Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 30, no. 2, pp. 175–197 (2026)
DOI: 10.7155/jgaa.v30i2.3114

OOPS: Optimized One-Planarity Solver via SAT

Sergey Pupyrev[1]

[1]Menlo Park, CA, USA

Submitted: October 2025	Accepted: July 2026	Published: July 2026
Article type: Regular paper	Communicated by: V. Dujmović, F. Montecchiani	

Abstract. We present OOPS (Optimized One-Planarity Solver), a practical heuristic for recognizing 1-planar graphs and several important subclasses. A graph is 1-planar if it can be drawn in the plane such that each edge is crossed at most once—a natural generalization of planar graphs that has received increasing attention in graph drawing and beyond-planar graph theory. Although testing planarity can be done in linear time, recognizing 1-planar graphs is NP-complete, making effective practical algorithms especially valuable.

The core idea of our approach is to reduce the recognition of 1-planarity to a propositional satisfiability (SAT) instance, enabling the use of modern SAT solvers to efficiently explore the search space. Despite the inherent complexity of the problem, our method is substantially faster in practice than naïve or brute-force algorithms. In addition to demonstrating the empirical performance of our solver on synthetic and real-world instances, we show how OOPS can be used as a discovery tool in theoretical graph theory. Specifically, we employ OOPS to investigate two research problems concerning 1-planarity of specific graph families. Our implementation of the algorithm is publicly available to support further exploration in the field.

1 Introduction

Planarity is a central concept in graph drawing; it describes graphs that can be embedded in the plane without edge crossings. Thanks to classical results, planarity testing can be carried out in linear time, and planar embeddings can be constructed for very large instances. However, most graphs arising in real-world applications, such as bioinformatics and software engineering, are non-planar. This motivates the study of *beyond-planar* graph classes, which extend planarity by allowing limited types of edge crossings.

Special issue on Selected papers from the Thirty-third International Symposium on Graph Drawing and Network Visualization, GD 2025

A preliminary version of this work appeared in the proceedings of the 33rd International Symposium on Graph Drawing and Network Visualization (GD'25) [35].

E-mail address: spupyrev@gmail.com (Sergey Pupyrev)

One of the most natural such classes is that of *1-planar* graphs, introduced by Ringel in 1965 in the context of vertex-face colorings of plane graphs [37]. A graph is 1-planar if it can be drawn in the plane such that each edge is crossed at most once. This definition generalizes planar graphs but retains many of their structural properties, such as bounded edge density, bounded chromatic number, small separators, and low treewidth. The class of 1-planar graphs is receiving increasing attention in recent years: While the annotated bibliography on 1-planarity by Kobourov, Liotta, and Montecchiani from 2017 [28] lists 143 references on the topic, a recent Google Scholar search of "1-planar graph" reveals several thousand related publications.

Despite their structural similarities to planar graphs, 1-planar graphs are substantially more challenging from a computational standpoint. In contrast to the linear-time algorithms for planarity testing, recognizing whether a given graph is 1-planar is NP-complete, even for restricted graph families [14, 24, 29]. As a result, research on 1-planarity recognition has focused on developing algorithms for restricted subclasses [5, 25, 41], or on parameterized and approximation algorithms [8, 20]. While these results have deepened our theoretical understanding of 1-planarity, there remains a significant gap between theory and practice: existing exact algorithms are typically not scalable for use on general graphs arising in real-world scenarios. The design and evaluation of practical algorithms for recognizing (subclasses of) 1-planar graphs remains an underexplored direction [10,28].

1.1 Summary of Contributions

In this paper, we make progress towards reducing the gap by designing and implementing a practical heuristic for testing 1-planarity. Besides a naïve exhaustive search to enumerate all possible crossings in a graph, we are aware of only one attempt to implement a general algorithm for recognizing 1-planar graphs [10].[1] While this backtracking algorithm is carefully engineered, it is not efficient enough to answer basic research questions like "What is the smallest bipartite non-1-planar graph?" or "Is the 28-vertex Coxeter graph 1-planar?". In contrast, our new algorithm, which we call OOPS (Optimized One-Planarity Solver), provides a substantial speedup and thus is capable of answering such questions in seconds.

Our heuristic is based on converting the 1-planarity problem into a propositional satisfiability (SAT) instance, which enables the use of modern SAT solvers. To this end, we design two novel schemes for encoding the drawing constraints of 1-planarity into a discrete SAT instance. The first one is based on the Hanani-Tutte characterization of planar graphs, which has been used earlier for designing practical algorithms for upward planarity [15]. This type of encoding is rather flexible and can be easily extended to other use cases, such as upward 1-planarity of directed graphs. Our second encoding is tailored to 1-planar graphs and exploits a connection to book embeddings. It results in a specifically compact encoding and much faster processing times, and thus can be considered an interesting contribution on its own.

We summarize the main contributions of the paper as follows.

- In Section 2 we describe *our algorithm for the recognition of 1-planar graphs.* It starts with a preprocessing step and an analysis of global properties of the input graph, such as edge density, that can eliminate some obviously non-1-planar instances. Then we present the two SAT encoding schemes for 1-planarity, followed by a set of additional rules for breaking symmetries and reducing the search space.

[1] Simultaneously and independently, Fink, Münch, Pfretzschner, and Rutter [22] are developing a new heuristic for 1-planarity, appearing after the preparation of this manuscript.

- An *implementation of the algorithm*, `OOPS`, is open sourced in [36]. The implementation is self-contained and requires only a `C++` compiler with `C++17` support.
- Section 3 presents an *extensive evaluation* of the new approach by comparing its efficiency on a suite of benchmark graphs. In particular, we provide the status of 1-planarity for named WIKIPEDIA graphs (Table 2). Furthermore, we use `OOPS` to investigate two theoretical questions concerning 1-planarity regarding the smallest non-1-planar instances for several graph families (Problem 1) and the existence of a subclass of 1-planar graphs with certain properties (Problem 2).

1.2 Other Related Work

Although the family of 1-planar graphs shares some similarities with planar graphs, there is a fundamental difference. Planarity can be characterized by Wagner's theorem (forbidden minors K_5 and $K_{3,3}$) or by Kuratowski's theorem (forbidden subdivisions of K_5 and $K_{3,3}$). In contrast, every graph admits a 1-planar subdivision, making it impossible to characterize 1-planar graphs via a Kuratowski-type theorem. Testing 1-planarity is NP-complete. The first proof of this result is given by Grigoriev and Bodlaender [24], via a reduction from the 3-partition problem. An independent proof by Korzhik and Mohar [29] uses a reduction from the 3-coloring problem for planar graphs. The recognition problem remains NP-complete even for graphs with bounded bandwidth, pathwidth, or treewidth [8]; for graphs obtained from planar graphs by the addition of a single edge [14], and for graphs with a fixed rotation system [6]. However, the problem becomes fixed-parameter tractable when parameterized by the vertex-cover number, cyclomatic number, or tree-depth [8]. Polynomial-time algorithms are known only for restricted subfamilies of 1-planar graphs, such as IC-planar and outer-1-planar graphs [1, 5, 13, 25, 41]. The extensive literature on the topic reflects a strong interest in determining the boundary between tractable and intractable cases. Nevertheless, general-purpose algorithms that are both practical and widely applicable remain scarce.

To prove that a given graph is not 1-planar, one can attempt several approaches. One method is to show that the graph has too many edges, as a 1-planar graph with n vertices has at most $4n - 8$ edges [11]; better bounds exist for subclasses such as bipartite 1-planar graphs [26, 27]. Another approach is to leverage the chromatic number: 1-planar graphs are 6-colorable [12], so a graph requiring more colors cannot be 1-planar. Alternatively, one may identify a small non-1-planar subgraph (e.g., a complete multipartite graph [16]), or show that any drawing of the graph necessarily contains too many crossings [17]. However, once a graph passes these basic filters (consider for example a random cubic graph on 30 vertices), determining whether a 1-planar drawing exists—or proving its nonexistence—remains a challenging and elusive problem. A brute-force approach is practical only for small instances, typically those with up to $16-20$ vertices, depending on the level of engineering effort. For this reason, Eppstein [48], Kobourov, Liotta, and Montecchiani [28] and Binucci, Didimo, and Montecchiani [10] discuss the need for a more powerful technique to attack larger graphs.

2 Model

2.1 Preliminaries

Let $G = (V, E)$ be a simple undirected graph with n vertices and m edges. A *drawing* Γ of G maps vertices V to distinct points of the plane and edges E to Jordan curves connecting corresponding endpoints; the curves may not pass through vertices except their endpoints. Two edges *cross* if their Jordan curves intersect in a point different from the endpoints. For the ease of notation we often identify a vertex $v \in V$ and its drawing $\Gamma(v)$ as well as an edge $e \in E$ and its drawing $\Gamma(e)$. A drawing is *planar* if every edge is crossing-free, and a graph is planar if it admits a planar drawing. A drawing of a graph partitions the plane into connected regions, called *faces*; the unbounded region is called the *outer face*. The set of all faces describes the *embedding* of a planar graph G.

A graph is *1-planar* if it admits a 1-planar drawing, that is, a drawing in which every edge is crossed at most once. The *planarization* of a 1-planar drawing of G is a (planar) graph that replaces each pair of crossing edges, $(u, v) \in E$ and $(x, y) \in E$, by four edges $(u, d), (v, d), (x, d), (y, d)$, where d is a new *dummy* vertex. The vertices of G in the planarization are referred to as *original*. There are several important subclasses of 1-planar graphs; for example, *optimal* 1-planar graphs containing the maximum possible number of edges, $4n - 8$ [11]. A graph is *IC-planar* (independent crossing planar) if it has a 1-planar embedding so that each vertex is incident to at most one crossing edge [1]. In *NIC-planar* graphs (near-independent crossing planar), two pairs of crossing edges share at most one vertex [41]. *Outer 1-planar* graphs are another subclass of 1-planar graphs; they admit a 1-planar drawing such that all vertices are in the unbounded region [5, 25].

