the instance we obtain from H by removing C. Clearly every solution of H is also a solution for H'. Note that every solution for H' remains valid if we reorder the children of μ according to C' and, if necessary, flip the embedding of e_1^v in order to satisfy C; see Fig. 21a for an illustration.

If $e_1^v = e_1^u$, let e_r denote the virtual edge in μ that contains the red edge in $E(v)$. Let G_1 be the graph represented by e_r in μ plus a uv-edge. Let G_2 be the graph represented by the virtual edges distinct from e_r in μ. Note that both G_1 and G_2 are biconnected. Let H' be the instance we obtain from H by (i) replacing the biconnected component containing v by G_1 and G_2 such that every vertex except for the copies of u, v keeps its constraint (if any) and (ii) equipping the copy of v in G_2 with the PQ-tree shown in Fig. 22a and (iii) equipping the copy of u in G_2 with a suitable constraint depending on its degree and the distribution of its edges. The constraint for u depends on the degree of u. First assume $\deg(u) = 5$. Then u has a K_3^{-r}-constraint in H and the two edges of $E(u)$ in e_1^v are blue and purple. If the red edge in $E(u)$ is contained in G_1, then we label the copy of u in G_2 with the PQ-tree shown in Fig. 22a. Otherwise we may assume without loss of generality that the copy of u in G_2 is incident to two purple edges and thus we label it with the PQ-tree shown in Fig. 22b. It remains to consider the case $\deg(u) = 6$. By Lemma 13 we may assume that u has a K_3-constraint in H and that the edges of $E(u)$ in e_1^v have pairwise distinct colors. As the edge colors are completely symmetric in this case, we may assume that the edge of $E(u)$ contained in e_r is red and we equip the copy of u in G_2 with a K_3^{-r}-constraint.

From every solution of H we obtain a solution of H' by copying the embedding of the connected component of v, removing unnecessary vertices and edges from the first copy and replacing unnecessary vertices and edges by a single uv-edge in the second copy. Conversely, assume that we have a solution for H'. Since the position of the red edge around v is arbitrary, it is always possible to reinsert the graph represented by e_r into an embedding of the remaining graph, such that both alternation constraints are fulfilled.

Note that we obtain the SPQR-tree T' of H' from T in constant-time as follows. Let ν be the node adjacent to μ that contains the twin edge of e_r. Note that ν is an S-node since v has degree 2 in the skeleton of ν. We remove e_r from μ and add a new Q-node adjacent to ν; see Fig. 21b for an illustration. □

Finally we show that every vertex that appears with a distribution vector (3, 1, 1) in the SPQR-tree, also appears in a P-node with distribution vector (2, 1, 1, 1). By Lemmata 15 and 14, we may thus assume that no vertex with alternation constraint appears with distribution vector (3, 1, 1) in T.

Lemma 16 *Let v be a vertex with alternation constraint C that appears in a node μ in T with distribution vector* (3, 1, 1). *Then there is a P-node ν in T that contains v with distribution vector* (2, 1, 1, 1).

Proof The distribution vector (3, 1, 1) implies that there is a node ν in T containing v, such that the three edges of $E(v)$ that appear in the same virtual edge in μ, are distributed among at least two virtual edges in ν. Moreover, there is a distinct virtual edge in the skeleton of ν that contains the two remaining edges from $E(v)$. Hence the distribution vector of v in ν is either (2, 2, 1) or (2, 1, 1, 1). In the first case we immediately get two consecutive edge pairs by Lemma 11(iv), thus we assume the latter case. Note that ν cannot be an S-node, since v has degree 4 in the skeleton of ν. If ν is an R-node, we have two consecutive edge pairs by Lemma 11 (v), hence v appears in a P-node with distribution vector (2, 1, 1, 1). □

Next we show that by applying Lemmata 10 to 14 we can assume from now on that only K_3-constraints and PQ-constraints occur and that every vertex with alternation constraint appears in a P-node in T with distribution vector (3, 1, 1, 1).

Lemma 17 *Let H be an instance of* 2-CONNECTED GACP. *There is a linear-time algorithm that either recognizes that H is a no-instance, or computes an equivalent instance H' such that every alternation constraint is a K_3-constraint and every vertex with K_3-constraint appears in the skeleton of a P-node with distribution vector* (3, 1, 1, 1).

