Planar Stories of Graph Drawings: Algorithms and Experiments

Carla Binucci
University of Perugia, Italy

Walter Didimo
University of Perugia, Italy

Eleni Katsanou
National Technical University of Athens, Greece

Antonios Symvonis
National Technical University of Athens, Greece

Sabine Cornelsen
University of Konstanz, Germany

Seok-Hee Hong
University of Sydney, Australia

Maurizio Patrignani
Roma Tre University, Italy

Samuel Wolf
University of Würzburg, Germany

Abstract

We address the problem of computing a dynamic visualization of a geometric graph G as a sequence of frames. Each frame shows only a portion of the graph but their union covers G entirely. The two main requirements of our dynamic visualization are: (i) guaranteeing drawing stability, so to preserve the user's mental map; (ii) keeping the visual complexity of each frame low. To satisfy the first requirement, we never change the position of the vertices. Regarding the second requirement, we avoid edge crossings in each frame. More precisely, in the first frame we visualize a suitable subset of non-crossing edges; in each subsequent frame, exactly one new edge enters the visualization and all the edges that cross with it are deleted. We call such a sequence of frames a *planar story* of G. Our goal is to find a planar story whose minimum number of edges contemporarily displayed is maximized (i.e., a planar story that maximizes the minimum frame size). Besides studying our model from a theoretical point of view, we also design and experimentally compare different algorithms, both exact techniques and heuristics. These algorithms provide an array of alternative trade-offs between efficiency and effectiveness, also depending on the structure of the input graph.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Mathematics of computing → Graph theory; Theory of computation → Graph algorithms analysis

Keywords and phrases Graph Drawing, Dynamic Graphs, Graph Stories, Heuristics, ILP

Digital Object Identifier 10.4230/LIPIcs.GD.2025.32

Related Version *Full Version*: `https://arxiv.org/abs/2508.17532` [5]

Supplementary Material *Dataset*: `https://github.com/ekatsanou/edge-based-graph-stories`

Funding Carla Binucci and Walter Didimo were partially supported by the Italian Ministry of University and Research (MUR) under the PRIN Project no. 2022TS4Y3N – EXPAND and the PON Project RASTA, ARS01_00540; Maurizio Patrignani was partially supported by the Italian Ministry of University and Research (MUR) under PRIN Project no. 2022TS4Y3N - EXPAND.

Acknowledgements This work started at the Bertinoro Workshop on Graph Drawing BWGD 2025.

1 Introduction

Conveying the structure of a graph in a single visualization is the goal of many graph drawing algorithms. Among these, force-directed algorithms are the most popular and widely used in practice [28]. They produce layouts in which vertices are represented as points in the plane and edges as straight-line segments. However, when a graph is complex or locally dense, the layout may feature numerous edge crossings, which significantly reduce its readability, as also witnessed by several human cognitive experiments in graph drawing [34, 35, 36, 37].

This scenario naturally motivates strategies that either attempt to simplify the (single) graph visualization, for instance through graph sampling [16, 24, 32, 38, 41, 42] or edge bundling [2, 23, 31], or that attempt to "distribute" the nodes and edges of the graph across multiple visualizations, i.e., a sequence of frames, each showing only a portion of the graph (see, e.g., [7, 9, 13, 15]). While the first type of strategy focuses on a static visualization that may cause ambiguity or loss of information, computing a sequence of visualization frames allows for a dynamic exploration of the entire graph, without altering its structure. However, it should guarantee *drawing stability* (i.e., the layout of the portion of the graph shared by two consecutive frames should remain unchanged) and each frame should have *low visual complexity*, for example by guaranteeing planarity or few edges crossings.

Contribution. In this paper we address the second type of strategy mentioned above. We introduce a new model for the dynamic visualization of a static graph as a sequence of frames. This model assumes that the input is a geometric graph G (i.e., a graph drawn in the plane with straight-line edges, for example by some force-directed algorithm) and aims to generate a sequence of visualization frames such that: (i) the vertices of G are all present in any frame, always at their initial input coordinates; (ii) the initial frame contains a subset of edges that yields a planar subgraph; (iii) in each subsequent frame, exactly one new edge e enters the visualization and the minimum subset of edges of the current frame that together with e violate planarity disappear; (iv) when an edge leaves the visualization, it no longer appears in the future; (v) the union of all frames contains the whole edge set of G. We call such a sequence of frames a *planar story* of G (see Section 3 for a formal definition).

A natural objective function when computing a planar story is that the amount of edges in any frame is not too small. More precisely, we are interested in planar stories whose minimum number of edges contemporarily displayed (across the entire sequence of frames) is maximized. We call such a story *min-frame optimal*. Besides studying our model from a theoretical point of view, we design and experimentally compare different algorithms (both exact techniques and heuristics). Our main results are as follows:

- We show in Section 5 that a min-frame optimal planar story of a geometric graph G can be efficiently computed when G is 2-plane (i.e., each edge is crossed at most twice), although the problem is NP-hard in the general case (Section 4).
- We describe several heuristics based on a common greedy framework; all of them work for general graphs (Section 5). They differ in the strategy for computing the initial and the last frames, and in the criteria used to select the edge that enters the visualization in each subsequent frame. We also describe an Integer Linear Program (ILP) that solves the problem optimally (Section 6).
- We discuss the results of an extensive experimental analysis that compares our heuristics and the exact algorithm. The results highlight algorithmic trade-offs between efficiency and effectiveness, and show how some of our heuristics are able to achieve the optimum in many cases in a short runtime. See Section 7.

The paper concludes by discussing some future research directions (Section 8). Due to space limitations some proofs and technical details can be found in [5].

2 Related Work

The research in this paper is inspired by a recent topic in graph drawing named *graph stories*. A graph story corresponds to a temporal sequence of frames, each displaying only part of the graph and whose union covers all nodes and edges of the graph. The term "graph story"

was introduced by Borrazzo, Da Lozzo, Di Battista, Frati, and Patrignani [9]. Different from our scenario, in their model no drawing of the graph is provided in advance. The vertices enter the visualization one at a time and persist in the visualization for a fixed amount of time (i.e., for a given subsequence of frames). When a vertex enters the visualization its coordinates have to be computed. In any frame, the user sees only the graph induced by the displayed vertices; the drawing in each frame must be straight-line and planar, and the layout of the graph portion shared by two consecutive frames cannot change. The authors in [9] provide bounds on the area requirement of graph stories for paths and trees, and exhibit planar graphs that do not admit a graph story within their rules. The model in [9] has been further studied in [13, 12], by allowing edges to be Jordan curves rather than straight-line segments. Also, several authors proposed a different setting in which each vertex persists in the visualization until all its neighbors have been displayed [6, 7, 17]. In all the aforementioned papers, the main focus is on exploring the time complexity and theoretical questions, often restricting to very specific graph families. Conversely, our model works for general graphs and, besides providing a solid theoretical basis, our goal is to derive practical implementations with different trade-offs between effectiveness and efficiency.

Another problem related to our research has been studied in [14, 15]. Similarly to our setting, it assumes that the input is a straight-line drawing of a graph; the objective is to partition the edge set into the minimum number of frames, such that the drawing in each frame has some desired property (e.g., planarity). However, unlike our model, the visualization frames do not form a temporal sequence and each edge appears in just one frame.

Finally, we remark that representing a graph across a sequence of frames (using small multiples or animation approaches) is strongly related to the problem of visually conveying dynamic graphs (see [4] for a survey on this topic). However, a dynamic graph evolves over time, and the ordered sequence of elements that appear or disappear during the graph evolution is part of the input for the drawing algorithm. In our setting, the graph is static and the algorithm can decide the temporal sequence of edges for dynamically visualizing it.

3 Basic Definitions and Preliminary Considerations

Given a graph G, we denote by $V(G)$ and $E(G)$ the set of vertices and the set of edges of G, respectively. For a set $V' \subseteq V(G)$, let $N(V')$ be the set of neighbors of vertices in V' and let $G - V'$ be the subgraph of G induced by $V(G) \setminus V'$. Let G be a *geometric graph*, that is, each vertex $v \in V(G)$ is a point in the plane and each edge $e = uv \in E(G)$ is the straight-line segment connecting u and v. A *planar frame* F of G is any subgraph of G consisting of all the vertices of G and a subset of edges of G that do not cross. A *planar story* of G is a sequence $\sigma = \langle F_1, F_2, \ldots, F_\tau \rangle$ such that:

1. Each F_i is a planar frame of G, with $i \in \{1, \ldots, \tau\}$.
2. Each edge $e \in E(G)$ appears in a non-empty subsequence of consecutive planar frames.
3. For each $i \in \{2, \ldots, \tau\}$, F_i contains exactly one edge e of G that is not in the union of all frames F_j, with $j < i$, and exactly those edges of F_{i-1} that do not cross e.