2.2 Preprocessing

We may simplify the testing of 1-planarity of an input graph G by preprocessing the graph and eliminating some instances early. First, we split G into (edge-disjoint) biconnected components and test each component independently, as a 1-planar drawing of G can easily be composed from 1-planar drawings of its biconnected components. Second, we check the edge density of G and reject instances violating the maximum possible density in a 1-planar graph. While there are several density-related results in the area (e.g., for IC and NIC graphs [7, 42]), we apply only the $4n - 8$ bound for general graphs [11] and the $3n - 8$ bound for bipartite ones [27]. Third, we test if G is *planar*, which can be done efficiently. Finally, we verify if G admits a 1-planar drawing with one crossing, that is, a drawing in which only one pair of edges cross. This can be done by enumerating all possible pairs of edges, and testing if replacing the pair of crossed edges with a vertex of degree four yields a planar graph.

After the preprocessing step, we may assume that the input graph is biconnected, non-planar, and does not violate the edge density.

2.3 SAT-Based Encoding of 1-Planarity

To test whether a given graph $G = (V, E)$ admits a 1-planar drawing, we formulate a Boolean Satisfiability Problem that has a solution if and only if G is 1-planar. We develop two encoding schemes for the problem; both are based on the concept of a linear graph layout (also referred to as an ordered embedding in [15]). Refer to [34] for a survey of various types of linear layouts. Assume that G admits a 1-planar drawing; planarize it by adding a dummy vertex for every crossing and consider a straight-line planar drawing of the planarization. Now it is easy to perturb the vertices such that they have different x-coordinates, yielding a total (linear) order of the vertices; see Fig. 1. Observe that the drawing can be modified so that the vertices lie on the same horizontal line (called the *spine*) while keeping planarity and the vertex x-coordinates. The edges in the drawing are curves that are *monotone* in the x direction; they cannot cross each other but can pass through the spine multiple times, as the red edge (c, e) in Fig. 1c.

Our encoding schemes operate with two auxiliary graphs built as follows. Subdivide every

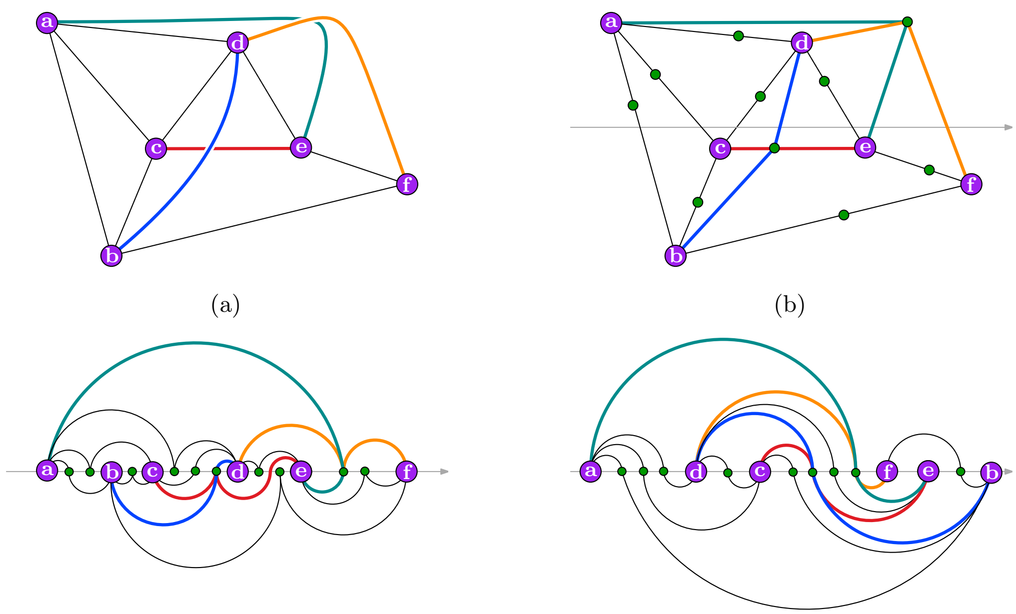


Figure 1: (a) A 1-planar drawing of a graph G with crossings between edges (a, e) and (d, f) and between (b, d) and (c, e). (b) An auxiliary graph G'_p with green *division* vertices; merged division vertices represent crossings. (c) A linear layout (a.k.a. ordered embedding) of the graph for the Hanani-Tutte-based SAT encoding of 1-planarity. (d) A stack layout of the graph for the Stack-based encoding.

edge of G with a new *division* vertex to get its subdivision $G' = (V', E')$ with $|E'| = 2m$ and $|V'| = n + m$. The division vertices are $D = V' \setminus V$. From G' we build another graph, denoted G'_p, by allowing to *merge* pairs of division vertices. Let $(u, v) \in E$ and $(x, y) \in E$ be two edges of G that are subdivided by $d_{uv} \in D$ and $d_{xy} \in D$ in G'. By merging d_{uv} and d_{xy}, we get a new (dummy) vertex, d, and four edges in G'_p, namely (u, d), (v, d), (x, d), and (y, d), replacing the four edges incident to d_{uv} and d_{xy}; see Fig. 1b. Thus merging preserves the number of edges but decreases the number of vertices. The construction of G'_p is similar to planarization of (a 1-planar drawing of) G, except that G'_p contains extra non-merged division vertices corresponding to crossing-free edges. The importance of G'_p is highlighted by the following observation.

Lemma 1. *Let G be a graph and G' be its subdivision. Then G is 1-planar if and only if merging some pairs of division vertices in G' results in a planar graph G'_p.*

Proof: In one direction, if G is 1-planar, consider its planarization. Subdividing every edge non-incident to a dummy vertex yields a desired planar graph G'_p. In another direction, start with a planar drawing of G'_p; *smooth* all (non-merged) degree-2 division vertices and replace all degree-4 dummy vertices with two edges. The result is a 1-planar drawing of G. □

To model the merging process in G' by a SAT formula, consider a partial order σ on V'. In this order all vertices are pairwise comparable, except for pairs of merged division vertices that "share" a position in σ; these exceptional pairs correspond to edge crossings in G. Formally, we introduce variables encoding the relative order of vertices V' in σ:

$\mathbb{V}1$: two variables $\sigma(v,u)$ and $\sigma(u,v)$ for every pair of distinct vertices $v,u \in V'$ indicating whether v precedes u in the order.

Intuitively, $\sigma(v,u) = \texttt{true}$ means that v precedes u in the order, and symmetrically, $\sigma(v,u) = \texttt{false}$ means that v follows u. We stress that this is a partial order, meaning that for some pairs of division vertices, called *merged*, it holds that $\sigma(v,u) = \sigma(u,v) = \texttt{false}$.

To indicate that a pair of division vertices is merged (or equivalently, that a pair of edges in G cross each other), we introduce variables for pairs of division vertices. Note that not all division vertices can be merged. For example, one may assume that adjacent edges in G do not cross in its 1-planar drawing; hence, we forbid to merge the corresponding division vertices by introducing a set of *candidate* pairs: $C \subseteq D \times D$. In the simplest scenario, C contains pairs of division vertices corresponding to independent edges in E; Section 2.4 shows how to further shrink the set of candidates. The introduced variables are as follows:

$\mathbb{V}2$: a variable $\chi(v,u)$ for every pair of distinct division vertices v,u with $(v,u) \in C$ indicating whether v is merged with u.

To ensure the correctness of the order, σ, we enforce the following constraints on associativity of non-candidate vertex pairs and on order transitivity:

$\mathbb{C}1$: $\sigma(v,u) \leftrightarrow \neg\sigma(u,v)$ $\quad \forall (u,v) \in V' \times V' \setminus C$.

$\mathbb{C}2$: $\neg\sigma(v,u) \vee \neg\sigma(u,v)$ $\quad \forall (u,v) \in C$.

$\mathbb{C}3$: $\big(\sigma(v,u) \wedge \sigma(u,w)\big) \rightarrow \sigma(v,w)$ $\quad \forall$ distinct $u,v,w \in V'$.

In order to link relative variables $\mathbb{V}1$ with merge variables $\mathbb{V}2$, we use:

$\mathbb{C}4$: $\sigma(v,u) \vee \sigma(u,v) \vee \chi(v,u)$ $\quad \forall (u,v) \in C$.

Finally, a division vertex can be merged with only one other division vertex:

$\mathbb{C}5$: $\neg\big(\chi(v,u) \wedge \chi(v,w)\big)$ $\quad \forall$ distinct $u,v,w \in D$.

Next we discuss how the basic scheme can be customized in two different ways to encode 1-planarity of a graph.

2.3.1 Hanani-Tutte Encoding

The encoding presented in the previous section can model a linear layout of a graph and crossings between pairs of edges. However, one still needs to guarantee that the edges can be routed in a planar way. This is achieved with the help of a Hanani-Tutte-type characterization for monotone drawings.

Lemma 2 ([23, 33])**.** *Let G be a graph. If G has an x-monotone drawing such that every pair of independent edges crosses an even number of times, then there is an x-monotone planar drawing of G with the same vertex locations.*

Based on the characterization, Chimani and Zeranski [15] designed a SAT formulation for *upward planarity*, where edges of a directed acyclic graph have to be drawn in a monotone fashion. We observe that essentially the same encoding can be used for graph G'_p. To this end, Chimani and Zeranski mimic an earlier algorithm by Fulek, Pelsmajer, Schaefer, and Štefankovič [23], introducing boolean (e, v)-move variables that model how an edge $e \in E'$ is routed around a vertex $v \in V'$. The `true` (resp., `false`) value of the variable indicates that edge e is routed above (below) v.

$\mathbb{V}3$: a variable $\psi(e, v)$ for every edge $e \in E'$ and every vertex $v \in V'$ to indicate whether we perform an (e, v)-move.

For a fixed vertex order, as in the setup of Lemma 2, the move variables can be combined into a set of 2-clauses that guarantee an even number of crossings for independent edges, and hence guarantee a planar drawing. When the vertex order is not fixed, we need to enumerate all possible combinations of edge endpoints. Here we consider a pair of edges $e = (s_e, t_e) \in E'$, $f = (s_f, t_f) \in E'$ and assume that s_e precedes t_e in σ and that s_f precedes t_f in σ.

$\mathbb{C}6$: $\big(\sigma(s_e, s_f) \wedge \sigma(s_f, t_e) \wedge \sigma(t_e, t_f)\big) \rightarrow \big(\psi(e, s_f) \leftrightarrow \neg\psi(f, t_e)\big) \qquad \forall e, f \in E'$.