Proof Let v be a vertex in H with alternation constraint C. First we show that v appears in a P-node in T. Assume that v appears in an R-node μ. Then by Lemmata 10 and 11 and since every vertex in the skeleton of an R-node has degree at least 3, we may assume that the distribution vector of v in μ is (3, 1, 1). Now by Lemma 16, v appears in a P-node. Next assume that v appears in an S-node ν in T. Since the vertices in the skeleton of an S-node have degree 2 and v has degree 5 or 6, there is a node μ adjacent to ν in T also containing v. Note that μ is either a P-node or an R-node, as there are no two adjacent S-nodes in an SPQR-tree. If $C \neq K_3^{-r}$, μ is a P-node by Lemma 11(iii). Otherwise, if $C = K_3^{-r}$ then μ is either a P-node or an R-node. In the latter case it follows by the above discussion that v appears in a P-node. Hence every vertex with alternation constraint appears in a P-node in T.

Recall that the vertices in the skeleton of a P-node have degree at least 3; i.e., the edges of $E(v)$ are distributed among at least three virtual edges. The only possible distributions without consecutive edge pairs that can be handled by Lemma 11, are (1, 1, 1, 1, 1, 1), (1, 1, 1, 1, 1), (2, 1, 1, 1), (3, 1, 1) and (3, 1, 1, 1). By Lemma 16 the case of a (3, 1, 1) distribution vector does not have to be considered and thus by applying Lemma 12, 14 and 15 exhaustively, we either reject or only the last case remains; see Fig. 19. Finally, by Lemma 13, a vertex v with alternation constraint that appears in a P-node with distribution vector (3, 1, 1, 1) has a K_3-constraint. □

Hence, we may assume that only K_3-constraints occur and that every vertex v with K_3-constraint appears in the skeleton of a P-node ν in T with distribution vector (3, 1, 1, 1), whose pole distinct from v also has a K_3-constraint. This concludes Step 1.

Step 2: Remaining Alternation Constraints

Let v be a vertex with K_3-constraint. The three edges from $E(v)$ contained in the same virtual edge in the skeleton of ν must have pairwise distinct colors; otherwise, H is a no-instance. Since there are no two adjacent P-nodes in an SPQR-tree and by assumption no vertex with alternation constraint appears in an R-node, v also appears in an S-node with distribution vector (3, 3). Let μ be an S-node in T that contains v. Since there are no two adjacent S-nodes in an SPQR-tree, for each neighbor u of v in the skeleton of μ, there is a P-node adjacent to μ in T with poles v, u. Thus, by assumption, the neighbors of v in the skeleton of μ also have a K_3-constraint. Observe that the distribution vector of the neighbors of v in μ cannot be (5, 1) as all neighbors of μ are R- or P-nodes; i.e., the vertices in the corresponding skeletons have degree at least 3. Hence the neighbors of v in μ also have distribution vectors (3, 3). Iteratively, it follows that every vertex in the skeleton of μ has a K_3-constraint and shares a P-node with each of its two neighbors.

Consider an S-node μ in T that contains alternation-constrained vertices $v_0, \ldots, v_{k-1}$ in this order; see Fig. 23. In the following, we consider the indices of the vertices and edges in μ modulo k. For every $0 \leq i < k$, we denote the virtual edge between v_i and v_{i+1} by e_i and let ν_i be the P-node adjacent to μ in T with poles v_i, v_{i+1}. Note that for every i, the virtual edge e in the skeleton of ν_i that contains three edges from $E(v_i)$ also contains three edges from $E(v_{i+1})$, since e is the virtual edge representing μ. Thus, if we fix the order of the three edges from $E(v_i)$ in e_i, this fixes the order of the three edges from $E(v_{i+1})$ in e_i. Since v_{i+1} has an alternation constraint, this also fixes the order of the edges from $E(v_{i+1})$ in e_{i+1}. In this way, a fixed order of the three edges from $E(v_1)$ in e_1 implies an order of the edges from $E(v_{k-1})$ in e_{k-1}, which in turn implies an order on the three edges from $E(v_1)$ in e_{k-1}. If there exists an order of the three edges from $E(v_1)$ in e_1 that implies an order of the remaining edges from $E(v_1)$ in e_{k-1} such that the K_3-constraint is satisfied, we obtain an equivalent instance by removing all K_3-constraints of vertices in the skeleton of μ, since we can reorder the parallels adjacent to ν independently of the remaining graph. Otherwise, H is a no-instance. By the discussion above, we have the following.

