Simple Realizability of Abstract Topological Graphs

Giordano Da Lozzo
Roma Tre University, Italy

Walter Didimo
University of Perugia, Italy

Fabrizio Montecchiani
University of Perugia, Italy

Miriam Münch
University of Passau, Germany

Maurizio Patrignani
Roma Tre University, Italy

Ignaz Rutter
University of Passau, Germany

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 classical result that both ATR and SATR are NP-complete [16, 19].

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 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.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Mathematics of computing → Graph algorithms; Human-centered computing → Graph drawings

Keywords and phrases Abstract Topological Graphs, SPQR-Trees, Synchronized PQ-Trees

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2024.23

Related Version *Full Version*: `http://arxiv.org/abs/2409.20108` [5]

Funding 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.

Acknowledgements Research started at the 1^{st} Summer Workshop on Graph Drawing, September 2021, Castiglione del Lago, Italy.

35th International Symposium on Algorithms and Computation (ISAAC 2024).
Editors: Julián Mestre and Anthony Wirth; Article No. 23; pp. 23:1–23:15
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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 crossing (a point where the two edges properly cross); (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 [18], establishing the complexity of the ATR (and of the SATR) problem has been the subject of an intensive research activity, 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, 15]. However, for $\mathcal{X} \neq \emptyset$, a seminal paper by Kratochvíl [16] 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 [25] and then to NP [24]. 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 [19, 20]. 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 [21, 22]. Refer to [21] for the complexity of other variants of ATR, and to [11] for a connection with the popular Simultaneous Graph Embedding problem.

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., [9, 12, 14, 26]), 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 $\lambda(A) \geq 6$); see Section 3. A consequence of our result is that, unless P = NP, the problem is not fixed-parameter tractable 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. As the results in [16, 19, 20], our hardness proof uses a reduction from 3-Connected Planar 3-SAT. However, the reduction in [16] does not deal with the simplicity of the realization, whereas the reduction in [19, 20] 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 Section 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 [21, 22]. 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 (in the rotation system), and a sequence of transformations that exploit the interplay between alternation constraints and the SPQR-tree [7, 8] 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, 23] and cannot be handled straightforwardly with PQ-trees.

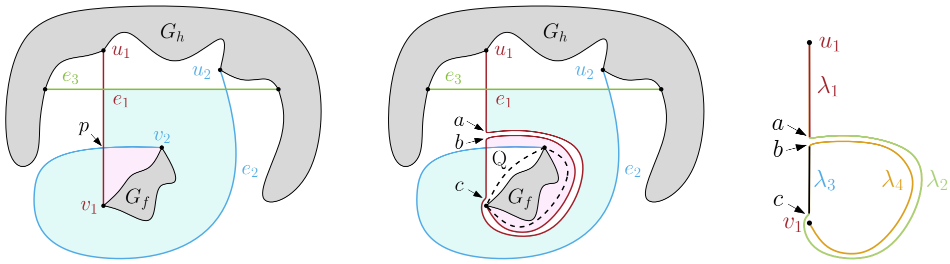
For proofs of lemmas and theorems marked with ($\star$) we refer to the full version [5].

2 Basic Definitions and Tools

For basic definitions about graphs and their drawings, refer to [6, 10]. 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 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 $2k$ vertices in $V(E')$ are incident to a common face (see Fig. 1b); otherwise, it is *tangled* (see Fig. 1a). The next lemma will turn useful in Section 4.

▶ **Lemma 1** ($\star$). *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.*



(a) Realization Γ_A. (b) Realization Γ'_A. (c) Curves of e_1.

Figure 1 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.