Intuitively, constraint $\mathbb{C}6$ forbids a crossing between e and f (by rerouting one of the two edges near s_f or t_e, respectively) in the case when the order of their endpoints is $x(s_e) < x(s_f) < x(t_e) < x(t_f)$. Similarly, $\mathbb{C}7$ is introduced for the order $x(s_e) < x(s_f) < x(t_f) < x(t_e)$:

$\mathbb{C}7$: $\big(\sigma(s_e, s_f) \wedge \sigma(s_f, t_f) \wedge \sigma(t_f, t_e)\big) \rightarrow \big(\psi(e, s_f) \leftrightarrow \psi(e, t_f)\big) \qquad \forall e, f \in E'$.

Finally, we need to ensure that merged division vertices correspond to the same move variables:

$\mathbb{C}8$: $\chi(u, v) \rightarrow \big(\psi(e, u) \leftrightarrow \psi(e, v)\big) \qquad \forall (u, v) \in C, e \in E'$.

The proof of the following theorem is inspired by results (for planar graphs) in [15, 23] with the key difference that G' is non-planar and contains pairs of division vertices that can be merged.

Theorem 1. *Let $G = (V, E)$ be a graph and $G' = (V', E')$ be its subdivision. Then G is 1-planar if and only if the SAT formula built for G' and comprised of variables $\mathbb{V}1 \cup \mathbb{V}2 \cup \mathbb{V}3$ and constraints $\mathbb{C}1 \cup \mathbb{C}2 \cup \mathbb{C}3 \cup \mathbb{C}4 \cup \mathbb{C}5 \cup \mathbb{C}6 \cup \mathbb{C}7 \cup \mathbb{C}8$ is satisfiable.*

Proof: Assume that graph $G = (V, E)$ is 1-planar and let $G' = (V', E')$ be its subdivision. Let us show that there is an assignment of variables $\mathbb{V}1 \cup \mathbb{V}2 \cup \mathbb{V}3$ such that constraints $\mathbb{C}1 \cup \mathbb{C}2 \cup \mathbb{C}3 \cup \mathbb{C}4 \cup \mathbb{C}5 \cup \mathbb{C}6 \cup \mathbb{C}7 \cup \mathbb{C}8$ are satisfied.

Consider a 1-planar drawing of G; planarize it by inserting a dummy vertex for every crossing to get a planar graph G_p. Now we subdivide every edge between original vertices and call the result G'_p. This graph, G'_p, is planar and hence admits a straight-line drawing. Perturb the vertices in the drawing so that they have distinct x-coordinates, and project the vertices to a horizontal line. We get a (planar) drawing of G'_p, where every edge is a curve monotone in the x direction; see Fig. 1c. Now we build a partial order, σ, of V' by assigning $\sigma(u, v) =$ `true` whenever $x(u) < x(v)$. Each dummy vertex in G'_p corresponds to a pair of merged division vertices $u, v \in V'$ with the same coordinates; set $x(u) = x(v)$ and $\sigma(u, v) = \sigma(v, u) =$ `false` for such pairs. This construction guarantees constraints $\mathbb{C}1$, $\mathbb{C}2$, $\mathbb{C}3$, $\mathbb{C}4$, and $\mathbb{C}5$. The move variables, $\mathbb{V}3$, are assigned depending on whether an edge $e \in E'$ is routed above or below vertex $v \in V'$. The absence of crossings implies $\mathbb{C}6$ and $\mathbb{C}7$; constraint $\mathbb{C}8$ is satisfied by construction.

For the opposite direction, assume that variables $\mathbb{V}1$, $\mathbb{V}2$, $\mathbb{V}3$ are assigned so that $\mathbb{C}1$, $\mathbb{C}2$, $\mathbb{C}3$, $\mathbb{C}4$, $\mathbb{C}5$, $\mathbb{C}6$, $\mathbb{C}7$, $\mathbb{C}8$ are satisfied for G'. Constraints $\mathbb{C}1$, $\mathbb{C}2$, and $\mathbb{C}3$ guarantee a partial order of

V' where all pairs of vertices are comparable except for merged pairs. We position the vertices of G' on a spine following the order. To draw edges, we use $\mathbb{V}3$ together with $\mathbb{C}6$, $\mathbb{C}7$, $\mathbb{C}8$ and apply Theorem 4.1 of [23]. Note that edges might cross the spine multiple times but do not cross each other. To get a 1-planar drawing of G, simply smooth (degree-2) division vertices; merged division vertices correspond to edge crossings. □

2.3.2 Stack-Based Encoding

While the Hanani-Tutte-based encoding already provides a practical approach for 1-planarity, it does not exploit all properties of 1-planar graphs. We observe that the auxiliary graph G' is *bipartite*, with one part being the original vertices, V, and another part being the division vertices, D. Furthermore, the graph remains bipartite after merging division vertices; that is, G'_p is bipartite too. To exploit the bipartiteness, we use the concept of (2-page) *book embeddings* (a.k.a. *stack layouts*), which are special types of linear graph layouts in which edges are represented by (x-monotone) semi-arcs that do not cross each other and the spine. We rely on the following result by Felsner, Fusy, Noy, and Orden [21]:

Lemma 3 ([21]). *Let $G = (V_1 \cup V_2, E)$ be a bipartite planar graph. Then G admits a (crossing-free) stack layout with order $<$ in which edge $(u, v) \in E$ with $u \in V_1$, $v \in V_2$, and $u < v$ (resp., $v < u$) is drawn as a semi-arc above (resp., below) the spine.*

The lemma yields a 2-page book embedding for a maximal planar bipartite graph (a quadrangulation) using its *separating decomposition*. Intuitively, a separating decomposition for a quadrangulation with a fixed planar embedding is an orientation and a (red/blue) coloring of the edges such that every vertex (except two special vertices on the outer face) is incident to a non-empty interval of red edges followed by a non-empty interval of blue edges in the cyclic order around the vertex. Such a decomposition naturally defines an *equatorial line* that separates blue and red edges. The properties of the separating decomposition imply Lemma 3, that is, the existence of a 2-page book embedding in which the equatorial line is a spine and red/blue edges form two trees, that are called *alternating* in [21]. One tree contains the edges above the spine and another one contains the edges below the spine; see Fig. 1d. Refer to Fig. 2 for an illustration of the concepts.

In order to apply the result, assume that G is 1-planar. Then combining Lemma 1 and Lemma 3, it follows that G'_p admits a stack layout with two alternating trees. This enables to encode 1-planarity of G as follows. Consider a pair of independent edges $e = (o_e, d_e) \in E'$ with $o_e \in V, d_e \in D$ and $f = (o_f, d_f) \in E'$ with $o_f \in V, d_f \in D$; assume that o_e precedes o_f in order σ. We forbid a crossing between e and f above or below the spine using the following constraints.

$\mathbb{C}9$: $\neg\sigma(o_e, o_f) \vee \neg\sigma(o_f, d_e) \vee \neg\sigma(d_e, d_f) \qquad \forall e, f \in E'$.

$\mathbb{C}10$: $\neg\sigma(d_e, d_f) \vee \neg\sigma(d_f, o_e) \vee \neg\sigma(o_e, o_f) \qquad \forall e, f \in E'$.

Observe that the constraints resemble $\mathbb{C}6$ and $\mathbb{C}7$ utilized for the Hanani-Tutte encoding. The crucial difference is the absence of variables indicating whether the edges are drawn above or below the spine, which are defined by relative positions of the endpoints in alternating trees.

Theorem 2. *Let $G = (V, E)$ be a graph and $G' = (V', E')$ be its subdivision. Then G is 1-planar if and only if the SAT formula built for G' and comprised of variables $\mathbb{V}1 \cup \mathbb{V}2$ and constraints $\mathbb{C}1 \cup \mathbb{C}2 \cup \mathbb{C}3 \cup \mathbb{C}4 \cup \mathbb{C}5 \cup \mathbb{C}9 \cup \mathbb{C}10$ is satisfiable.*

Proof: Assume that a graph $G = (V, E)$ is 1-planar and let $G' = (V', E')$ be its subdivision. Consider a 1-planar drawing of G and build graph G'_p by planarizing the drawing and subdividing

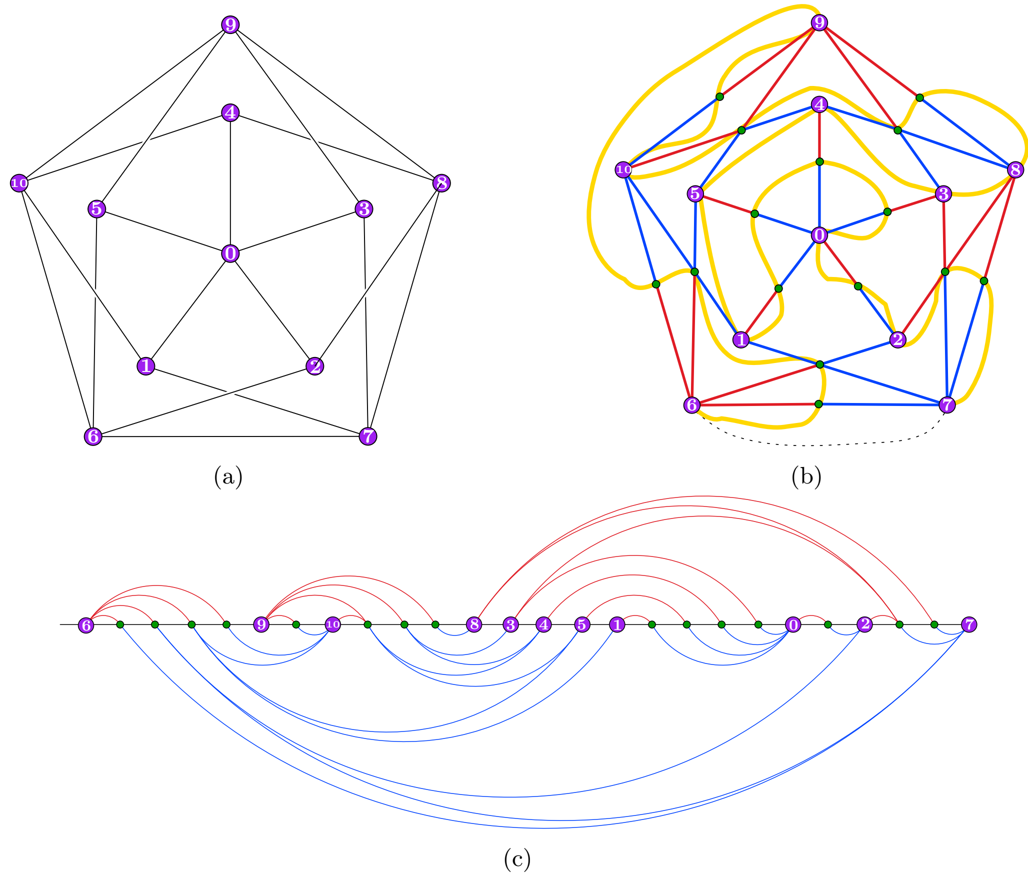


Figure 2: An illustration of the stack-based encoding for the `Grötzsch` graph: (a) A 1-planar drawing; (b) Separating decomposition with equatorial line; (c) The corresponding 2-page book embedding (stack layout) provided by Lemma 3.