Let $\Sigma(G)$ be the set of all planar stories of G. For a planar story $\sigma = \langle F_1, F_2, \ldots, F_\tau \rangle \in \Sigma(G)$, we call F_1 and F_τ the *initial frame* and the *last frame* of σ, respectively, let m_i be the *size* of F_i, that is, the number of edges in F_i. Let $\mu(\sigma) = \min\{m_1, \ldots, m_\tau\}$, and let $\mu(G)$ be the maximum $\mu(\sigma)$ over all planar stories σ of G, i.e., $\mu(G) = \max\{\mu(\sigma) \mid \sigma \in \Sigma(G)\}$. If $\mu(\sigma) = \mu(G)$ then σ is said to be *min-frame optimal*. We call MaxMinFramePlanarStory(G) the problem of computing a min-frame optimal planar story of G.

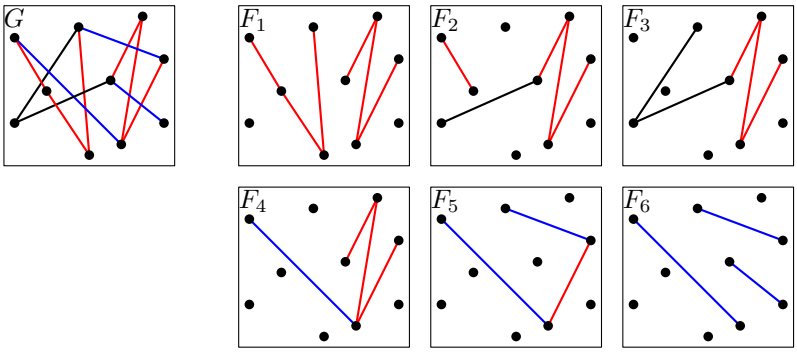


Figure 1 A planar story consisting of 6 frames. The top-leftmost figure shows the input geometric graph G; red edges and blue edges are those in the first frame and in the last frame, respectively. The black edges are the remaining edges.

▶ Remark 1. Since all the crossing-free edges of G appear in every frame of any planar story of G, we can safely disregard them when solving MaxMinFramePlanarStory(G). Hence, without loss of generality, we will assume that G does not contain crossing-free edges.

Fig. 1 shows a geometric graph G and a planar story σ of G, consisting of six frames and where $\mu(\sigma) = 3$. In the figure, the red edges are those of the initial frame, the blue edges are those of the final frame, and the black edges are the remaining edges.

▶ **Proposition 2.** *In any planar story $\sigma = \langle F_1, \ldots, F_\tau \rangle$, F_1 and F_τ are edge disjoint.*

Proof. Let e be an edge in F_1. By Remark 1, e is crossed by some edge e' of the graph. Let F_i be the first frame of σ that contains e', with $i \geq 2$. By definition of planar story, e does not occur in any frame F_j with $j \geq i$. In particular, e is not contained in F_τ. ◀

Given a planar frame F of G, let $\mu_F(G)$ be the maximum $\mu(\sigma)$ over all planar stories σ of G with initial frame F. Proposition 2 implies the following.

▶ **Corollary 3.** *Let F be a planar frame of G, and let F' be a maximum planar subgraph of $G - E(F)$. Then $\mu_F(G) \leq \min\{|E(F)|, |E(F')|\}$. Therefore, $\mu(G) \leq |E(G)|/2$.*

Planar Graph Stories and Crossing Graph. The *crossing graph* X of a geometric graph G is defined as follows: (i) the vertex set $V(X)$ corresponds to the edge set $E(G)$; (ii) there is an edge $uv \in E(X)$ if and only if the edges corresponding to u and v in $E(G)$ cross each other. The crossing-free edges of G correspond to isolated vertices of X; by Remark 1, we assume that X has no isolated vertices. Note that a graph is the crossing graph of a geometric graph if and only if it is a *segment intersection graph* [29]. This includes cliques, but also planar graphs [10, 19]. The crossing graph of a geometric graph with m edges and χ crossings can be computed in $\mathcal{O}(m \log m + \chi)$ time with the algorithm of Balaban [3] for segment intersection.

If $\sigma = \langle F_1, \ldots, F_\tau \rangle$ is a planar story of G, each frame F_i ($i \in \{1, \ldots, \tau\}$) corresponds to an *independent set* $X[F_i]$ of X, i.e., no two vertices of $X[F_i]$ are adjacent. We denote by $\sigma_X = \langle X[F_1], \ldots, X[F_\tau] \rangle$ the sequence that describes the planar story σ from the point of view of X. See Fig. 2 for an example. We will also refer to σ_X as a planar story and to the independent sets $X[F_i]$ as (planar) frames. If not stated otherwise, we use red for the initial frame and blue for the final frame in our figures. A *maximum pair of independent sets* is a pair (I_1, I_2) of two disjoint independent sets such that $\min\{|I_1|, |I_2|\}$ is maximum. A maximum pair (I_1, I_2) of independent sets is *Pareto optimal* if also $\max\{|I_1|, |I_2|\}$ is maximum. As a further corollary of Proposition 2, we obtain the following.

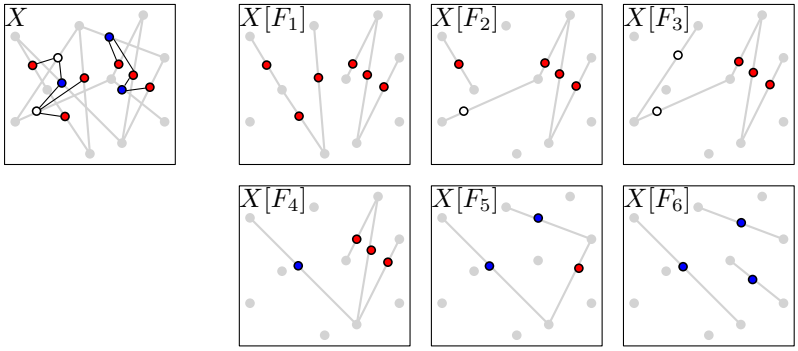
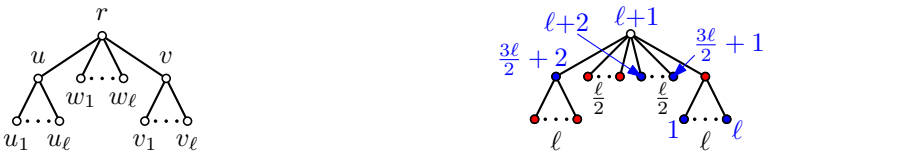
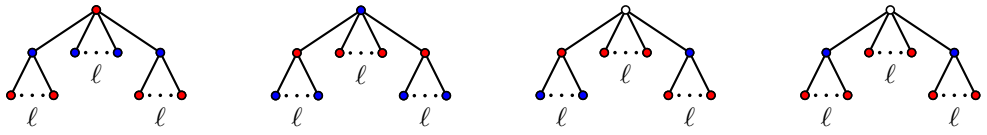


Figure 2 The crossing graph X of the graph G in Fig. 1 (top-left) and the planar story in Fig. 1 from the point of view of the crossing graph X of G.



(a) crossing graph **(b)** non-maximal initial frame, $\mu_F(G) = \mu(G) = \frac{3}{2}\ell + 1$



(c) $\mu_F(G) \leq |I'| = \ell + 2$ **(d)** $\mu_F(G) \leq |I| = \ell + 2$ **(e)** $\mu_F(G) \leq |I'| = \ell + 1$ **(f)** $\mu_F(G) \leq |I'| = 2$

Figure 3 The crossing graph X of some graph G. The set $I \subseteq V(X)$ of red vertices corresponds to the initial frame F. Blue vertices are a maximum independent set I' within $X - I$. (c)-(f) If we choose I as a maximal independent set, then there is a frame with at most $\min\{|I|, |I'|\} \leq \ell + 2$ edges. However, there is a planar story (b) with at least $\frac{3}{2}\ell + 1$ edges in any frame. Numbers close to the vertices indicate the order in which we add the respective edges into the planar story.

▶ **Corollary 4.** *Let G be a geometric graph, X be the crossing graph of G, and (I_1, I_2) be a* maximum pair of independent sets *of X. Then $\mu(G) \leq \min\{|I_1|, |I_2|\}$.*

We observe that, in order to get a min-frame optimal story of a geometric graph G, it might be necessary to choose as the initial frame a planar subgraph of G that is not a maximal planar subgraph. In terms of the crossing graph, this means that it might be necessary to start with an independent set that is not maximal. See Fig. 3 for an example where the crossing graph is a caterpillar.