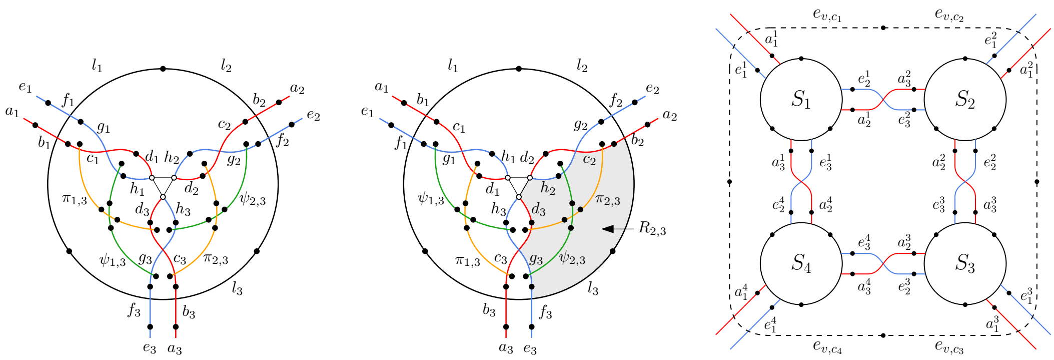
Proof Sketch. Let A be an AT-graph with $\lambda(A) \leq 3$ and let Γ_A be a simple realization of A that contains a tangled 3-crossing E'. We show how to obtain a new simple realization Γ'_A of A that coincides with Γ_A except for the drawing of one of the edges in E' and such that E' is untangled (refer to Fig. 1). Repeating such a transformation for each tangled 3-crossing yields the desired simple realization of A with no tangled 3-crossings.

Since Γ_A is simple and $|E'| = 3$, the arrangement $C_{E'}$ in Γ_A splits the plane into two faces, a bounded face f and an unbounded face h. 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, assume w.l.o.g. that f contains an endpoint of two edges of E' (and thus h contains the remaining four endpoints), that such endpoints are v_1 and v_2, and that traversing e_2 from u_2 to v_2, we see u_1 to the left at the intersection between e_2 and e_1. Let G_f (resp. G_h) be the subgraph of G formed by the vertices and edges of G in the interior of f (resp. of h). 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. 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 shown in Fig. 1c by following the drawing of e_1, e_2 and Q. Moving on e_2 from u_2 to v_2, we now see u_1 to the right at the intersection of e_2 with e_1. Hence, all the endpoints of the edges in E' lie in the same face of $C_{E'}$ in Γ'_A, i.e., E' is untangled in Γ'_A. ◀

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 6). We will exploit a reduction from the NP-complete problem 3-Connected Planar 3-SAT [17].
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



(a) **(b)** **(c)**

Figure 2 (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 skeleton H_φ.

literals appearing in a clause gadget encodes a `true` value, the clause gadget admits a simple realization. We start by describing the "skeleton" 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.

Skeleton. Arbitrarily choose a planar embedding $\mathcal{E}_\varphi$ of G_φ. The *skeleton* H_φ *of* φ is a 4-regular 3-connected plane graph obtained from $\mathcal{E}_\varphi$ as follows. For each degree-k variable v of φ, the graph H_φ contains a k-cycle formed by the sequence of edges $(e_{v,c_1}, e_{v,c_2}, \dots, e_{v,c_k})$, which we refer to as the *variable cycle* of v, where $c_1, \dots, c_k$ are the clause-nodes of G_φ adjacent to v in the clockwise order in which they appear around v in $\mathcal{E}_\varphi$.
For each clause c of φ, the graph H_φ contains a 3-cycle formed by the sequence of edges $(e_{c,v_1}, e_{c,v_2}, e_{c,v_3})$, which we refer to as the *clause cycle* of c, where v_1, v_2, v_3 are the variable-nodes of G_φ adjacent to c in the clockwise order in which they appear around c 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.