Lemma 18 *Let μ be an S-node in T that contains vertices with K_3-constraint. There is an $O(\deg(\mu))$-algorithm that either recognizes that H is a no-instance, or computes an equivalent instance by removing all K_3-constraints of vertices in the skeleton of μ.*

Step 3: Solve Reduced Instance

Now we may assume that our graph H does not contain alternation constraints and start with Step 3. To solve such an instance, we expand each vertex with its associated PQ-tree, if any, into a gadget that allows the same circular orders of its incident edges as the

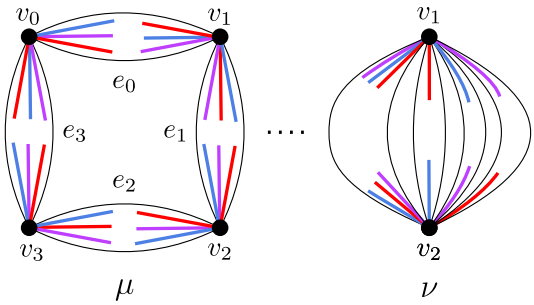


Fig. 23 An S-node μ and an adjacent P-node ν

PQ-tree (essentially a P-node becomes a normal vertex, whereas a Q-node expands into a wheel as described in [13]). Embedding the resulting graph $H^\star$ and then contracting the gadgets back into single vertices already ensures that for each vertex of H the order of its incident edges is represented by its PQ-tree. Since our synchronized PQ-trees only involve Q-nodes, it then suffices to ensure that synchronized Q-nodes are flipped consistently. To this end, observe that each wheel is 3-connected and hence its embedding is determined by a single R-node in the SPQR-tree of $H^\star$. This allows us to express such constraints in a simple 2-SAT formula of linear size, similar to, e.g., [1, 13]. Therefore, we obtain the following.

Theorem 2 *Let $A = (G, \mathcal{X})$ be an n-vertex AT-graph such that $\lambda(A) \leq 3$. There exists an $O(n)$-time algorithm that decides whether A is a positive instance of* SATR *and that, in the positive case, computes a simple realization of A.*

Proof By Lemmata 7 and 8, AT-graph realizability with $\lambda(A) \leq 3$ can be reduced to 2- CONNECTED GACP in linear time. Each of the resulting instances H of 2-CONNECTED GACP we solve independently as follows. First we compute an equivalent instance H' without alternation constraints by applying Lemmata 10,12 to 15, 17 and 18 exhaustively (and reject if necessary). Since in H', no vertex has an alternation constraint, a planar embedding of H' is feasible if and only if it satisfies the PQ-constraints. To solve such an instance, we expand each vertex with its associated PQ-tree, if any, into a gadget that allows the same circular orders of its incident edges as the PQ-tree (essentially a P-node becomes a normal vertex, whereas a Q-node expands into a wheel as described in [13]). Embedding the resulting graph $H^\star$ and then contracting the gadgets back into single vertices already ensures that for each vertex of H the order of its incident edges is represented by its PQ-tree. Since our synchronized PQ-trees only involve Q-nodes, it then suffices to ensure that synchronized Q-nodes are flipped consistently. To this end, observe that each wheel is 3-connected and hence its embedding is determined by a single R-node in the SPQR-tree of $H^\star$. This allows us to express such constraints in a simple 2-SAT formula of linear size, similar to, e.g., [1, 13]. If H' is a yes-instance, such a procedure also provides a feasible embedding of H', which can be obtained from an embedding of $H^\star$ that satisfies all the synchronization constraints by contracting each gadget to a single vertex. Starting from the embedding of H', by selecting suitable permutations for the children of the P-nodes (according to Lemmata 10, 12 to 15 and 18), we can obtain a feasible embedding of H. Similarly, feasible embeddings of each block of the input auxiliary graph can be combined, by

reordering the blocks around common cut vertices (according to Lemma 8), to obtain a feasible embedding Γ of it. To get a realization of A from Γ, we delete all crossing vertices and reinsert for every connected component X of the crossing graph $\mathcal{C}(A)$, the edges in $E(X)$.