4 Problem Complexity

In Section 5 we show how to solve MaxMinFramePlanarStory(G) optimally in linear time when G is a 2-plane graph and in cubic time if G is a 3-plane graph whose crossing graph has no cycles. We recall that a geometric graph is *k-plane* if each edge is crossed at most k times. However, MaxMinFramePlanarStory(G) is NP-hard in general. To this end, consider its decision version MaxMinFramePlanarStoryD(G,m): Given a graph G and a target value m, does G admit a planar story $\sigma = \{F_1, F_2, \ldots, F_\tau\}$ such that $\mu(\sigma) \geq m$?

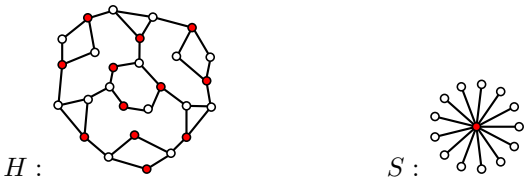


Figure 4 If the size of a maximum independent set of H is k then the size of the maximum frame containing the center of S – indicated by red vertices – is at most $k+1$.

▶ **Theorem 5.** *Problem* MaxMinFramePlanarStoryD(G,m) *is NP-complete.*

Proof. It is easy to see that MaxMinFramePlanarStoryD(G,m) is in NP. To prove the hardness, we apply a reduction from PlanarMaximumIndependentSetD(H,k), i.e., the problem of deciding whether a planar graph H admits an independent set of size at least k; this problem is known to be NP-hard [30, Theorem 4.1]. To this end, let X be the disjoint union of H and a star S with $k+1$ leaves (see Fig. 4). Since X is planar, it is a segment intersection graph [10], i.e., a crossing graph of some geometric graph G. We show that $\mu(G) \geq k+1 =: m$ if and only if H contains an independent set of size at least k.

Assume first that H contains an independent set I of size at least k. Consider the following planar story. The initial frame contains the edges corresponding to I and the center of the star S. The size of the initial frame is $k+1$. For the subsequent frames, we add the edges corresponding to the leaves of S, in any arbitrary order. This increases the size of the frame up to $2k+1$. Finally, we add in arbitrary order all edges corresponding to vertices of H that were not in I. During this process, the edges corresponding to the leaves of S will never be removed. Thus, the size of each frame is at least $k+1$.

Assume now that there is a planar story σ of G with $\mu(\sigma) \geq k+1$. We show that H contains an independent set of size at least k. Let F be the frame of σ that contains the edge corresponding to the center of S. In this frame, there is no edge corresponding to a leaf of S. This implies that frame F must contain at least k non-intersecting edges that correspond to at least k independent vertices of H. ◀

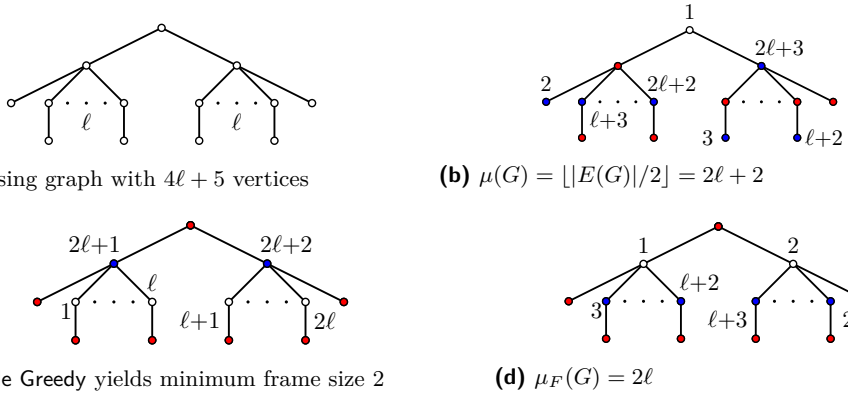
5 A Greedy Heuristic

We devise variants of a greedy heuristic for the problem MaxMinFramePlanarStory(G). The heuristic computes the frames one by one. Throughout the algorithm, we call the edges of the current frame *current edges*, the edges that were already removed *past edges*, and those that still have to be inserted *future edges*. The *current/future degree* of an edge e is the number of current/future edges that cross e. A simple greedy strategy may work as follows.

Simple Greedy

- Phase 1. Start with some initial planar frame F_1, and let $\tau = |E(G)| - |E(F_1)| + 1$ be the total number of frames in the story.
- Phase 2. For each step $i = 1, \ldots, \tau - 1$, let F_i be the current planar frame. Pick a future edge e with the minimum current degree and let F_{i+1} be the frame obtained from F_i by adding e and by removing the edges that are crossed by e.

Since Simple Greedy always first increases an initial frame to a maximal planar subgraph, the example in Fig. 3 implies that this strategy is not optimum in general. Moreover, Simple Greedy does not yield a constant-factor approximation even if the crossing graph is a tree; see the example in Fig. 5. The alternative order in Fig. 5d is better than the solution of Simple



(a) Crossing graph with $4\ell + 5$ vertices

(b) $\mu(G) = \lfloor |E(G)|/2 \rfloor = 2\ell + 2$

(c) Simple Greedy yields minimum frame size 2

(d) $\mu_F(G) = 2\ell$

Figure 5 The crossing graph X of some graph G. The vertex labels indicate the order in which we add the respective edges of G to the initial frame (red vertices). The two cases (c) and (d) in the bottom row start with the same initial frame F.

Greedy, as it ensures that the last frame remains sufficiently large. Hence, we introduce a refined version of the simple greedy strategy, described hereunder. We first describe the general strategy of the two phases of the Advanced Greedy algorithm. In the following we provide details about alternative variants for executing both of them.

Advanced Greedy

- Phase 1. Compute two edge-disjoint planar frames, the initial frame F_1 and the final frame F_τ, where $\tau = |E(G)| - |E(F_1)| + 1$.
- Phase 2. For each step $i = 1, \ldots, \tau - 1$, let F_i be the current planar frame. To make sure that all edges of F_τ are indeed in the final frame, we say that a future edge is *admissible* if it is (i) not in F_τ or (ii) not crossed by any other future edge. We pick an admissible edge e with the minimum current degree. Let F_{i+1} be the frame obtained from F_i by adding e and by removing the edges that are crossed by e.

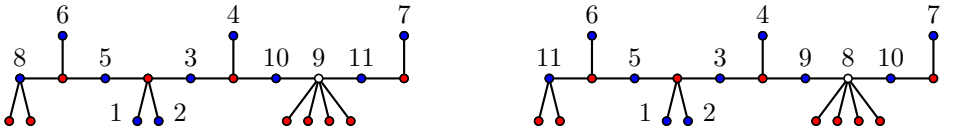
Note that Phase 2 of Advanced Greedy is related to the reconfiguration problem [25] that asks for a transformation between two feasible solutions of a problem such that all intermediate results are also feasible. In particular, [26] considered the problem of traversing between two given independent sets of the same size. However, in [26] at each step exactly one vertex is removed and one vertex is added, and vertices may disappear and reappear several times.

Alternative variants for Phase 1. We propose three alternative strategies for computing the initial frame F_1 and the last frame F_τ, or in other words two possibly large disjoint independent sets I_1 (initial frame) and I_τ (final frame) of the crossing graph X of G.

1a. (I_1, I_τ) is a Pareto optimal maximum pair of independent sets such that $|I_1| \leq |I_\tau|$. See Thm. 6 for the computation.

1b. I_1 is a large independent set of X of size at most $|E(G)|/2$ and I_τ is a maximal independent set of $X - I_1$. More precisely, we use the following approach: While $|I_1| \leq |E(G)|/2 - 1$ and $X - I_1 - N(I_1)$ is not empty, iteratively add a vertex of minimum degree in $X - I_1 - N(I_1)$ to I_1. Then, iteratively add a vertex of minimum degree in $X - I_1 - I_\tau - N(I_\tau)$ to I_τ.

1c. Alternating between $i = 1$ and $i = \tau$, iteratively add a vertex of minimum degree in $X - I_1 - I_\tau - N(I_i)$ (if there exists one) to I_i. If $|I_1| > |I_\tau|$, exchange I_1 and I_τ.



(a) greedy: 10,10,11,11,12,12,13,13,12,9,9,10 **(b)** optimal: 10,10,11,11,12,12,13,13,10,10,11,10

Figure 6 A Pareto-optimal maximum pair of independent sets in the crossing graph as initial and last frame with a non-optimal greedy story and a min-frame optimal story. The edges are added according to the numbers of the vertices in the crossing graph to the initial frame (red vertices). The subcaptions show the sequences of the frame sizes.