▶ **Lemma 2** ($\star$). *The skeleton H_φ obtained from $\mathcal{E}_\varphi$ is triconnected.*

Split gadget. The split gadget S is the AT-graph defined as follows; refer to Figs. 2a and 2b. 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 Figs. 2a and 2b) 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. 2a and 2b) and (e_i, f_i, g_i, h_i) (blue paths in Figs. 2a and 2b); (3) four length-3 paths $\pi_{1,3}$, $\pi_{2,3}$, $\psi_{1,3}$, and $\psi_{2,3}$. We denote the first, intermediate, and last edge of a length-3 path p as p', p'', and p''', respectively. The crossing graph $\mathcal{C}(S)$ of S consists of several connected components. Next, we describe the eight *non-trivial connected components* of $\mathcal{C}(S)$, i.e., those that are not isolated vertices, determined by the following crossings of the edges of S: (i) For $j = 1, 2, 3$,

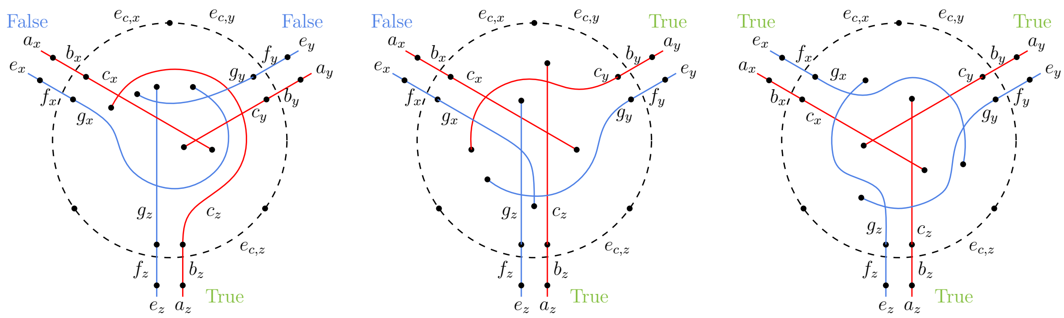


Figure 3 Illustrations for the existence of simple realizations of the clause gadget $\mathcal{Q}_c$ together with the clause cycle of c (dashed edges) when for at least one pair f_v, b_v, with $v \in \{x, y, z\}$, we have that b_v precedes f_v along $e_{c,v}$, while traversing the clause cycle of c clockwise.

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 $\psi'''_{2,3}$.

▶ **Lemma 3** ($\star$). *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.*

Variable gadget. For each variable v of degree k in φ and incident to clauses $c_1, c_2, \dots, c_k$ in G_φ, the *variable gadget* $\mathcal{V}_v$ is an AT-graph defined as follows; refer to Fig. 2c. Assume, w.l.o.g., that $c_1, c_2, \dots, c_k$ appear in this clockwise order around v in $\mathcal{E}_\varphi$. The underlying graph of $\mathcal{V}_v$ is composed of k split gadgets $S_1, S_2, \dots, S_k$. For each split gadget S_i, with $i = 1, \dots, k$, 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 $i = 1, \dots, k$, 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 $i = 1, \dots, k$. Moreover, for $i = 1, \dots, k$, 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 4** ($\star$). *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 $i = 1, \dots, k$, 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, $e^2_1, \dots,$ a^k_1, e^k_1 or e^1_1, a^1_1, e^2_1, $a^2_1, \dots,$ e^k_1, a^k_1.*

In the proof of Theorem 6, the two circular orders for the edges $\bigcup_{i=1}^k \{a^i_1, e^i_1\}$ of $\mathcal{V}_v$ considered in Lemma 4 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 [19], defined as follows; see Fig. 3. 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 Fig. 3) and a path formed by the edges (e_v, f_v, g_v) (blue paths in Fig. 3). The crossing graph $\mathcal{C}(\mathcal{Q}_c)$ of $\mathcal{Q}_c$ consists of one non-trivial connected component 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 5** ($\star$). *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.*

Based on Lemma 5, 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. 3. We can finally prove the main result of the section.