non-crossed edges. Since G'_p is bipartite, there exists a stack layout given by Lemma 3. We use the order of vertices along the spine in the stack layout to assign variables $\mathbb{V}1$ and $\mathbb{V}2$, satisfying constraints $\mathbb{C}1$, $\mathbb{C}2$, $\mathbb{C}3$, $\mathbb{C}4$, and $\mathbb{C}5$. Since the edges do not cross each other, $\mathbb{C}9$ and $\mathbb{C}10$ are also satisfied.

For the converse, note that variables $\mathbb{V}1$ and $\mathbb{V}2$ together with $\mathbb{C}1$, $\mathbb{C}2$, $\mathbb{C}3$, $\mathbb{C}4$, $\mathbb{C}5$ define a linear layout of G'_p. Constraints $\mathbb{C}9$ and $\mathbb{C}10$ guarantee that there are no crossings between edges of G'_p. To get a 1-planar drawing of G, smooth division vertices so that every edge of G is represented by two semi-arcs, one above and one below the spine; merged division vertices correspond to edge crossings. □

This completes the description of the Stack-based encoding. Compared with the Hanani-Tutte formulation from Section 2.3.1, this encoding is generally more efficient in practice, as shown in Section 3. The Hanani-Tutte-based encoding, however, is more flexible, since it preserves the vertex order along the spine obtained from a 1-planar drawing, whereas the Stack-based encoding may use a different order arising from the book-embedding construction. This flexibility makes the

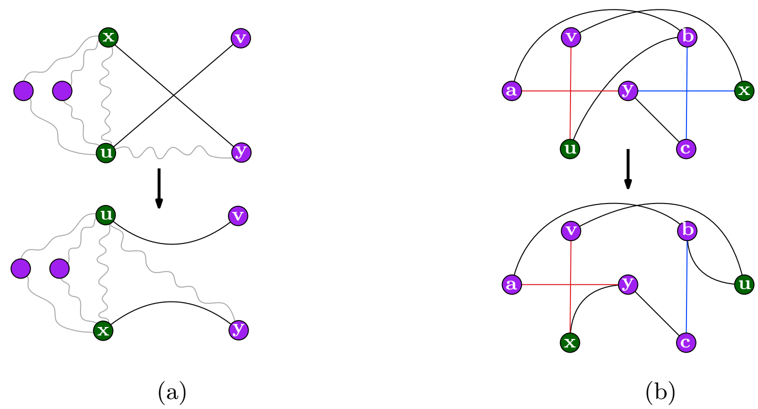


Figure 3: Possible configurations in which swapping (exchanging) x and u eliminates a crossing.

Hanani-Tutte formulation useful for variants in which the vertex order is constrained, such as the extensions discussed in Section 2.5.

2.4 Optimizations

Here we describe additional tricks to reduce the search space and speed up SAT solvers for recognizing 1-planar graphs. The first one is an improvement to relative variables, $\mathbb{V}1$. Observe that constraint $\mathbb{C}1$ enforces associativity for pairs of vertices that cannot be merged, including for example, pairs of original vertices. While modern SAT solvers are able to simplify a formula substituting variable $\sigma(v,u)$ with $\neg\sigma(u,v)$ (or vice versa), we found that performing this substitution directly during formula generation reduces both memory usage and solver runtime.

For the next optimization, we exploit a well-known property of 1-planar drawings. Consider a pair of crossing edges $e=(u,v)\in E$ and $f=(x,y)\in E$; then neither of the four adjacent edges (also called the *kite* edges in [10]) $(u,x),(x,v),(v,y),(y,u)$ can participate in a crossing. This leads to a set of extra constraints that forbid such a crossing: $\neg\chi(e,f)\vee\neg\chi(k,t)$, where $k\in E$ is one of the kite edges corresponding to pair e,f and $t\in E$ is an arbitrary edge of G.

A particularly useful technique for improving the effectiveness of our approach is reducing the number of candidate pairs of edges that can cross in a 1-planar drawing, C. To this end, consider a potential crossing between edges $e=(u,v)\in E$ and $f=(x,y)\in E$. In certain cases, it might be possible to exchange the positions of u and x (*swap* u and x) while keeping the remaining vertices of G unchanged so as to eliminate the crossing. Consider the neighborhoods (set of adjacent vertices) of u and x, that is, $N(u)=\{z:(u,z)\in E\}$ and $N(x)=\{z:(x,z)\in E\}$. If $N(u)\setminus\{v,x,y\}$ coincides with $N(x)\setminus\{u,v,y\}$ and at most one edge out of $(x,v),(u,y)$ is present in E, then one can swap u and x and eliminate the crossing from the drawing; see Fig. 3a. Hence, we may assume that $\chi(e,f)=\texttt{false}$ and remove the pair from the candidate set C. An extension of the rule can be applied for multiple crossing pairs. For example, if there are two crossings between $(u,v)\in E$ and $(a,y)\in E$ and between $(c,b)\in E$ with $(x,y)\in E$ and $N(u)=N(x)$ (see Fig. 3b), then again swapping u and x reduces the number of crossings. In our implementation, we enumerate all possible pairs of crossings and test if swapping two vertices reduces the number of crossings; the corresponding 2-clauses forbidding one of the crossings are then added to the formula.

Finally, we observe that when a 1-planar drawing of a graph exists, it is not unique. In other words, there might be multiple satisfying assignments for a formula. To reduce the search space explored by a SAT solver, we employ *symmetry breaking* techniques; a similar approach has been used by Bekos, Kaufmann, and Zielke [9] in the context of finding stack layouts of graphs. It is easy to see that one chosen vertex, say $v_0 \in V$, can be assumed to be the first in the order. This is enforced by constraints $\sigma(v_0, u) = \texttt{true}$ for all $u \in V' \setminus v_0$. Furthermore, we may force a relative order for *twin* vertices $u, v \in V'$ whose neighborhoods are identical; that is, $\sigma(u, v) = \texttt{true}$ for pairs of twins u, v. Last but not least, we integrate into the implementation of `OOPS` two generic symmetry-breaking engines, that can analyze the entire formula and add extra constraints without changing its satisfiability.

2.5 Extensions

It is possible to extend the schemes provided in Sections 2.3.1 and 2.3.2 to several other tasks in the area of 1-planarity. Recall that a graph is NIC-planar (resp., IC-planar) if it admits a 1-planar drawing in which any two pairs of crossing edges share at most one vertex (resp., no vertices). This can be straightforwardly formulated using the following constraints: Let $d_i \in D$ be a division vertex adjacent to original vertices u_i and v_i. For each quadruple of distinct division vertices d_i, d_j, d_k, d_t such that $|\{u_i, v_i, u_j, v_j\} \cap \{u_k, v_k, u_t, v_t\}| > 1$ (resp., > 0), we introduce constraint $\neg\chi(d_i, d_j) \vee \neg\chi(d_k, d_t)$ to forbid a crossing for one of the pairs.

Another interesting extension is for *directed acyclic* graphs (DAG), whose upward 1-planarity has been recently studied [3, 4]. In this context, the input is a DAG and the question is whether it admits an *upward 1-planar* drawing, that is, an upward drawing with at most one crossing per edge; a drawing is upward if for every (directed) edge (u, v), the y-coordinate of u is less than the y-coordinate of v. Our Hanani-Tutte encoding scheme can be applied for the setting by enforcing additional directional constraints $\sigma(u, v) = \texttt{true}$ for every edge (u, v) of the DAG.

3 Experiments

To validate the effectiveness of `OOPS`, we design an experimental evaluation focusing on three objectives. (i) Analyze how well the algorithm performs in terms of quality and scalability compared to alternatives. (ii) Study how available parameters of the algorithm contribute to its performance. (iii) Show how `OOPS` can be used as a discovery tool to answer open research questions in graph drawing and topological graph theory.

The experiments presented in this section were conducted in two modes referred to as *single-core* and *parallel*. For the former, we utilize a Linux-based laptop with a 3.6 GHz Intel Core Ultra 5 125U processor having 16GB RAM. For the latter, we use a more powerful server with a dual-node multi-core 3.2 GHz Intel Xeon Platinum 8488C (Sapphire) processor having 360GB RAM. Single-core computations are based on the (sequential) `MapleGlucose` SAT solver [39, 43], while parallel experiments use up to 32 cores of the server machine running `Painless` SAT solver [30, 44]. The implementation also supports two optional integrated symmetry-breaking engines, `BreakID` [18] and `satsuma` [2].

Our algorithm, `OOPS`, is fully open sourced [36]; the implementation is self-contained (except for the optional parallel mode, which requires an external parallel SAT solver) and can be built with a `C++` compiler having `C++17` support.