Variants 1b and 1c can be implemented in linear time in the size of X by sorting the vertices in buckets according to their degrees. Regarding Variant 1a, while it is NP-hard to find a maximum pair of independent sets in a graph [5], there exists a fixed-parameter tractable (FPT) algorithm parameterized by the treewidth of the graph (Thm. 6). Therefore, Variant 1a can be much slower than Variants 1b and 1c, although it may lead to better solutions. We also remark that Thm. 6 establishes a result of independent interest, which goes beyond the purposes of our specific problem.

A *tree decomposition* [39] of a graph X is a tree $\mathcal{T}$ such that each vertex $\nu \in V(\mathcal{T})$ is associated with a set $V(\nu) \subseteq V(X)$, called the *bag* of ν, such that

(a) $\bigcup_{\nu \in V(\mathcal{T})} V(\nu) = V(X)$,
(b) for each edge $uv \in E(X)$, there is a vertex $\nu \in V(\mathcal{T})$ such that $u, v \in V(\nu)$, and,
(c) for each $v \in V(X)$, the subgraph of $\mathcal{T}$ induced by the vertices $\{\nu \in V(\mathcal{T}) \mid v \in V(\nu)\}$ is connected.

The *width* of a tree decomposition is $\max_{\nu \in V(\mathcal{T})} |V(\nu)| - 1$. The *treewidth* $\mathrm{tw}(X)$ of a graph X is the minimum among the widths of any of its tree decompositions.

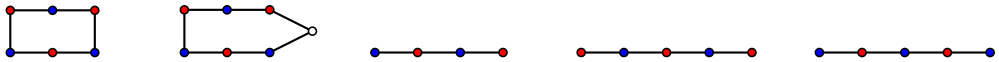
▶ **Theorem 6.** *A Pareto optimal maximum pair of independent sets can be computed in $\mathcal{O}(3^\omega \omega^2 n^3)$ time for a graph X with n vertices and a tree decomposition $\mathcal{T}$ of width ω.*

Proof Sketch: The proof follows the ideas of the FPT approach for constructing a minimum vertex cover as proposed by Niedermeier [33], enhancing it with the concept of Pareto optimal pairs. Root $\mathcal{T}$ at an arbitrary vertex. We use dynamic programming. For a vertex ν of $\mathcal{T}$ let $X(\nu)$ be the subgraph of X induced by the vertices in the bags of the subtree of $\mathcal{T}$ rooted at ν. For each map $c\colon V(\nu) \to \{\text{red, blue, white}\}$, we compute the list $L(\nu, c)$ of Pareto-optimal pairs (α, β) for which $X(\nu)$ contains two disjoint independent sets I_1 and I_2 of size α and β, respectively, such that the red vertices of $V(\nu)$ are contained in I_1, the blue vertices of $V(\nu)$ in I_2, and the white vertices of $V(\nu)$ are neither in I_1 nor I_2. ◀

Given a Pareto-optimal maximum pair (I_1, I_τ) of independent sets, Advanced Greedy 1a does not necessarily result in a planar story with minimum frame size among all planar stories with initial frame I_1 and final frame I_τ. An example is given in Fig. 6: here, there is a vertex that is neither in the initial nor in the final frame whose degree is greater than 3.

▶ **Lemma 7.** *Starting from a Pareto-optimal maximum pair (I_1, I_τ) of independent sets, Advanced Greedy yields a min-frame optimal planar story for a geometric graph G if the crossing graph X contains no cycles and the vertices not in $I_1 \cup I_\tau$ have degree at most three.*

Proof Sketch. The frame sizes are first monotonically increasing and then monotonically decreasing and, thus, at least $\min\{|I_1|, |I_\tau|\}$, which, by Corollary 4, is optimum. ◀



(a) even cycles **(b)** odd cycles **(c)** even paths **(d)** half of odd paths **(e)** other odd paths

Figure 7 The connected components of the crossing graph for a 2-plane graph.

▶ **Theorem 8.** *Advanced Greedy 1a solves* MaxMinFramePlanarStory(G) *optimally*
(a) *in linear time for 2-plane geometric graphs, and*
(b) *in cubic time for those 3-plane geometric graphs for which the crossing graph contains no cycles.*

Proof Sketch. The statement for 3-plane graphs is a corollary of Lemma 7. For 2-plane graphs, the crossing graph has maximum degree two and, thus, its connected components are paths and cycles, and a maximum pair (I_1, I_τ) of independent sets can be computed in linear time. See Fig. 7. Moreover, if we apply Advanced Greedy 1a then the minimum frame size is $\min\{|I_1|, |I_\tau|\}$, unless there is an even cycle but no odd path. In this case, the minimum frame size is $\min\{|I_1|, |I_\tau|\} - 1$. ◀

Alternative variants for Phase 2. In each step of Phase 2, the set E' of admissible future edges with minimum current degree might contain multiple edges. For an edge $e \in E'$, denote by $C(e)$ the set of current edges that cross e. Also, let $E'' \subseteq E'$ be the subset of edges in E' such that, for any $e \in E''$, the number of future edges crossing some edges in $C(e)$ is the maximum over all other edges in E'. We consider two alternative strategies for selecting the next future edge e that enters the drawing: 2a. e is chosen uniformly at random in E'; or 2b. e is chosen uniformly at random in E''.

Variant 2b applies a tie-breaking rule to further restrict the set of admissible edges that can be selected. The rationale is to maximize the amount of future edges that will benefit from the removal of the current neighbors of the edge that will enter the drawing.

6 An ILP Exact Approach

We present an exact integer linear program (ILP) for MaxMinFramePlanarStory(G). Let τ be the number of frames necessary, i.e., $\tau \leq |E(G)|$. For each edge $e \in E(G)$ and each $t \in \{1, \ldots, \tau\}$ we use a binary variable x_e^t to represent that edge e of G is present in frame t. Additionally, we use *lifting variables* $z_e^t \in \{0, 1\}$, where $z_e^t = 1$ indicates that an edge e of G appears in frame t. Finally, we use a variable $y_{\min}$ that corresponds to the minimum number of edges present over all frames.

$$\text{maximize} \quad y_{\min} \tag{1}$$

$$\text{subject to} \quad x_e^t + x_f^t \leq 1 \qquad \forall t \in \{1, \ldots, \tau\}\ \forall \{e, f\} \in E(X) \tag{2}$$

$$\sum_{t=1}^{\tau} x_e^t \geq 1 \qquad \forall e \in E(G) \tag{3}$$

$$\sum_{e \in E(G)} x_e^t \geq y_{\min} \qquad \forall t \in \{1, \ldots, \tau\} \tag{4}$$

$$\sum_{t=1}^{\tau} z_e^t = 1 \qquad \forall e \in E(G) \tag{5}$$

$$z_e^t + x_e^{t-1} \geq x_e^t \qquad \forall e \in E(G)\ \forall t \in \{2, \ldots, \tau\} \tag{6}$$

$$z_e^1 \geq x_e^1 \qquad \forall e \in E(G) \tag{7}$$

$$\sum_{e \in E(G)} z_e^t \leq 1 \qquad \forall t \in \{2, \ldots, \tau\} \tag{8}$$

$$x_e^t, z_e^t \in \{0, 1\} \tag{9}$$

$$y_{\min} \geq 0 \tag{10}$$

Constraints 2 and 3 ensure that no two edges cross in the same frame and that each edge appears in some frames, respectively. Constraints 4 are a linearization of the objective function.

Additionally, we need to model the following property. If an edge e is present in frame F_i and present in frame F_j with $i < j$, then e also needs to be present in all intermediate frames. This can be directly translated, but requires a cubic number of constraints for each edge. Hence, we introduce Constraints 5–7 that model the same property. Consider an edge e that appears at frame t for the first time. Due to Constraints 6 the lifting variable z_e^t must be set to 1 if the edge e appears in frame t, as $x_e^{t-1} = 0$ or since $t = 1$ (Constraints 7). The consecutive frame $t + 1$ can now include this edge, as $x_e^t = 1$ and the variable z_e^{t+1} is no longer required. However, once e disappears in a frame $j > t$, i.e., $x_e^j = 0$, this is no longer possible, as the lifting variables cannot be used anymore due to Constraints 5.

Finally, we include Constraints 8 to enforce that at most one edge appears in each frame, except for the initial frame in which multiple edges may appear.

7 Experimental Analysis

We present the results of an extensive experimental analysis, whose goal is to compare the effectiveness and the efficiency of our heuristics and of the exact algorithm. For the heuristics, we denote by AG-1x2y the variant of algorithm Advanced Greedy that uses Variant 1x for Phase 1 and Variant 2y for Phase 2, where x can be a, b, or c, while y can be a or b. In total we have 6 possible variants of our heuristic. In the following we describe the graph benchmark used for the experiments, the experimental setting, and the obtained results.