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

Proof Sketch. The membership in NP is obvious. We give a reduction from the NP-complete problem 3-Connected Planar 3-SAT [17]. 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 $\{a^i_1, e_{v,c_i}\}$ and $\{e^i_1, e_{v,c_i}\}$, where a^i_1 and e^i_1 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^i_1 of $\mathcal{V}_v$ with the edge a_y (resp. e_y) of $\mathcal{Q}_v$ and we identify the edge e^i_1 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^i_1 and e^i_1 to cross. Clearly, A_φ can be constructed in polynomial time. The equivalence between A_φ and φ immediately follows from Lemmata 4 and 5, and from the fact that, by Lemma 2, in any simple realization of A_φ, all the variable cycles and all the clause cycles maintain the same circular orientation. Finally, note that the size of the largest connected component of $\mathcal{C}(A_\varphi)$ is six. ◀

We remark that the NP-hardness of Theorem 6 holds for instances whose crossing graph is planar, and has maximum degree 3 and treewidth 3. Moreover, it implies that SATR is NP-complete when $\lambda(A) \geq k$, for any $k \geq 6$. 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 7** (⋆). *Unless ETH fails,* SATR *has no* $2^{o(\sqrt{n})}$*-time algorithm, where* n *is the number of vertices of the input AT-graph.*

4 A Linear-Time Algorithm for AT-Graphs 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 13. 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.

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 $E(X)$

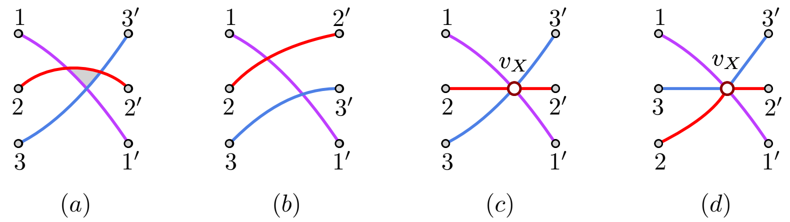


Figure 4 (a) A K_3-crossing. (b) A P_3-crossing. A circular order of the neighbors around a crossing vertex v_X satisfying (c) a K_3-constraint but not a P_3-constraint, (d) a P_3- but not a K_3-constraint.

and add a *crossing vertex* v_X adjacent to all vertices in $V(X)$; see Fig. 4. 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 8** ($\star$). *If A admits a simple realization, then H is planar.*

Observation 8 gives an immediate necessary condition for the realizability of A, which is, however, not sufficient. 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. 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; see Fig. 4 for an example. 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; see Fig. 4. The auxiliary graph H with alternation constraints is *feasible* if it admits a planar embedding that satisfies the alternation constraints of all vertices. Thus, we have the following.

▶ **Lemma 9.** *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.*

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 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

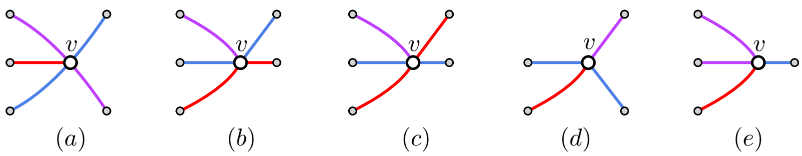


Figure 5 Circular orders of edges incident to a vertex v satisfying (a) a K_3^{-r}- and a P_3^{-r}-constraint, (b) a P_3^{-p}- but not a K_3^{-p}-constraint, (c) a K_3^{-p}- but not a P_3^{-p}-constraint,(d) a $P_3^{-(p,r)}$-but not a $K_3^{-(p,r)}$-constraint , (e) a $P_3^{-(b,r)}$- and a $K_3^{-(b,r)}$-constraint.

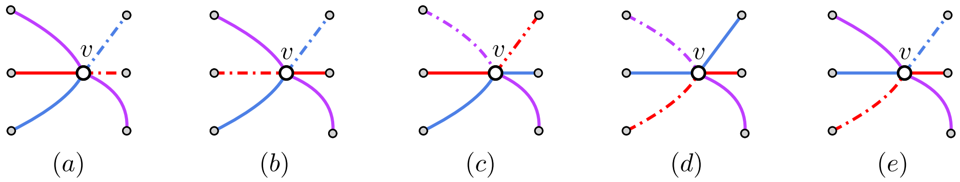


Figure 6 Circular orders of edges around a vertex v allowing to insert two edges of distinct colors (dashed) so that every color occurs twice and a $(a–b)$ K_3-constraint, $(c–e)$ P_3-constraint is satisfied.