Table 1: Comparison of various heuristics for 1-planarity on real-world and synthetic benchmarks. The runtimes (in seconds) are represented by mean values along with 95% confidence intervals. The best results in each category are bold.

benchmark	algorithm	1-planar instances	1-planar runtime, sec	non-1-planar instances	non-1-planar runtime, sec	unsolved instances
NORTH50	`1PlanarTester`	117	0.15	26	0.15	154
	`brute-force`	156	8.0 ± 12.8	28	0.1 ± 0.1	113
	`OOPS` (H-T)	190	3.9 ± 2.6	29	0.1 ± 0.1	78
	`OOPS` (Stack)	190	0.2 ± 0.1	89	9.5 ± 5.9	18
	`OOPS` (Parallel)	**191**	1.1 ± 0.2	**105**	42 ± 58	**1**
ROME50	`1PlanarTester`	412	0.15	0		2538
	`brute-force`	1961	12.5 ± 6.5	0		989
	`OOPS` (H-T)	2815	7.8 ± 0.7	0		135
	`OOPS` (Stack)	2887	1.3 ± 0.3	0		63
	`OOPS` (Parallel)	**2893**	1.8 ± 0.2	**54**	319 ± 210	**3**
3REG-GIR7	`brute-force`	0		0		546
	`OOPS` (H-T)	41	105 ± 15	0		505
	`OOPS` (Stack)	411	75 ± 5	0		135
	`OOPS` (Parallel)	**545**	14 ± 2	0		**1**
5REG-B	`brute-force`	0		0		100
	`OOPS` (H-T)	0		1	160	99
	`OOPS` (Stack)	0		73	90 ± 12	27
	`OOPS` (Parallel)	0		**100**	39 ± 4	**0**

3.1 Comparison with Alternatives

Here we analyze how the SAT-based 1-planarity solver compares to alternative solutions. Arguably the simplest approach for testing 1-planarity is based on an exhaustive search, which enumerates all possible combinations of crossing edges, inserts a dummy vertex for every crossing, and tests whether the resulting graph is planar. We call the heuristic `brute-force`; refer to [36] for the implementation. Another candidate solver is by Binucci, Didimo, and Montecchiani [10], which we refer to as `1PlanarTester`. Since the solver is not open sourced[2], we use the same datasets and report the measurements presented in the paper [10]. The algorithms are compared with two versions of our SAT-based solver, described in Section 2.3.1 and Section 2.3.2; they are referred to as `OOPS` (H-T) and `OOPS` (Stack), respectively[3].

We use the following datasets to compare the solvers:

- NORTH50 is a subset of the NORTH benchmark [47] containing all instances with at most 50 vertices; all planar graphs have been removed from the collection, resulting in a total of 297 instances;

[2]There exists an implementation of the algorithm at `https://github.com/seemanne/1PlanarTester` but a close inspection of the source code reveals substantial gaps both in performance and correctness; thus, we do not use it in our evaluation.

[3]Our comparison excludes a new heuristic that is developed by Fink, Münch, Pfretzschner, and Rutter [22].

- ROME50 is a set of all 2950 non-planar ROME graphs [47] with at most 50 vertices.

In addition to the real-world graphs, we selected two datasets of graphs containing 45 edges. The synthetic graph data is generated with the `geng` tool from `nauty` [31, 45].

- 3REG-GIR7 are all 546 connected cubic (3-regular) graphs with $n = 30, m = 45$ and having girth 7; all but one instance are known to be 1-planar;
- 5REG-B is a random sample of 100 connected 5-regular bipartite graphs with $n = 18, m = 45$; all the graphs are non-1-planar.

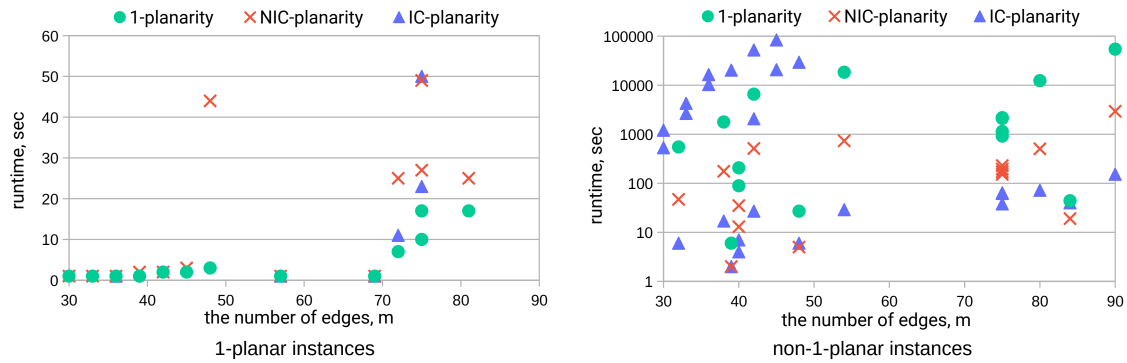
We report the results of the experiment in Table 1, which contains the number of identified 1-planar and non-1-planar instances, along with the mean processing times (in seconds) and 95% confidence intervals, for each of the benchmarks. To run the `brute-force` and `OOPS` (H-T, Stack) solvers on the data, we use the *single-core* mode (which is comparable to the configuration reported in [10]) and set the time limit for each execution to 3 *minutes*; for comparison, the limit for `1PlanarTester` in [10] is 3 *hours*. The `OOPS` (Parallel) entries correspond to the *parallel* mode of the solver with a limit of 3 hours per instance, matching the limit used in [10]. Note that the `OOPS` (Stack) and `OOPS` (Parallel) configurations use the same Stack-based encoding but different solving resources. We include both to show two usage scenarios: a near-online single-core run with a short timeout, and a parallel run on a powerful machine with a more generous time limit, which is especially helpful for negative instances.

One prominent observation from the results is that there is a large gap between SAT-based approaches (`OOPS`) and the naïve ones (`brute-force` and `1PlanarTester`). In most of the cases, `OOPS` is able to solve substantially more instances, both in the 1-planar and non-1-planar category. The relatively low runtime of `1PlanarTester` on NORTH50 and ROME50 suggests that it can handle only "easy" instances that do not require exploring a large search space. Similarly, the performance of `brute-force` on ROME50 is explained by the fact that the algorithm enumerates crossing sets in increasing order of their size. Within the time limit, the search typically reaches only crossing sets of size at most three; nevertheless, this already solves about two thirds of the instances (see Table 1). This indicates that the collection contains many almost-planar graphs admitting 1-planar drawings with only a few crossings. The difference becomes striking for the "harder" collections, 3REG-GIR7 and 5REG-B, where the exhaustive search is not able to solve *any* instance, while `OOPS` (Stack, Parallel) solves a majority of cases. For this reason, we do not consider existing algorithms as a viable alternative to the new SAT-based technique.

A comparison between Hanani-Tutte and Stack-based SAT encodings reveals a significant advantage of the latter on all the benchmarks. While both encodings contain $\Theta\big((n+m)^2\big)$ variables and $\Theta\big((n+m)^3\big)$ constraints, the Stack-based encoding is more succinct, requiring approximately half the number of constraints. (Note however, that Hanani-Tutte-based encoding is more flexible, as discussed in Section 2.5). Thus, we use the Stack-based encoding as a default one in the following experiments.

3.2 Named Graphs

To further validate `OOPS` on a real-world benchmark, we apply the algorithm for a collection of named WIKIPEDIA graphs available at [46]. We filtered out all planar instances, and all dense instances with $m > 4n - 8$ that are obviously non-1-planar, and all instances with $m \geq 100$, which are likely too difficult for our algorithm. The remaining 43 graphs are tested with `OOPS` using the Stack-based encoding in the parallel mode. The status of 1-planarity, NIC-planarity, and



(a) (b)

Figure 4: The runtime of `OOPS` (in seconds) on (a) 1-planar and (b) non-1-planar named Wikipedia graphs as a function of graph size, m. Observe that in (b) the runtime is on a logarithmic scale.

IC-planarity (refer to Section 2.5) are presented in Table 2. Here we enforce a time limit of 24 hours for the SAT solver.

To the best of our knowledge, this is the first study of 1-planarity for many of the graphs in the dataset. `OOPS` is able to determine the status in the majority of the cases. Interestingly, all the unsolved instances (marked TLE in the table) correspond to cubic (3-regular) graphs. To understand the performance of `OOPS` on the data, we plot its runtime as a function of the number of edges in the graph; see Fig. 4. We look separately at satisfiable instances (that is, 1-planar graphs) and non-satisfiable ones. Observe that on the benchmark data, *every* 1-planar graph takes less than 20 seconds (and at most 50 seconds for NIC/IC-planar instances). In contrast, most of the non-1-planar instances require substantially more time: the median runtime is around 4 *minutes*, the mean is over 1.5 *hours*, while the maximum (measured on IC-planarity of `Tutte-Coxeter`) is 23 *hours*. The memory consumption of a SAT solver (`Painless` in this case) follows the same trend: it is below $4GB$ of RAM for all 1-planar instances but reaches 50−80GB on the hardest non-1-planar graphs. Given the amount of RAM required for processing non-satisfiable instances, it seems tempting to explore alternative SAT solving strategies, such as cloud-based or incremental, which we leave as a possible future work.

We have also evaluated the encoding optimizations presented in Section 2.4. Table 3 reports the SAT solver runtimes on a selection of graphs. As a baseline, we consider the basic encoding (Section 2.3.2), and then apply symmetry breaking, search-space reduction techniques, and their combination (Section 2.4). While the optimizations have a mixed impact on the runtime of `OOPS` for 1-planar graphs, they are very effective for large non-1-planar instances. For example, proving the non-1-planarity of the `Robertson` graph requires 2300 seconds using the basic encoding, but only 1750 seconds with the optimizations—a 24% speed up. For larger instances, the improvement is even more pronounced: 30% (reduction from 2 hours to 85 minutes) for the `Brinkmann` graph, and 50% (from 10 to 5 hours) for `Holt`. A somewhat unexpected observation is that, in some cases, applying search-space reduction alone leads to a slowdown of the solver. A possible explanation is that the additional constraints increase the overall formula size and propagation overhead without providing sufficiently strong pruning of the search space; yet combining the two optimizations together yields a clear benefit.