Graph Benchmark. To evaluate the performance of our algorithms on different types of graphs, we used several test suites, totaling 637 instances[1]. We considered two main types of instances: Geometric Graphs and Crossing Graphs.

Geometric Graphs. A collection of both random and real graphs drawn with a force-directed algorithm. The different graph classes are as follows

- `random graphs`: 200 geometric graphs: For each $n \in \{10, 20, \ldots, 100\}$ and for each density[2] $d \in \{1.2, 1.6, 2.0, 2.4\}$, we randomly generated 5 distinct graphs with n vertices and $m = d \cdot n$ edges, with Erdős–Rényi's model, which guarantees uniform probability distribution. For each generated graph we computed a straight-line drawing through the popular Fruchterman–Reingold force-directed algorithm [18]. We exploited the NetworkX Python library [22] both for generating the graphs and for computing their drawings. The resulting sizes of the crossing graphs is shown in Fig. 8.

[1] To allow replicability, we made the whole graph benchmark publicly available at `https://github.com/ekatsanou/edge-based-graph-stories`.

[2] The *density* of a graph is the number of its edges divided by the number of its vertices.

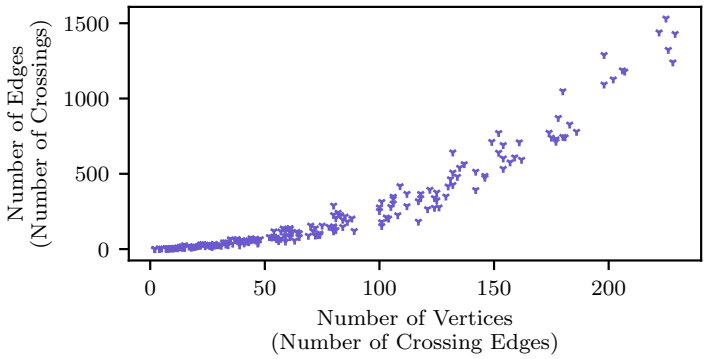


Figure 8 The sizes of the crossing graphs for the `random graphs`. The x-axis is the number of vertices of the crossing graph (i.e., the number of crossing edges of the instance); the y-axis is the number of edges of the crossing graph (i.e., the number of crossings in the instance).

- `real graphs`: 12 graphs with 30 to 379 vertices representing data from various real-world domains (see Table 1).They usually have higher density than the `random graphs` and, consequently, more edge crossings. All these graphs have been taken from the well-known Network Repository [40] (`https://networkrepository.com`). For each graph, we removed self-loops (if any) and computed a straight-line drawing with the implementation of Fruchterman–Reingold's force-directed algorithm in NetworkX.

Table 1 Real graphs with n vertices, m edges, and χ edge-crossings.

Graph name	Category	n	m	χ
insecta-beetle-group-c1-period-1	animal social science network	30	185	1,737
road-chesapeake	road network	39	170	1,049
eco-stmarks	eco network	54	353	6,320
lesmis	miscellaneous network	77	254	838
ca-sandi_auths	collaboration network	86	124	8
gd06-theory	miscellaneous network	101	190	1,015
polbooks	miscellaneous network	105	441	2,465
adjnoun	miscellaneous network	112	425	6,868
rajat11	miscellaneous network	135	680	290
email-enron-only	email network	143	623	5,230
bwm200	miscellaneous	200	596	7
ca-netscience	collaboration network	379	914	901

Crossing Graphs. We generate different types of crossing graphs that are planar. Recall that planar graphs are segment intersection graphs [10, 19]. Therefore, they always reflect the structure of some geometric graphs in terms of edge crossings.

- `caterpillars`: 60 n-vertex caterpillar trees, 10 instances chosen uniformly at random for each value of $n \in \{10, 20, \ldots, 60\}$. First, we selected the diameter δ based on the probability of an n-vertex caterpillar to have such diameter; then we chose how to randomly attach the $n - \delta - 1$ leaves to the $\delta - 1$ internal vertices of the caterpillar [27].

We rejected with probability 0.5 non-symmetric caterpillars, as they would have been generated twice. Due to the large integers involved in the computation, this method allowed us to generate caterpillars with up to 60 vertices.

- `trees`: 80 general trees, with $n \in \{10, 20, \ldots, 80\}$ vertices; for each fixed n we collected 10 distinct instances, generated uniformly at random with the algorithm in [1]
- `series-parallel graphs`: 105 series-parallel graphs, with $n \in \{10, 20, \ldots, 70\}$ vertices and density $d \in \{1.2, 1.4, 1.6\}$. For each pair (n, d) we collected 5 instances, by repeatedly generating biconnected planar graphs using the OGDF library [11] (`https://ogdf.uos.de/`), and discarding those graphs that were not series-parallel.
- `planar graphs`: 180 general connected planar graphs, with $n \in \{10, 20, \ldots, 90\}$ and density $d \in \{1.2, 1.6, 2.0, 2.4\}$. Precisely, we generated 5 instances for each pair (n, d), still using the random generator available in the OGDF library [11].

Since `trees` and `caterpillars` have treewidth 1 and the treewidth of `series-parallel graphs` is 2, these instances are of particular interest for evaluating the performances of the heuristics AG-1a2y, which works in polynomial-time for crossing graphs of bounded treewidth.

Experimental setting. We performed the experiments on a Windows 11 Pro machine with an Intel Core i5-12500 CPU and 16 GB RAM. We used Gurobi [21] version 12.0.1 as ILP solver that used all available cores of the CPU. We implemented the heuristics in Python, using the NetworkX library [22].

On those instances for which the exact algorithm was able to finish the computation within the given time (see below), we compared the optimum with the value computed by the various heuristics. Precisely, we measured the ratio between the minimum frame size of the story computed by each heuristic and the minimum frame size of the story computed by the exact algorithm. With a slight abuse of terminology, we call this value the *approximation ratio*. For the `random graphs` and `real graphs`, the computation of the approximation ratio is restricted to the subset of crossing edges, because crossing-free edges persist in all frames of a planar graph story, regardless of the algorithm used to compute it.

For each instance, we also measured the runtime of each algorithm. We fixed an upper bound of $T = 15$ minutes for the runtime allowed for the exact algorithm, and we recorded the value of the best solution achieved within T (i.e., the incumbent solution). For those instances for which the runtime of the exact algorithm exceeded T, we also measured the optimality gap, that is, the *gap* between the incumbent solution (IS) and the upper bound (UB) estimated by the ILP solver at time T. In formula: $gap = \frac{UB-IS}{UB}$.

Furthermore, after verifying that, for all instances optimally solved by the exact algorithm, the heuristics were also able to produce a solution in less time, we set a maximum time limit of $T' = 10$ minutes for the heuristics based on Variant 1a, whose runtime may grow exponentially in the treewidth of the crossing graph. This was done to allow for faster experimentation, given the large number of instances. The instances for which Variant 1a could not finish within T' were considered unfeasible for AG-1a2a and AG-1a2b.

Performance of the exact algorithm. Fig. 9 shows the percentage of instances per graph class that the ILP was able to solve within the given time limit of $T = 15$ minutes (900 seconds). In each chart, the x-axis is the time in seconds and the y-axis (blue line) reports the number of instances solved optimally within the time at the corresponding x coordinate. Fig. 10 shows the optimality gap across the instances for each graph class (a point for each instance), where the instances were sorted by the number of crossings; points of instances with the same gap coincide in the chart.