(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. 5. 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. 5(a). 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. 5(b).

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. 6. 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. 5(d), (e) for examples. 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 either $c = c' = r$ or $c = r, c' = b$ in this case. For P_3-constraints, only red and blue are symmetric, i.e., we may assume without loss of generality that $(c, c') \in \{(r, r), (p, p), (r, p), (r, b)\}$. 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. 5(*e*). For a $P_3^{-(r,p)}$-constraint to be fulfilled, the two blue edges must not occur consecutively (see Fig. 5(*d*)); 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. 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. By Lemma 9, 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 H is 2-connected.

▶ **Lemma 10** ($\star$). *There is a linear-time algorithm that either recognizes that an instance H of* ACP *is a no-instance or computes a collection $H_1, \ldots, H_k$ of instances of* 2-connected ACP*, such that H is a yes-instance if and only if H_i is a yes-instance for every $1 \leq i \leq k$.*

Proof Sketch. Our reduction strategy considers one cut vertex at a time and splits the graph at that vertex into a collection of smaller 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 each H_i 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 is feasible if and only if each cut component H_i, with $i = 1, \ldots, l$, is and hence all copies of v remain unconstrained. Otherwise, v has an alternation constraint C. This implies $|E(v)| \leq 6$ 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 $C \in \{P_3, P_3^{-p}\}$, 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. If $|E_1(v)| \leq 3$, H is either a no-instance, or we can always arrange the cut components around v such that C is satisfied. In all positive cases, it suffices to leave each copy of v unconstrained. It remains to consider the remaining split-vectors with $|E_1(v)| \geq 4$. Here we only describe the case $(5, 1)$; the remaining cases can be found in the full version [5].

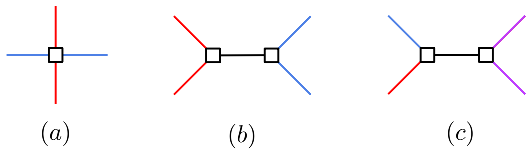


Figure 7 The PQ-trees representing alternation constraints of degree-4 vertices. (a) K_2-constraint, (b) $P_3^{-p,p}$-constraint, and (c) $K_3^{-(r,b)}$-constraint.

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. Therefore, 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.

Note that 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 resulting alternation-constrained blocks of H. ◀

Algorithm for the Embedding Problem. In the following we assume familiarity with the PQ-tree [4, 3] and SPQR-tree data structures [7]. 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 that 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. 7, see the full version [5] for details. Hence we may assume from now on that no vertex with an alternation constraint in H has degree 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. In the full version [5] we show that, with the exception of K_3^{-r}, an alternation constraint at a vertex incident to a consecutive edge pair can be replaced by a PQ-tree. The overall strategy of the remaining section consists of three steps. In Step 1 we identify consecutive edge pairs in H with the help of the SPQR-tree of H and replace the corresponding alternation constraints by PQ-trees. By doing this exhaustively and using a special operation described in [5] to remove the K_3^{-r}-constraints, 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.

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.

▶ **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*
(i) *there is a skeleton in T with a virtual edge containing exactly two edges from $E(v)$ or*
(ii) *there is a skeleton in T with a virtual edge containing all but two edges from $E(v)$ or*
(iii) *v appears in the skeleton of an R-node in T.*

Since we immediately replace alternation constraints by PQ-trees whenever we find a consecutive edge pair, we assume from now on that no vertex with alternation constraint different from K_3^{-r} satisfies one of the conditions of Lemma 11. Let μ be a node of T and let v be a vertex of its skeleton incident to the virtual edges $e_1, \dots, e_k$. Then the *distribution vector* $(d_1, \dots, d_k)$ 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.