Table 2: Non-planar named WIKIPEDIA graphs containing at most $4n-8$ edges and their 1-planarity status; taken from https://en.wikipedia.org/wiki/List_of_graphs_by_edges_and_vertices. The runtime column lists the times for the 1-planarity, NIC-planarity, and IC-planarity tests, respectively; TLE entries correspond to runtimes exceeding the limit of 24 hours.

graph name	n	m	degrees	1-plan.	NIC	IC	runtime		
Petersen	10	15	[3 . . . 3]	*yes*	*yes*	*yes*	1s	1s	1s
Franklin	12	18	[3 . . . 3]	*yes*	*yes*	*yes*	1s	1s	1s
Tietze	12	18	[3 . . . 3]	*yes*	*yes*	*yes*	1s	1s	1s
Grotzsch	11	20	[3 . . . 5]	*yes*	*no*	*no*	1s	5s	1s
Heawood	14	21	[3 . . . 3]	*yes*	*yes*	*yes*	1s	1s	1s
Chvatal	12	24	[4 . . . 4]	*yes*	*yes*	*no*	1s	1s	1s
Möbius–Kantor	16	24	[3 . . . 3]	*yes*	*yes*	*yes*	1s	1s	1s
Sousselier	16	27	[3 . . . 5]	*yes*	*yes*	*no*	1s	1s	10s
Blanusa Snark (1st)	18	27	[3 . . . 3]	*yes*	*yes*	*yes*	1s	1s	1s
Blanusa Snark (2nd)	18	27	[3 . . . 3]	*yes*	*yes*	*yes*	1s	1s	1s
Pappus	18	27	[3 . . . 3]	*yes*	*yes*	*no*	1s	1s	10m
Desargues	20	30	[3 . . . 3]	*yes*	*yes*	*no*	1s	1s	20m
Flower snark (J5)	20	30	[3 . . . 3]	*yes*	*yes*	*no*	1s	1s	10m
Hoffman	16	32	[4 . . . 4]	*no*	*no*	*no*	10m	50s	5s
Loupekine snark (1st)	22	33	[3 . . . 3]	*yes*	*yes*	*no*	1s	1s	1.2h
Loupekine snark (2nd)	22	33	[3 . . . 3]	*yes*	*yes*	*no*	1s	1s	40m
McGee	24	36	[3 . . . 3]	*yes*	*yes*	*no*	1s	1s	4.6h
Nauru	24	36	[3 . . . 3]	*yes*	*yes*	*no*	1s	1s	2.9h
Robertson	19	38	[4 . . . 4]	*no*	*no*	*no*	30m	1m	20s
Paley-13	13	39	[6 . . . 6]	*no*	*no*	*no*	5s	1s	1s
F26A	26	39	[3 . . . 3]	*yes*	*yes*	*no*	1s	1s	5.6h
Clebsch	16	40	[5 . . . 5]	*no*	*no*	*no*	1m	10s	5s
Folkman	20	40	[4 . . . 4]	*no*	*no*	*no*	5m	30s	5s
Brinkmann	21	42	[4 . . . 4]	*no*	*no*	*no*	1.8h	10m	30s
Flower snark (J7)	28	42	[3 . . . 3]	*yes*	*yes*	*no*	1s	1s	30m
Coxeter	28	42	[3 . . . 3]	*yes*	?	*no*	1s	TLE	14.5h
Double-Star snark	30	45	[3 . . . 3]	*yes*	*yes*	*no*	1s	1s	5.8h
Tutte-Coxeter	30	45	[3 . . . 3]	?	?	*no*	TLE	TLE	23.2h
Shrikhande	16	48	[6 . . . 6]	*no*	*no*	*no*	30s	5s	5s
Dyck	32	48	[3 . . . 3]	*yes*	*yes*	*no*	1s	40s	8.1h
Holt	27	54	[4 . . . 4]	*no*	*no*	*no*	5.1h	10m	30s
Foster cage	30	75	[5 . . . 5]	*no*	*no*	*no*	20m	5m	1m
Meringer	30	75	[5 . . . 5]	*no*	*no*	*no*	40m	1m	1m
Robertson–Wegner	30	75	[5 . . . 5]	*no*	*no*	*no*	40m	5m	1m
Wong	30	75	[5 . . . 5]	*no*	*no*	*no*	20m	1m	40s
Szekeres snark	50	75	[3 . . . 3]	*yes*	*yes*	*yes*	20s	50s	50s
Watkins snark	50	75	[3 . . . 3]	*yes*	*yes*	*yes*	10s	30s	20s
Wells	32	80	[5 . . . 5]	*no*	*no*	*no*	3.4h	10m	1m
Ellingham–Horton (54)	54	81	[3 . . . 3]	*yes*	*yes*	?	20s	20s	TLE
Gray	54	81	[3 . . . 3]	?	?	?		TLE	
Klein (7-regular)	24	84	[7 . . . 7]	*no*	*no*	*no*	40s	20s	40s
Klein (cubic)	56	84	[3 . . . 3]	?	?	?		TLE	
Sylvester	36	90	[5 . . . 5]	*no*	*no*	*no*	15.1h	50m	1m

Table 3: The impact of optimizations on 1-planarity encoding: the runtime (in seconds) of `OOPS` (Parallel) on selected Wikipedia graphs. TLE entries exceed the time limit of 24 hours.

graph	1-planar	basic	symmetry breaking	search-sp. reduction	symmetry & reduction
Hoffman	*no*	1580	600	1600	540
Robertson	*no*	2300	2080	10500	1750
Clebsch	*no*	90	95	140	80
Folkman	*no*	18700	2000	TLE	310
Brinkmann	*no*	7600	8400	6100	5000
Holt	*no*	35700	22300	TLE	18300
Szekeres snark	*yes*	5	15	10	10
Watkins snark	*yes*	5	30	10	12
Ellingham–Horton (54)	*yes*	12	60	30	12

3.3 Applications

Arguably among the most natural applications of a tool for recognizing 1-planar graphs is finding the smallest non-1-planar instance in a family of graphs.

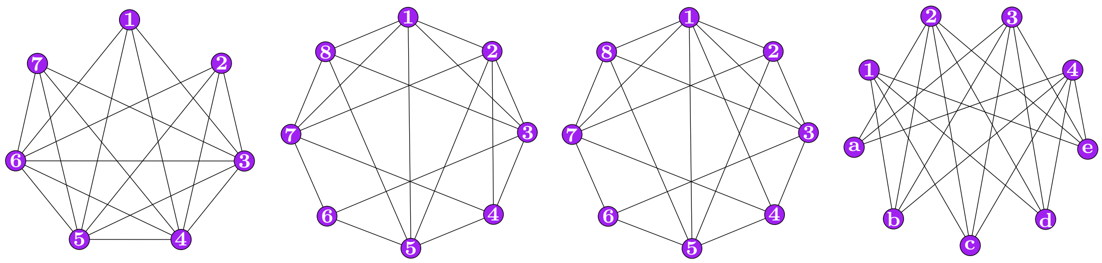
Problem 1. *What is the smallest non-1-planar instance in a family of graphs?*

This question has two flavors depending on whether one is interested in vertex-minimal or edge-minimal instances. For planar graphs, it is well-known that smallest non-planar graphs contain 5 vertices (K_5) or 9 edges ($K_{3,3}$). For 1-planar graphs, it is known that K_7 is the smallest complete non-1-planar graph (see e.g., [16]), however, no such bound is known for an edge-minimal non-1-planar graph. To fill the gap, we computationally tested all simple connected instances with up to $m = 18$ edges (generated with [45]). All graphs with up to $m = 17$ are 1-planar and there are exactly 3 (out of 164,551,477 connected) non-1-planar instances with $m = 18$; see Fig. 5. For bipartite graphs, a graph on 9 vertices (in fact, $K_{4,5}$ as shown by Czap and Hudák [16]) is one of the two vertex-minimal bipartite non-1-planar graphs. The second such graph, namely $K_{4,5} \setminus e$, is in addition the unique edge-minimal bipartite non-1-planar graph; see Fig. 5d. This is confirmed by testing all 25,124,511 connected bipartite graphs with $m = 19$ and all smaller instances. The average processing time per instance is about 2 milliseconds, corresponding to about 17 hours of total CPU time, or roughly half an hour on a 32-core machine.

A related question (discussed in [49]) is: *What is the minimum difference between edge and vertex counts in a connected non-1-planar graph?* For planar graphs, one can show (see [49]) that every connected graph with $m \leq n + 2$ is planar, while non-planar $K_{3,3}$ has $m = n + 3$. For 1-planarity, such a condition is unknown. Existing non-1-planar graphs, such as $K_{4,5}$ and $K_{3,7}$ [16], indicate that the difference can be $m - n = 11$. The two examples we found (Figs. 5a and 5d) reduce the bound to 10. It is intriguing to fully resolve the question.

A particularly challenging variant of Problem 1 is to find the smallest 3-regular (cubic) graph that is not 1-planar. While it is possible to show that certain large instances are not 1-planar (e.g., consider a large cubic *random* graph [19] or the *cube-connected cycle* of large order [38]), the smallest such instance is unknown.

Problem 1b ([48, 50]). *What is the smallest cubic non-1-planar graph?*



(a) (b) (c) (d)

Figure 5: (a-c) The three connected non-1-planar graphs with $m = 18$ edges. (d) The unique connected bipartite non-1-planar graph with $m = 19$; it is obtained from $K_{4,5}$ by removing an edge.

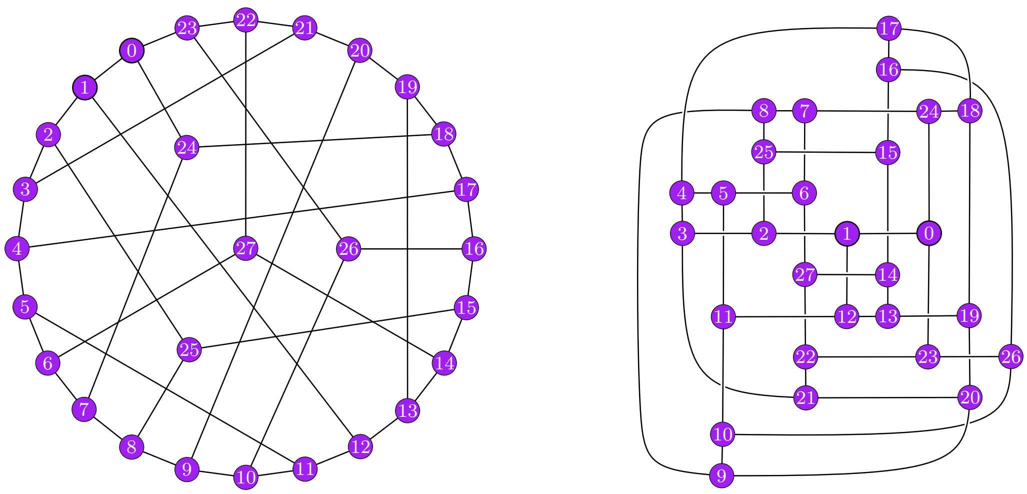


Figure 6: The `Coxeter` graph with $n = 28$ vertices and $m = 42$ edges (left) and its 1-planar drawing (right) constructed by `OOPS`.