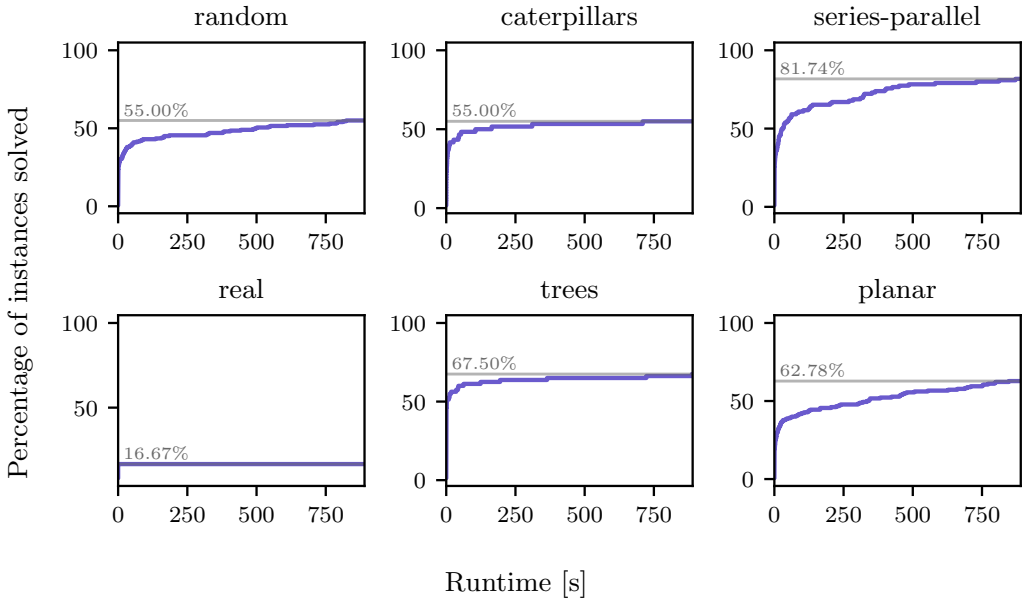


Figure 9 Percentage of instances solved (optimally) by the exact algorithm within time T.

Regarding the Geometric Graphs benchmark, more than half (55%) of the `random graphs` were solved optimally. In particular, all instances with 90 or less crossing edges were solved optimally in the given time, and 90% of the instances with 150 or less crossing edges were solved optimally. Conversely, only the two `real graphs` with small number of crossings (namely, the graphs bmw200 and ca-sandi_auths) could be solved optimally within T. However, for the graph rajat11 (which contains 290 crossings), the optimality gap is close to 0 (namely, 0.08). Overall, for most of the instances in the `random graphs` and `real graphs`, the optimality gap tends to increase with the number of edge crossings.

Regarding the Crossing Graphs benchmark, most of the `series-parallel graphs` (81.74%) and a high percentage of the `trees` (67.5%) and `planar graphs` (62.78%) could be solved optimally. Interestingly, more randomly sampled trees were solved than caterpillars (55.00%) in the given time. However, it is worth noting that the regular pattern observed in the optimality gaps for the `caterpillars` and `trees` is due to the fact that, for instances not solved to optimality within the time limit, the absolute difference between the best bound and the incumbent solution was exactly 1.

Performance of the heuristics. Full details on the runtime of all the heuristics can be found in the long version of the paper [5]. Heuristics AG-1b2a, AG-1b2b, AG-1c2a, and AG-1c2b computed all the instances of our graph benchmark very efficiently. On average they took less than 0.1 seconds on the `random graphs` and about 0.34 seconds on `real graphs` (the most expensive required about 1.6 seconds). Also, they took less than 10 milliseconds for the instances in the Crossing Graphs datasets. Runtime differences among these heuristics are negligible across Variants 2a and 2b for Phase 2, although 2a is slightly faster than 2b. The runtime is mostly affected by the variant chosen for Phase 1, where Variant 1b is a bit faster than Variant 1c, as expected.

Conversely, Variant 1a is significantly more expensive. We implemented it, using the Minimum Fill-in heuristic [8] to compute tree decompositions. For the analysis of the heuristics based on this variant, we took into account only those instances where a Pareto-

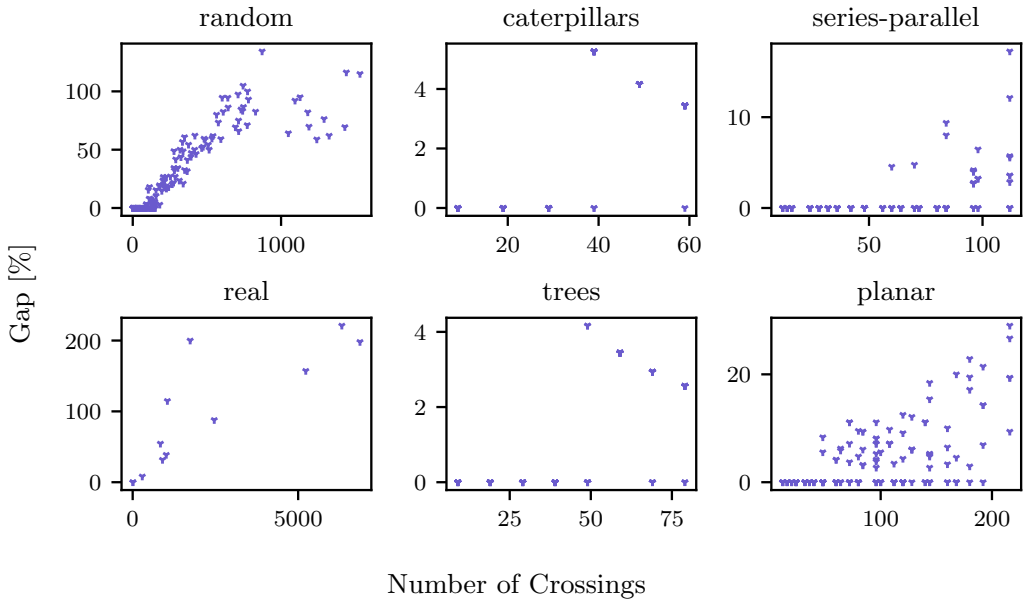


Figure 10 The optimality gap (%) reached by the ILP solver in the given time for each graph class. The instances of each graph class is sorted in ascending order by the number of crossings.

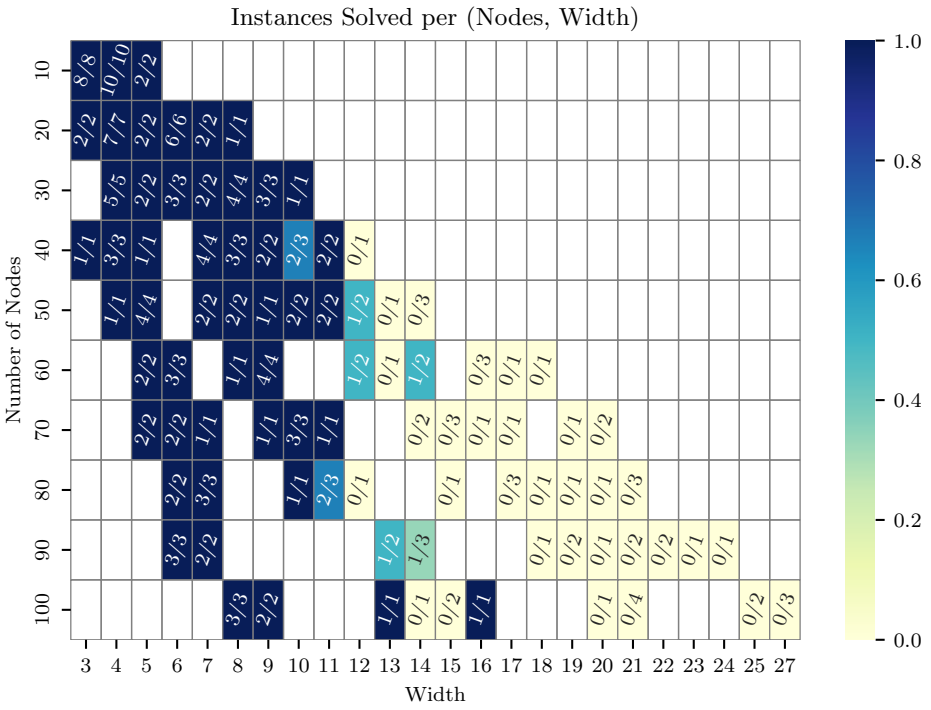


Figure 11 Fraction of the instances in the `random graphs` dataset terminated by Variant `1a` within T', with respect to the number of graph nodes and the width of the tree decomposition.

Table 2 Runtime in seconds for the different heuristics on the `real graphs`. The empty cells refer to the instances that could not be solved by Variant 1a with the given time limit.

graph name	AG-1a2a	AG-1a2b	AG-1b2a	AG-1b2b	AG-1c2a	AG-1c2b
bwm200	0.00	0.00	0.00	0.00	0.00	0.00
ca-sandi_auths	0.00	0.00	0.00	0.00	0.00	0.00
rajat11	0.09	0.13	0.06	0.08	0.10	0.13
lesmis			0.05	0.05	0.08	0.08
ca-netscience			0.61	1.00	1.00	1.00
gd06_theory			0.07	0.07	0.07	0.07
road-chesapeake			0.04	0.04	0.05	0.05
insecta-beetle-group-c1-period-1			0.04	0.04	0.06	0.06
polbooks			0.32	0.32	0.53	0.53
email-enron-only			1.05	1.05	1.64	1.64
eco-stmarks			0.32	0.32	0.40	0.40
adjnoun			0.53	0.53	0.70	0.70

Table 3 The performance of our heuristics in comparison to the exact algorithm.