Consider a vertex v with alternation constraint different from K_3^{-r} in H. Assume that v appears in an S-node ν in T. Since the vertices in the skeleton of an S-node have degree 2, v also appears in at least one other node μ adjacent to ν in T. Note that μ is a P-node since there are no two adjacent S-nodes in an SPQR-tree. Hence, we may assume in the following that every vertex with alternation constraint appears in a P-node μ. 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. Since by assumption no virtual edge contains exactly two edges from $E(v)$ or all but two edges from $E(v)$, the only possible distributions without a consecutive edge pair are $(1, 1, 1, 1, 1, 1)$, $(1, 1, 1, 1, 1)$ and $(3, 1, 1, 1)$.

In the first two cases it can be shown that we can get rid of the alternation constraint C of v as it is either always possible to reorder the children of μ according to C in a realization of H without C or H without C (and thus H) is not realizable. Similar techniques allow us to show that we can get rid of *(i)* P_3-constraints, *(ii)* K_3^{-r}-constraints and of *(iii)* K_3-constraints of poles of P-nodes such that the other pole is either unconstrained or has a PQ-tree. The proofs are deferred to the full version [5]. 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.

Now move to Step 2. 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. 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 $v_0, \dots, v_{k-1}$ in this order; see Fig. 8. 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.

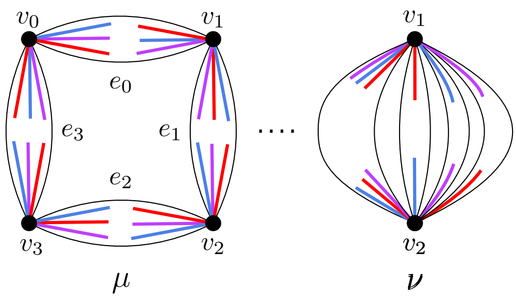


Figure 8 An S-node μ and an adjacent P-node ν.

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 12.** *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.*

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 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 13** (⋆)**.** *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.*

5 Conclusions and Open Problems

We proved that deciding whether an AT-graph A is simply realizable is NP-complete, already when the size $\lambda(A)$ of the largest connected components of the crossing graph $\mathcal{C}(A)$ satisfies $\lambda(A) \leq 6$. On the other hand, we described an optimal linear-time algorithm that solves the problem when $\lambda(A) \leq 3$. 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 $4 \leq \lambda(A) \leq 5$.

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 $\mathcal{C}(A)$. Finally, one can try to extend our approach to the "weak" setting (i.e., the Weak AT-graph Realizability problem), still requiring a simple realization.

References

1 Thomas Bläsius, Simon D. Fink, and Ignaz Rutter. Synchronized planarity with applications to constrained planarity problems. *ACM Trans. Algorithms*, 19(4):34:1–34:23, 2023. doi:10.1145/3607474.

2 Thomas Bläsius and Ignaz Rutter. Simultaneous PQ-ordering with applications to constrained embedding problems. *ACM Trans. Algorithms*, 12(2):16:1–16:46, 2016. doi:10.1145/2738054.

3 Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci.*, 13(3):335–379, 1976. doi:10.1016/S0022-0000(76)80045-1.

4 Kellogg Speed Booth. *PQ-tree algorithms.* University of California, Berkeley, 1975.

5 Giordano Da Lozzo, Walter Didimo, Fabrizio Montecchiani, Miriam Münch, Maurizio Patrignani, and Ignaz Rutter. Simple realizability of abstract topological graphs. *CoRR*, abs/2409.20108, 2024. doi:10.48550/arXiv.2409.20108.

6 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice-Hall, 1999.

7 Giuseppe Di Battista and Roberto Tamassia. On-line maintenance of triconnected components with SPQR-trees. *Algorithmica*, 15(4):302–318, 1996. doi:10.1007/BF01961541.

8 Giuseppe Di Battista and Roberto Tamassia. On-line planarity testing. *SIAM J. Comput.*, 25(5):956–997, 1996. doi:10.1137/S0097539794280736.

9 Walter Didimo, Giuseppe Liotta, and Fabrizio Montecchiani. A survey on graph drawing beyond planarity. *ACM Comput. Surv.*, 52(1):4:1–4:37, 2019. doi:10.1145/3301281.