Clearly, applying Lemmata 10,12,13,15 and 17 exhaustively takes linear time in total. Also applying Lemma 14 exhaustively can be done efficiently, if we store all vertices that initially satisfy the conditions of the lemma in a list L. When the alternation constraint of a vertex v in L is removed, we remove v from L and add all vertices to L that have an alternation constraint and appear in a P-node together with v. This way every vertex is processed only once. For applying Lemma 18 exhaustively, we iterate over all S-nodes in T. For each S-node we need time linear in its degree. This yields a linear running time in total. □

5 Conclusions and Open Problems

We proved that deciding whether an AT-graph A is simply realizable is NP-complete, already when the largest connected components of the crossing graph $\mathcal{C}(A)$ have 6 vertices only. On the other hand, we described an optimal linear-time algorithm that solves the problem when the largest components of $\mathcal{C}(A)$ have size at most 3. This is the first efficient algorithm for the Simple AT- graph Realizability problem that works on general graphs.

An open problem that naturally arises from our findings is filling the gap between tractability and intractability: What is the complexity of Simple AT- graph Realizability if $\lambda(A)$ is 4 or 5? A first issue is that Lemma 1 only allows to untangle cliques of size 3 and it is not clear whether a similar result can be proved for components of size 4. Furthermore, contracting larger crossing structures yields more complicated alternation constraints and it is not clear whether they can be turned into PQ-trees, similar to the case of components of size 3. We therefore feel that different techniques may be necessary to tackle the cases where $4 \leq \lambda(A) \leq 5$. Analogously, the hardness proof relies on the existence of a component of size 6, both in split and in the clause gadget. Reducing the size of either of these components, while requiring the realization to be simple, appears to be challenging.

Another interesting direction is to study alternative structural parameters under which the problem can be tackled, and which are not ruled out by our hardness result, as discussed in the introduction; for example the vertex cover number of $\mathcal{C}(A)$. Moreover, one can try to extend our approach to the "weak" setting (i.e., the Weak AT- graph Realizability problem), still requiring a simple realization.

Finally, it is interesting to study the complexity of the problem when the crossing graph belongs to some restricted graph families, such as trees or forests.

Acknowledgements Research by Da Lozzo, Didimo, Montecchiani, and Patrignani was supported, in part, by MUR of Italy (PRIN Project no. 2022ME9Z78 – NextGRAAL and PRIN Project no. 2022TS4Y3N – EXPAND). Research by Münch and Rutter was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 541433306. Research started at the 1st Summer Workshop on Graph Drawing, September 2021, Castiglione del Lago, Italy. An extended abstract of this work appeared in the

proceeding of the 35rd International Symposium on Algorithms and Computation (ISAAC 2024) [10]. The present submission contains detailed proofs of all claims.

Author Contributions All authors equally contributed to the research described in the paper and to the writing of the manuscript

Funding Open Access funding enabled and organized by Projekt DEAL.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Bläsius, T., Fink, S.D., Rutter, I.: Synchronized planarity with applications to constrained planarity problems. ACM Trans. Algorithms **19**(4), 34:1-34:23 (2023). https://doi.org/10.1145/3607474
2. Bläsius, T., Rutter, I.: Simultaneous PQ-ordering with applications to constrained embedding problems. ACM Trans. Algorithms **12**(2), 16:1-16:46 (2016). https://doi.org/10.1145/2738054
3. Booth, K.S., Lueker, G.S.: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. J. Comput. Syst. Sci. **13**(3), 335–379 (1976). https://doi.org/10.1016/S0022-0000(76)80045-1
4. Booth, K.S.: PQ-tree Algorithms. University of California, Berkeley (1975)
5. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing: Algorithms for the Visualization of Graphs. Prentice-Hall (1999)
6. Di Battista, G., Tamassia, R.: On-line maintenance of triconnected components with SPQR-trees. Algorithmica **15**(4), 302–318 (1996). https://doi.org/10.1007/BF01961541
7. Di Battista, G., Tamassia, R.: On-line planarity testing. SIAM J. Comput. **25**(5), 956–997 (1996). https://doi.org/10.1137/S0097539794280736
8. Didimo, W., Liotta, G., Montecchiani, F.: A survey on graph drawing beyond planarity. ACM Comput. Surv. **52**(1), 4:1-4:37 (2019). https://doi.org/10.1145/3301281
9. Diestel, R.: Graph Theory, 4th Edition, volume 173 of Graduate texts in mathematics. Springer, (2012)
10. Da Lozzo, G., Didimo, W., Montecchiani, F., Münch, M., Patrignani, M., Rutter, I.: Simple realizability of abstract topological graphs. In: 35th International Symposium on Algorithms and Computation (ISAAC'24), volume 322 of LIPIcs, pp. 23:1–23:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, (2024). https://doi.org/10.4230/LIPIcs.ISAAC.2024.23
11. Gassner, E., Jünger, M., Percan, M., Schaefer, M., Schulz, M.: Simultaneous graph embeddings with fixed edges. In: 32nd International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2006, volume 4271 of Lecture Notes in Computer Science, pp. 325–335. Springer, (2006). https://doi.org/10.1007/11917496_29
12. Goodman, J.E., O'Rourke, J.: editors. Handbook of Discrete and Computational Geometry, Second Edition. Chapman and Hall/CRC, (2004). https://doi.org/10.1201/9781420035315
13. Gutwenger, C., Klein, K., Mutzel, P.: Planarity testing and optimal edge insertion with embedding constraints. J. Graph Algorithms Appl. **12**(1), 73–95 (2008). https://doi.org/10.7155/JGAA.00160