	All Graphs Combined			
	% Opt	Avg approx. ratio	Min–Max approx. ratio	SD approx. ratio
AG-1a2a	59.9% (243/406)	94.14%	50.00% – 100.00%	9.87%
AG-1a2b	63.1% (256/406)	94.72%	50.00% – 100.00%	9.37%
AG-1b2a	5.7% (23/406)	67.91%	33.33% – 100.00%	13.44%
AG-1b2b	5.4% (22/406)	68.21%	42.86% – 100.00%	13.07%
AG-1c2a	14.3% (58/406)	82.76%	44.44% – 100.00%	10.36%
AG-1c2b	15.3% (62/406)	83.45%	50.00% – 100.00%	10.35%

optimal maximum pair of independent sets was found within the given time $T' = 10$ minutes. Variant 1a was able to find a solution for all the instances in the Crossing Graphs benchmark, as the Minimum Fill-in heuristic produced tree decompositions of width at most six for planar graphs and at most two for series-parallel graphs. For the `random graphs`, Variant 1a solved 137 (of the 200) instances within time T'; the fraction of instances solved for each combination of the two parameters "number of nodes" and "width of the tree decomposition" is illustrated in the heatmap of Fig. 11. For the `real graphs`, Variant 1a solved only 3/12 instances within T' (see Table 2).

Regarding the effectiveness of our heuristics, Table 3 summarizes the comparison over all instances that the exact algorithm managed within the given time $T = 15$ minutes. Detailed results for the different types of graphs are reported in [5].

The results show that AG-1a2a and AG-1a2b achieve a $\sim$ 60% optimality rate and an average approximation ratio of $\sim$ 95%, significantly outperforming the alternatives AG-1b2a/AG-1b2b and AG-1c2a/AG-1c2b in both metrics. The standard deviation of the approximation ratio for AG-1a2a and AG-1a2b remains below 10%, indicating consistent performance. Notably, for the `trees` and the `caterpillars`, AG-1a2a and AG-1a2b achieve near-perfect optimality rates (97.0% and 100.0%, respectively) with approximation ratios close to 100%. In contrast, the heuristics based on Variant 1b achieve fewer than 6% optimal

solutions on average, with lower approximation ratios around 68%. The heuristics using Variant 1c, while also exhibiting low optimality rates ($\sim 15\%$), maintain a stronger average approximation ratio of $\sim 83\%$ and a good standard deviation ($\sim 10.4\%$).

We remark that, for the `real graphs` the exact algorithm managed only 2 instances within the given time T, and AG-1a2a and AG-1a2b managed only 1 additional instance within the given time T'. This data is not enough to get a clear picture of how the other heuristics compare on this dataset. However, by examining the data for all the instances in the `real graphs` dataset, we confirm that also in this case the heuristics based on Variant 1c outperformed those based on Variant 1b, with an improvement of about 29% on average.

Main findings. The exact algorithm typically handles geometric graphs with up to $90 - 100$ crossing edges in few minutes. However, it becomes generally impractical for real-world graphs that are locally dense and inevitably contain many more crossing edges. Regarding the heuristics, the difference between Variants 2a and 2b, in terms of both efficiency and effectiveness, is often negligible. A more detailed analysis of this phenomenon is given in [5]. Conversely, the choice of the variant for Phase 1 makes a big difference, namely:

- The heuristics based on Variant 1a yield solutions close to the optimum in many cases. However, they become computationally unfeasible for graphs with high number of nodes or with crossing graphs of high treewidth; in particular, they exhibit more or less the same limits as the exact algorithm on the real graphs, although they make it possible to manage all instances in the Crossing Graphs benchmark.
- The heuristics based on Variant 1c offer the best trade-off between efficiency and effectiveness. They perform fast on all instances and maintain high average approximation ratios (approximately $80\% - 88\%$, depending on the graph class).

8 Final Remarks and Future Work

We have proposed the MaxMinFramePlanarStory model, where we visualize a graph in a sequence of planar frames such that only one edge at the time enters the story and the minimum frame size is maximized. We suggest the following research directions.

RD1. In addition to ensuring drawing stability and low visual complexity, our model promotes smooth transitions between consecutive frames (i.e., minimal local changes from one frame to the next), as only the current edges that intersect the upcoming edge are removed. However, this approach may result in long stories. To balance these aspects, we can compress a planar story by allowing multiple edges to be inserted simultaneously: for each subsequence of consecutive, pairwise non-crossing edges, all of them can be added at once. In this way, the resulting compressed story remains planar, and its smallest frame is at least as large as that of the original story. Nonetheless, enabling multiple edges to enter simultaneously in the visualization motivates the design of dedicated algorithms aimed at producing short stories while still maximizing the minimum frame size.

RD2. We focused on constructing a planar story of geometric graphs, i.e., of graphs with a given drawing. It would be interesting to investigate how different drawings of the same graph affect the quality of the story. Furthermore, what kind of crossing graphs would allow us to construct good edge stories?

RD3. Our heuristics based on Variants 1b and 1c are fast enough to solve also more complex instances than those in our benchmark. It would be interesting to investigate whether the quality of these variants could be improved by using different heuristics for choosing the independent sets for the initial and final frames than just adding vertices of minimum degree. See, e.g., [20] for a list of heuristics for computing large independent sets.

RD4. Regarding the problem complexity: Is MaxMinFramePlanarStoryD(G,m) NP-complete for 3-plane graphs? Is the problem in FPT parameterized by the treewidth of the crossing graph of G? Is Advanced Greedy 1a a constant-factor approximation algorithm?

References

1 Laurent Alonso, Jean-Luc Rémy, and René Schott. A linear-time algorithm for the generation of trees. *Algorithmica*, 17(2):162–183, 1997. doi:10.1007/BF02522824.

2 Daniel Archambault, Giuseppe Liotta, Martin Nöllenburg, Tommaso Piselli, Alessandra Tappini, and Markus Wallinger. Bundling-aware graph drawing. In Stefan Felsner and Karsten Klein, editors, *32nd International Symposium on Graph Drawing and Network Visualization (GD 2024)*, volume 320 of *LIPIcs*, pages 15:1–15:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.GD.2024.15.

3 Ivan J. Balaban. An optimal algorithm for finding segments intersections. In *Proceedings of the 11th Annual Symposium on Computational Geometry (SoCG'95)*, pages 211–219. Association for Computing Machinery, 1995. doi:10.1145/220279.220302.

4 Fabian Beck, Michael Burch, Stephan Diehl, and Daniel Weiskopf. A taxonomy and survey of dynamic graph visualization. *Comput. Graph. Forum*, 36(1):133–159, 2017. doi:10.1111/CGF.12791.

5 Carla Binucci, Sabine Cornelsen, Walter Didimo, Seok-Hee Hong, Eleni Katsanou, Maurizio Patrignani, Antonios Symvonis, and Samuel Wolf. Planar stories of graph drawings: Algorithms and experiments. Tech. Report arXiv:2508.17532, Cornell University, 2025. doi:10.48550/arXiv.2508.17532.

6 Carla Binucci, Emilio Di Giacomo, William J. Lenhart, Giuseppe Liotta, Fabrizio Montecchiani, Martin Nöllenburg, and Antonios Symvonis. On the complexity of the storyplan problem. In Patrizio Angelini and Reinhard von Hanxleden, editors, *30th International Symposium Graph Drawing and Network Visualization (GD'22)*, volume 13764 of *Lecture Notes in Computer Science*, pages 304–318. Springer, 2022. doi:10.1007/978-3-031-22203-0_22.

7 Carla Binucci, Emilio Di Giacomo, William J. Lenhart, Giuseppe Liotta, Fabrizio Montecchiani, Martin Nöllenburg, and Antonios Symvonis. On the complexity of the storyplan problem. *J. Comput. Syst. Sci.*, 139:103466, 2024. doi:10.1016/J.JCSS.2023.103466.

8 Hans L. Bodlaender. Discovering treewidth. In Peter Vojtáš, Mária Bieliková, Bernadette Charron-Bost, and Ondrej Sýkora, editors, *SOFSEM 2005: Theory and Practice of Computer Science*, pages 1–16, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. doi:10.1007/978-3-540-30577-4_1.

9 Manuel Borrazzo, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, and Maurizio Patrignani. Graph stories in small area. *J. Graph Algorithms Appl.*, 24(3):269–292, 2020. doi:10.7155/JGAA.00530.

10 Jérémie Chalopin and Daniel Gonçalves. Every planar graph is the intersection graph of segments in the plane: extended abstract. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009*, pages 631–638. ACM, 2009. doi:10.1145/1536414.1536500.