10 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

11 Elisabeth Gassner, Michael Jünger, Merijam Percan, Marcus Schaefer, and Michael Schulz. Simultaneous graph embeddings with fixed edges. In Fedor V. Fomin, editor, *32nd International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2006*, volume 4271 of *Lecture Notes in Computer Science*, pages 325–335. Springer, 2006. doi:10.1007/11917496_29.

12 Jacob E. Goodman and Joseph O'Rourke, editors. *Handbook of Discrete and Computational Geometry, Second Edition.* Chapman and Hall/CRC, 2004. doi:10.1201/9781420035315.

13 Carsten Gutwenger, Karsten Klein, and Petra Mutzel. Planarity testing and optimal edge insertion with embedding constraints. *J. Graph Algorithms Appl.*, 12(1):73–95, 2008. doi:10.7155/JGAA.00160.

14 Seok-Hee Hong. Beyond planar graphs: Introduction. In Seok-Hee Hong and Takeshi Tokuyama, editors, *Beyond Planar Graphs, Communications of NII Shonan Meetings*, pages 1–9. Springer, 2020. doi:10.1007/978-981-15-6533-5_1.

15 John E. Hopcroft and Robert Endre Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974. doi:10.1145/321850.321852.

16 Jan Kratochvíl. String graphs. II. Recognizing string graphs is NP-hard. *J. Comb. Theory, Ser. B*, 52(1):67–78, 1991. doi:10.1016/0095-8956(91)90091-W.

17 Jan Kratochvíl. A special planar satisfiability problem and a consequence of its NP-completeness. *Discrete Applied Mathematics*, 52(3):233–252, 1994. `doi:10.1016/0166-218X(94)90143-0`.

18 Jan Kratochvíl, Anna Lubiw, and Jaroslav Nesetril. Noncrossing subgraphs in topological layouts. *SIAM J. Discret. Math.*, 4(2):223–244, 1991. `doi:10.1137/0404022`.

19 Jan Kratochvíl and Jiří Matoušek. NP-hardness results for intersection graphs. *Commentationes Mathematicae Universitatis Carolinae*, 30(4):761–773, 1989. URL: `http://eudml.org/doc/17790`.

20 Jan Kratochvíl and Jirí Matousek. Intersection graphs of segments. *J. Comb. Theory, Ser. B*, 62(2):289–315, 1994. `doi:10.1006/JCTB.1994.1071`.

21 Jan Kyncl. Simple realizability of complete abstract topological graphs in P. *Discret. Comput. Geom.*, 45(3):383–399, 2011. `doi:10.1007/S00454-010-9320-X`.

22 Jan Kyncl. Simple realizability of complete abstract topological graphs simplified. *Discret. Comput. Geom.*, 64(1):1–27, 2020. `doi:10.1007/S00454-020-00204-0`.

23 Marcus Schaefer. Toward a theory of planarity: Hanani-tutte and planarity variants. *J. Graph Algorithms Appl.*, 17(4):367–440, 2013. `doi:10.7155/JGAA.00298`.

24 Marcus Schaefer, Eric Sedgwick, and Daniel Stefankovic. Recognizing string graphs in NP. *J. Comput. Syst. Sci.*, 67(2):365–380, 2003. `doi:10.1016/S0022-0000(03)00045-X`.

25 Marcus Schaefer and Daniel Stefankovic. Decidability of string graphs. *J. Comput. Syst. Sci.*, 68(2):319–334, 2004. `doi:10.1016/J.JCSS.2003.07.002`.

26 Ileana Streinu, Károly Bezdek, János Pach, Tamal K. Dey, Jianer Chen, Dina Kravets, Nancy M. Amato, and W. Randolph Franklin. Discrete and computational geometry. In Kenneth H. Rosen, John G. Michaels, Jonathan L. Gross, Jerrold W. Grossman, and Douglas R. Shier, editors, *Handbook of Discrete and Combinatorial Mathematics*. CRC Press, 1999. `doi:10.1201/9781439832905.CH13`.