14. Gutwenger, C., Mutzel, P.: A linear time implementation of SPQR-trees. In: 8th International Symposium on Graph Drawing, GD 2000, Proceedings, volume 1984 of LNCS, pp. 77–90. Springer, (2000). https://doi.org/10.1007/3-540-44541-2_8
15. Hong, S.-H.: Beyond planar graphs: Introduction. In: Beyond Planar Graphs, Communications of NII Shonan Meetings, pp. 1–9. Springer, (2020). https://doi.org/10.1007/978-981-15-6533-5_1
16. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? J. Comput. Syst. Sci. **63**(4), 512–530 (2001). https://doi.org/10.1006/jcss.2001.1774
17. John, E.: Hopcroft and Robert Endre Tarjan. Efficient planarity testing. J. ACM **21**(4), 549–568 (1974). https://doi.org/10.1145/321850.321852
18. Kratochvíl, J.: String graphs. II. Recognizing string graphs is NP-hard. J. Comb. Theory Ser. B **52**(1), 67–78 (1991). https://doi.org/10.1016/0095-8956(91)90091-W
19. Kratochvíl, J.: A special planar satisfiability problem and a consequence of its NP-completeness. Discret. Appl. Math. **52**(3), 233–252 (1994). https://doi.org/10.1016/0166-218X(94)90143-0
20. Kratochvíl, J., Lubiw, A., Nesetril, J.: Noncrossing subgraphs in topological layouts. SIAM J. Discret. Math. **4**(2), 223–244 (1991). https://doi.org/10.1137/0404022
21. Kratochvíl, J., Matoušek, J.: NP-hardness results for intersection graphs. Comment. Math. Univ. Carol. **30**(4), 761–773 (1989). (http://eudml.org/doc/17790)
22. Kratochvíl, J., Matousek, J.: Intersection graphs of segments. J. Comb. Theory, Ser. B, **62**(2), 289–315 (1994). https://doi.org/10.1006/JCTB.1994.1071
23. Kyncl, J.: Simple realizability of complete abstract topological graphs in P. Discret. Comput. Geom. **45**(3), 383–399 (2011). https://doi.org/10.1007/s00454-010-9320-x
24. Kyncl, J.: Simple realizability of complete abstract topological graphs simplified. Discret. Comput. Geom. **64**(1), 1–27 (2020). https://doi.org/10.1007/s00454-020-00204-0
25. Lichtenstein, D.: Planar formulae and their uses. SIAM J. Comput. **11**(2), 329–343 (1982). https://doi.org/10.1137/0211025
26. Menger, K.: Zur allgemeinen Kurventheorie. Fundam. Math. **10**(1), 96–115 (1927). (http://eudml.org/doc/211191)
27. Schaefer, M.: Toward a theory of planarity: Hanani-tutte and planarity variants. J. Graph Algorithms Appl. **17**(4), 367–440 (2013). https://doi.org/10.7155/JGAA.00298
28. Schaefer, M., Sedgwick, E., Stefankovic, D.: Recognizing string graphs in NP. J. Comput. Syst. Sci. **67**(2), 365–380 (2003). https://doi.org/10.1016/S0022-0000(03)00045-X
29. Schaefer, M., Stefankovic, D.: Decidability of string graphs. J. Comput. Syst. Sci. **68**(2), 319–334 (2004). https://doi.org/10.1016/J.JCSS.2003.07.002
30. Streinu, I., Bezdek, K., Pach, J., Dey, T.K., Chen, J., Kravets, D., Amato, N.M., Franklin, W.R.: Discrete and computational geometry. In: Handbook of Discrete and Combinatorial Mathematics. CRC Press, (1999). https://doi.org/10.1201/9781439832905.CH13

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.