In an attempt to answer the question, Eppstein [48] manually constructed 1-planar drawings of several graphs and suggested that either the `Coxeter` graph or the `Tutte-Coxeter` graph is not 1-planar. As pointed out in Table 2, the former is indeed 1-planar; see Fig. 6 for a 1-planar drawing. The latter is very likely non-1-planar, but we were unable to fully verify this, despite running several parallel SAT solvers multiple days. We did, however, verify the 1-planarity of several families of cubic graphs; refer to Table 4. In particular, (i) all cubic graphs with up to $n = 24$ vertices and (ii) all cubic bipartite graphs with up to $n = 28$ vertices are 1-planar. Note that there are around 150×10^6 instances across (i) and (ii), and the total computation took ≈ 1.5 machine-months, averaging less than a second per instance. Thus, verifying the 1-planarity of all cubic graphs with $n < 30$ remains a challenging task that will likely require novel ideas.

As another application of `OOPS`, we investigate the following question proposed by Zhang, Ouyang, and Huang [40] regarding *claw-free* 1-planar graphs; recall that a claw is graph $K_{1,3}$.

Problem 2 ([40]). *Do there exist infinitely many 6-connected claw-free 1-planar graphs?*

Table 4: Testing 1-planarity of cubic graphs with OOPS (Parallel).

graph class	total count	total runtime	runtime per instance	all 1-planar?
$n = 22$	7,319,447	5h 20m	85 ms	*yes*
$n = 24$	117,940,535	6d 9h 40m	150 ms	*yes*
bipartite $n = 26$	245,627	25m	200 ms	*yes*
girth≥ 5 $n = 26$	31,478,584	3d 6h 20m	290 ms	*yes*
$n = 26$	2,094,480,864			?
bipartite $n = 28$	2,291,589	6h 55m	340 ms	*yes*
girth≥ 6 $n = 28$	4,624,501	2d 23h 20m	1700 ms	*yes*
$n = 28$	40,497,138,011			?
bipartite $n = 30$	23,466,857			?

In fact, the original version of the paper [40] suggests a candidate family, namely the cube of a cycle C_n. That is, a graph with vertex set $\{v_0, v_1, \ldots, v_{n-1}\}$ in which an edge (v_i, v_j) exists if $\min(|i-j|, n-|i-j|) \leq 3$. Such graphs are (i) claw-free, since the neighborhood of any vertex is an interval on the cycle, so no vertex can have three pairwise non-adjacent neighbors, and (ii) 6-connected (for $n \geq 7$), since to disconnect two vertices, u and v, one needs to remove three consecutive vertices going clockwise from u to v and the same counter-clockwise. However, the status of 1-planarity of such graphs is open. Using OOPS, we verified that such graphs are indeed 1-planar whenever $n = 8k$ for every $k > 0$.

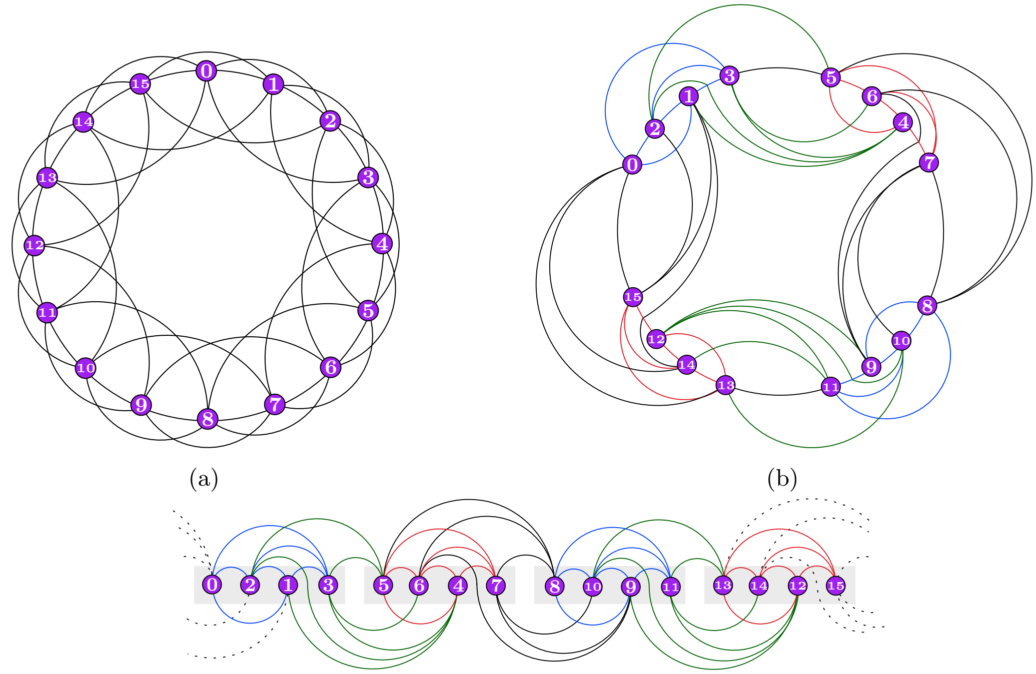
Theorem 3. *Let C_n^3 be a graph obtained from a cycle C_n by connecting every pair of vertices that are at most 3 edges apart in C_n. Then C_n^3 is 1-planar if $n = 8k$ for every $k > 0$.*

Proof: Let the vertices of C_n^3 for $n = 8k$ be $\{v_0, v_1, \ldots, v_{n-1}\}$. Split the vertices into $2k$ (consecutive) groups of size four, that is, $\{v_0, v_1, v_2, v_3\}, \ldots, \{v_{n-4}, v_{n-3}, v_{n-2}, v_{n-1}\}$; observe that all the edges of C_n^3 are either within the same group or between vertices of two consecutive groups (modulo $2k$). A 1-planar drawing is built by placing the vertices on a horizontal line (spine), following the order given by the groups. Within the i-th group, $\{v_x, v_{x+1}, v_{x+2}, v_{x+3}\}$, place the four vertices in the order $v_x, v_{x+2}, v_{x+1}, v_{x+3}$ if i is even, and in the order $v_{x+1}, v_{x+2}, v_x, v_{x+3}$ if i is odd; see Fig. 7. Because of the symmetry, we only need to specify the drawing of the edges between groups i and $i+1$, and between groups $i+1$ and $i+2$. Refer to Fig. 7c for the construction; note that most edges are represented as semi-arcs (either above or below the spine) except for one inter-group edge for every pair of groups, which is a bi-arc, e.g., edges $(2, 4)$, $(6, 9)$, and $(10, 12)$ in the figure. □

According to OOPS (for small values of n), graph C_n^3 is 1-planar only if $n = 8k, k > 0$ but proving that direction might be challenging.

4 Conclusion

We presented OOPS, a new SAT-based algorithm for testing 1-planarity and provided its implementation [36] to facilitate further research in the field. While the algorithm is shown to be useful for resolving various problems concerning 1-planarity of specific graph families, many questions remain unanswered. One particularly challenging task, discussed in Section 3.3, is finding the smallest cubic graph that is not 1-planar.



(a) (b) (c)

Figure 7: An illustration for Theorem 3: (a) C_{16}^3 and (b) its 1-planar drawing. (c) The edge pattern is generalized for arbitrary $n = 8k, k > 0$.

We conjecture that the `Tutte-Coxeter` graph (which is the smallest cubic graph of girth 8) and one other 30-vertex graph of girth 7 are such instances; none of the tested (parallel) SAT solvers was able to find a solution even after a week of processing. We stress that proving the minimality would also require finding a 1-planar drawing for more than *forty billion* 28-vertex cubic graphs.

Another intriguing direction to explore is 1-planarity of DAGs; see Section 2.5 for an adaptation of `OOPS` for the case. Among many possible questions to study (refer for example to [3, 32]), we highlight one discussed by Angelini et al. [3, 4]:

Problem 3 ([3, 4])**.** *Is there a directed acyclic outerpath that is not upward 1-planar?*

Finally, extending our formulations or designing new ones for other classes of beyond-planar graphs, such as k-planar graphs for $k > 1$, is of interest. While there is a straightforward adaptation of our scheme to 2-planarity (by inserting more division vertices per edge and enforcing appropriate merging rules), it might not be effective in practice due to the explosion of the search space. Hence, designing further techniques for simplifying or restricting the resulting SAT formulas, in addition to the ones discussed in Section 2.4, is an avenue for future research.

References

[1] M. O. Albertson. Chromatic number, independence ratio, and crossing number. *Ars Math. Contemp.*, 1(1):1–6, 2008. doi:10.26493/1855-3974.10.2D0.

[2] M. Anders, S. Brenner, and G. Rattan. Satsuma: Structure-based symmetry breaking in SAT. In S. Chakraborty and J. R. Jiang, editors, *Theory and Applications of Satisfiability Testing*, volume 305 of *LIPIcs*, pages 4:1–4:23, 2024. doi:10.4230/LIPICS.SAT.2024.4.

[3] P. Angelini, T. Biedl, M. Chimani, S. Cornelsen, G. Da Lozzo, S.-H. Hong, G. Liotta, M. Patrignani, S. Pupyrev, I. Rutter, and A. Wolff. The price of upwardness. In *International Symposium on Graph Drawing and Network Visualization (GD)*, volume 320, pages 13:1–13:20, 2024. doi:10.4230/LIPIcs.GD.2024.13.

[4] P. Angelini, T. Biedl, M. Chimani, S. Cornelsen, G. D. Lozzo, S.-H. Hong, G. Liotta, M. Patrignani, S. Pupyrev, I. Rutter, and A. Wolff. The price of upwardness. *Discret. Math. Theor. Comput. Sci.*, vol. 27:3, 2025. doi:10.46298/dmtcs.15222.