11 Markus Chimani, Carsten Gutwenger, Michael Jünger, Gunnar W. Klau, Karsten Klein, and Petra Mutzel. The open graph drawing framework (OGDF). In Roberto Tamassia, editor, *Handbook on Graph Drawing and Visualization*, pages 543–569. Chapman and Hall/CRC, 2013.

12 Giuseppe Di Battista, Walter Didimo, Luca Grilli, Fabrizio Grosso, Giacomo Ortali, Maurizio Patrignani, and Alessandra Tappini. Small point-sets supporting graph stories. In Patrizio Angelini and Reinhard von Hanxleden, editors, *30th International Symposium on Graph Drawing and Network Visualization (GD 2022)*, volume 13764 of *Lecture Notes in Computer Science*, pages 289–303. Springer, 2022. doi:10.1007/978-3-031-22203-0_21.

13 Giuseppe Di Battista, Walter Didimo, Luca Grilli, Fabrizio Grosso, Giacomo Ortali, Maurizio Patrignani, and Alessandra Tappini. Small point-sets supporting graph stories. *J. Graph Algorithms Appl.*, 27(8):651–677, 2023. doi:10.7155/JGAA.00639.

14 Emilio Di Giacomo, Walter Didimo, Giuseppe Liotta, Fabrizio Montecchiani, and Ioannis G. Tollis. Exploring complex drawings via edge stratification. In Stephen K. Wismath and Alexander Wolff, editors, *Graph Drawing - 21st International Symposium, GD 2013, Bordeaux, France, September 23-25, 2013, Revised Selected Papers*, volume 8242 of *Lecture Notes in Computer Science*, pages 304–315. Springer, 2013. doi:10.1007/978-3-319-03841-4_27.

15 Emilio Di Giacomo, Walter Didimo, Giuseppe Liotta, Fabrizio Montecchiani, and Ioannis G. Tollis. Techniques for edge stratification of complex graph drawings. *J. Vis. Lang. Comput.*, 25(4):533–543, 2014. doi:10.1016/J.JVLC.2014.05.001.

16 Peter Eades, Quan Hoang Nguyen, and Seok-Hee Hong. Drawing big graphs using spectral sparsification. In Fabrizio Frati and Kwan-Liu Ma, editors, *25th International Symposium on Graph Drawing and Network Visualization (GD 2017)*, volume 10692 of *Lecture Notes in Computer Science*, pages 272–286. Springer, 2017. doi:10.1007/978-3-319-73915-1_22.

17 Jirí Fiala, Oksana Firman, Giuseppe Liotta, Alexander Wolff, and Johannes Zink. Outerplanar and forest storyplans. In Henning Fernau, Serge Gaspers, and Ralf Klasing, editors, *49th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2024)*, volume 14519 of *Lecture Notes in Computer Science*, pages 211–225. Springer, 2024. doi:10.1007/978-3-031-52113-3_15.

18 Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991. doi:10.1002/spe.4380211102.

19 Daniel Gonçalves, Lucas Isenmann, and Claire Pennarun. Planar graphs as L-intersection or L-contact graphs. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2018)*, pages 172–184. SIAM, 2018. doi:10.1137/1.9781611975031.12.

20 Ernestine Großmann, Sebastian Lamm, Christian Schulz, and Darren Strash. Finding near-optimal weight independent sets at scale. *Journal of Graph Algorithms and Applications*, 28(1):439–473, 2024. doi:10.7155/jgaa.v28i1.2997.

21 Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024. URL: https://www.gurobi.com.

22 Aric A Hagberg, Daniel A Schult, and Pieter J Swart. Exploring network structure, dynamics, and function using networkx. *Proceedings of the 7th Python in Science Conferences (SciPy 2008)*, pages 11–15, 2008.

23 Danny Holten and Jarke J. van Wijk. Force-directed edge bundling for graph visualization. *Comput. Graph. Forum*, 28(3):983–990, 2009. doi:10.1111/J.1467-8659.2009.01450.X.

24 Seok-Hee Hong, Quan Hoang Nguyen, Amyra Meidiana, Jiaxi Li, and Peter Eades. BC tree-based proxy graphs for visualization of big graphs. In *IEEE PacificVis 2018*, pages 11–20, 2018. doi:10.1109/PACIFICVIS.2018.00011.

25 Takehiro Ito, Erik D. Demaine, Nicholas J.A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno. On the complexity of reconfiguration problems. *Theoretical Computer Science*, 412(12):1054–1065, 2011. doi:10.1016/j.tcs.2010.12.005.

26 Takehiro Ito, Marcin Kamiński, Hirotaka Ono, Akira Suzuki, Ryuhei Uehara, and Katsuhisa Yamanaka. Parameterized complexity of independent set reconfiguration problems. *Discrete Applied Mathematics*, 283:336–345, 2020. doi:10.1016/j.dam.2020.01.022.

27 Yosuke Kikuchi, Hiroyuk Tanaka, Shin-ichi Nakano, and Yukio Shibata. How to obtain the complete list of caterpillars. In Tandy Warnow and Binhai Zhu, editors, *Computing and Combinatorics*, pages 329–338, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

28 Stephen G. Kobourov. Force-directed drawing algorithms. In Roberto Tamassia, editor, *Handbook on Graph Drawing and Visualization*, pages 383–408. Chapman and Hall/CRC, 2013.

29 J. Kratochvíl and J. Matousek. Intersection graphs of segments. *Journal of Combinatorial Theory, Series B*, 62(2):289–315, 1994. doi:10.1006/jctb.1994.1071.

30 Bojan Mohar. Face covers and the genus problem for apex graphs. *Journal of Combinatorial Theory, Series B*, 82(1):102–117, 2001. doi:10.1006/jctb.2000.2026.

31 Quan Hoang Nguyen, Seok-Hee Hong, and Peter Eades. TGI-EB: A new framework for edge bundling integrating topology, geometry and importance. In Marc J. van Kreveld and Bettina Speckmann, editors, *19th International Symposium on Graph Drawing (GD 2011)*, volume 7034 of *Lecture Notes in Computer Science*, pages 123–135. Springer, 2011. doi:10.1007/978-3-642-25878-7_13.

32 Quan Hoang Nguyen, Seok-Hee Hong, Peter Eades, and Amyra Meidiana. Proxy graph: Visual quality metrics of big graph sampling. *IEEE Trans. Vis. Comput. Graph.*, 23(6):1600–1611, 2017. doi:10.1109/TVCG.2017.2674999.

33 Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*, volume 31 of *Oxford Lecture Series in Mathematics and its Applications*. Oxford University Press, 2006. doi:10.1093/ACPROF:OSO/9780198566076.001.0001.

34 Helen C. Purchase. Which aesthetic has the greatest effect on human understanding? In Giuseppe Di Battista, editor, *5th International Symposium on Graph Drawing (GD '97)*, volume 1353 of *Lecture Notes in Computer Science*, pages 248–261. Springer, 1997. doi:10.1007/3-540-63938-1_67.

35 Helen C. Purchase. Effective information visualisation: a study of graph drawing aesthetics and algorithms. *Interact. Comput.*, 13(2):147–162, 2000. doi:10.1016/S0953-5438(00)00032-1.

36 Helen C. Purchase, David A. Carrington, and Jo-Anne Allder. Empirical evaluation of aesthetics-based graph layout. *Empir. Softw. Eng.*, 7(3):233–255, 2002.

37 Helen C. Purchase, Christopher Pilcher, and Beryl Plimmer. Graph drawing aesthetics - created by users, not algorithms. *IEEE Trans. Vis. Comput. Graph.*, 18(1):81–92, 2012. doi:10.1109/TVCG.2010.269.

38 D. Rafiei. Effectively visualizing large networks through sampling. In *VIS 05. IEEE Visualization, 2005.*, pages 375–382, 2005. doi:10.1109/VISUAL.2005.1532819.

39 Neil Robertson and Paul D. Seymour. Graph minors. iii. planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984. doi:10.1016/0095-8956(84)90013-3.

40 Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. URL: https://networkrepository.com.

41 Yanhong Wu, Nan Cao, Daniel Archambault, Qiaomu Shen, Huamin Qu, and Weiwei Cui. Evaluation of graph sampling: A visualization perspective. *IEEE Trans. Vis. Comput. Graph.*, 23(1):401–410, 2017. doi:10.1109/TVCG.2016.2598867.

42 Hong Zhou, Panpan Xu, Xiaoru Yuan, and Huamin Qu. Edge bundling in information visualization. *Tsinghua Science and Technology*, 18(2):145–156, 2013. doi:10.1109/TST.2013.6509098.