[5] C. Auer, C. Bachmaier, F. J. Brandenburg, A. Gleißner, K. Hanauer, D. Neuwirth, and J. Reislhuber. Outer 1-planar graphs. *Algorithmica*, 74(4):1293–1320, 2016. doi:10.1007/S00453-015-0002-1.

[6] C. Auer, F. Brandenburg, A. Gleißner, and J. Reislhuber. 1-planarity of graphs with a rotation system. *Journal of Graph Algorithms and Applications*, 19(1):67–86, 2015. doi:10.7155/JGAA.00347.

[7] C. Bachmaier, F. J. Brandenburg, K. Hanauer, D. Neuwirth, and J. Reislhuber. NIC-planar graphs. *Discret. Appl. Math.*, 232:23–40, 2017. doi:10.1016/J.DAM.2017.08.015.

[8] M. J. Bannister, S. Cabello, and D. Eppstein. Parameterized complexity of 1-planarity. *J. Graph Algorithms Appl.*, 22(1):23–49, 2018. doi:10.7155/JGAA.00457.

[9] M. A. Bekos, M. Kaufmann, and C. Zielke. The book embedding problem from a SAT-solving perspective. In E. D. Giacomo and A. Lubiw, editors, *Graph Drawing and Network Visualization GD*, volume 9411 of *Lecture Notes in Computer Science*, pages 125–138. Springer, 2015. doi:10.1007/978-3-319-27261-0_11.

[10] C. Binucci, W. Didimo, and F. Montecchiani. 1-planarity testing and embedding: An experimental study. *Comput. Geom.*, 108:101900, 2023. doi:10.1016/J.COMGEO.2022.101900.

[11] R. Bodendiek, H. Schumacher, and K. Wagner. Über 1-optimale graphen. *Mathematische Nachrichten*, 117(1):323–339, 1984. doi:10.1002/mana.3211170125.

[12] O. V. Borodin. Solution of the Ringel problem on vertex-face coloring of planar graphs and coloring of 1-planar graphs. *Metody Diskret. Analiz.*, 41(108):12–26, 1984.

[13] F. J. Brandenburg. Characterizing and recognizing 4-map graphs. *Algorithmica*, 81(5):1818–1843, 2019. doi:10.1007/S00453-018-0510-X.

[14] S. Cabello and B. Mohar. Adding one edge to planar graphs makes crossing number and 1-planarity hard. *SIAM J. Comput.*, 42(5):1803–1829, 2013. doi:10.1137/120872310.

[15] M. Chimani and R. Zeranski. Upward planarity testing in practice: SAT formulations and comparative study. *ACM J. Exp. Algorithmics*, 20:1–27, 2015. doi:10.1145/2699875.

[16] J. Czap and D. Hudák. 1-planarity of complete multipartite graphs. *Discret. Appl. Math.*, 160(4–5):505–512, 2012. doi:10.1016/J.DAM.2011.11.014.

[17] J. Czap and D. Hudák. On drawings and decompositions of 1-planar graphs. *Electron. J. Comb.*, 20(2):P54, 2013. doi:10.37236/2392.

[18] J. Devriendt and B. Bogaerts. BreakID: Static symmetry breaking for ASP (system description). *arXiv preprint arXiv:1608.08447*, 2016.

[19] V. Dujmović, K.-i. Kawarabayashi, B. Mohar, and D. R. Wood. Improved upper bounds on the crossing number. In *Proceedings of the Twenty-Fourth Annual Symposium on Computational Geometry*, pages 375–384. ACM, 2008. doi:10.1145/1377676.1377739.

[20] E. Eiben, R. Ganian, T. Hamm, F. Klute, and M. Nöllenburg. Extending partial 1-planar drawings. In A. Czumaj, A. Dawar, and E. Merelli, editors, *International Colloquium on Automata, Languages, and Programming*, LIPIcs, pages 43:1–43:19, 2020. doi:10.4230/LIPIcs.ICALP.2020.43.

[21] S. Felsner, É. Fusy, M. Noy, and D. Orden. Bijections for Baxter families and related objects. *Journal of Combinatorial Theory, Series A*, 118(3):993–1020, 2011. doi:10.1016/j.jcta.2010.03.017.

[22] S. D. Fink, M. Münch, M. Pfretzschner, and I. Rutter. Heuristics for exact 1-planarity testing. In V. Dujmovic and F. Montecchiani, editors, *33rd International Symposium on Graph Drawing and Network Visualization, GD 2025*, volume 357 of *LIPIcs*, pages 4:1–4:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPICS.GD.2025.4.

[23] R. Fulek, M. J. Pelsmajer, M. Schaefer, and D. Štefankovič. Hanani-Tutte, monotone drawings, and level-planarity. In J. Pach, editor, *Thirty Essays on Geometric Graph Theory*, pages 263–287. Springer, 2013. doi:10.1007/978-1-4614-0110-0_14.

[24] A. Grigoriev and H. L. Bodlaender. Algorithms for graphs embeddable with few crossings per edge. *Algorithmica*, 49(1):1–11, 2007. doi:10.1007/s00453-007-0010-x.

[25] S.-H. Hong, P. Eades, N. Katoh, G. Liotta, P. Schweitzer, and Y. Suzuki. A linear-time algorithm for testing outer-1-planarity. *Algorithmica*, 72:1033–1054, 2015. doi:10.1007/s00453-014-9890-8.

[26] Y. Huang, Z. Ouyang, and F. Dong. On the sizes of bipartite 1-planar graphs. *Electron. J. Comb.*, 28(2):P2.22, 2021. doi:10.37236/10012.

[27] D. V. Karpov. An upper bound on the number of edges in an almost planar bipartite graph. *Journal of Mathematical Sciences*, 196:737–746, 2014. doi:10.1007/s10958-014-1690-9.

[28] S. G. Kobourov, G. Liotta, and F. Montecchiani. An annotated bibliography on 1-planarity. *Computer Science Review*, 25:49–67, 2017. doi:10.1016/j.cosrev.2017.06.002.

[29] V. P. Korzhik and B. Mohar. Minimal obstructions for 1-immersions and hardness of 1-planarity testing. *Journal of Graph Theory*, 72(1):30–71, 2013. doi:10.1002/jgt.21630.

[30] L. Le Frioux, S. Baarir, J. Sopena, and F. Kordon. PaInleSS: A framework for parallel SAT solving. In *Theory and Applications of Satisfiability Testing*, pages 233–250. Springer, 2017. doi:10.1007/978-3-319-66263-3_15.

[31] B. D. McKay and A. Piperno. Practical graph isomorphism, ii. *Journal of Symbolic Computation*, 60:94–112, 2014. doi:10.1016/j.jsc.2013.09.003.

[32] M. Nöllenburg and S. Pupyrev. On families of planar dags with constant stack number. In M. A. Bekos and M. Chimani, editors, *Graph Drawing and Network Visualization (GD)*, pages 135–151. Springer, 2023. doi:10.1007/978-3-031-49272-3_10.

[33] J. Pach and G. Tóth. Monotone drawings of planar graphs. *Journal of Graph Theory*, 46(1):39–47, 2004. doi:10.1002/jgt.10168.

[34] S. Pupyrev. A collection of existing results on stack and queue numbers. https://spupyrev.github.io/linearlayouts.html, 2020. Accessed: 5/23/2025.

[35] S. Pupyrev. OOPS: optimized one-planarity solver via SAT. In V. Dujmovic and F. Montecchiani, editors, *33rd International Symposium on Graph Drawing and Network Visualization, GD 2025*, volume 357 of *LIPIcs*, pages 14:1–14:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPICS.GD.2025.14.

[36] S. Pupyrev. Optimized One-Planarity Solver. https://github.com/spupyrev/oops, 2025.

[37] G. Ringel. Ein Sechsfarbenproblem auf der Kugel. In *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, volume 29, pages 107–117. Springer, 1965. doi:10.1007/BF02996313.

[38] O. Sýkora and I. Vrto. On crossing numbers of hypercubes and cube connected cycles. *BIT*, 33(2):232–237, 1993. doi:10.1007/BF01989746.

[39] V. Vallade, L. Le Frioux, R. Oanea, S. Baarir, J. Sopena, F. Kordon, S. Nejati, and V. Ganesh. New concurrent and distributed painless solvers: p-mcomsps, pmcomsps-com, p-mcomsps-mpi, and p-mcomsps-com-mpi. *SAT Competition*, page 40, 2021.

[40] L. Zhang, Z. Ouyang, and Y. Huang. Maximum degree and connectivity on claw-free 1-planar graphs, 2025. v1 accessed: 1/25/2025. URL: https://arxiv.org/abs/2501.15124v1, arXiv:2501.15124.

[41] X. Zhang. Drawing complete multipartite graphs on the plane with restrictions on crossings. *Acta Mathematica Sinica, English Series*, 30(12):2045–2053, 2014. doi:10.1007/s10114-014-3763-6.

[42] X. Zhang and G. Liu. The structure of plane graphs with independent crossings and its applications to coloring problems. *Central European Journal of Mathematics*, 11:308–321, 2013. doi:10.2478/s11533-012-0094-7.

[43] MapleGlucose SAT solver. https://maplesat.github.io/solvers.html, 2021.

[44] Painless SAT solver. https://www.lrde.epita.fr/wiki/Painless, 2021.

[45] nauty and Traces. https://pallini.di.uniroma1.it. Accessed: 5/23/2025.

[46] List of graphs by edges and vertices. https://en.wikipedia.org/wiki/List_of_graphs_by_edges_and_vertices. Accessed: 5/23/2025.

[47] AT&T and Rome graphs. http://graphdrawing.org/data.html. Accessed: 5/23/2025.

[48] David Eppstein. Cubic 1-planarity. https://11011110.github.io/blog/2014/05/11/cubic-1-planarity.html. Accessed: 5/23/2025.

[49] Mathematics StackExchange. Lower bound on the number of edges in a non-k-planar graph. https://math.stackexchange.com/questions/4637623. Accessed: 5/23/2025.

[50] MathOverflow. Can a 3-regular non-1-planar graph be constructed? https://mathoverflow.net/questions/441370. Accessed: 5/23/2025.