




Exact Algorithms for Clustered Planarity with Linear Saturators

Giordano Da Lozzo   

Roma Tre University, Italy

Robert Ganian   

Algorithms and Complexity Group, TU Wien, Austria

Siddharth Gupta   

BITS Pilani, K K Birla Goa Campus, India

Bojan Mohar   

Department of Mathematics, Simon Fraser University, Burnaby, Canada

Sebastian Ordyniak   

University of Leeds, UK

Meirav Zehavi   

Ben-Gurion University of the Negev, Beer-Sheva, Israel

Abstract

We study CLUSTERED PLANARITY WITH LINEAR SATURATORS, which is the problem of augmenting an n -vertex planar graph whose vertices are partitioned into independent sets (called *clusters*) with paths – one for each cluster – that connect all the vertices in each cluster while maintaining planarity. We show that the problem can be solved in time $2^{\mathcal{O}(n)}$ for both the variable and fixed embedding case. Moreover, we show that it can be solved in subexponential time $2^{\mathcal{O}(\sqrt{n} \log n)}$ in the fixed embedding case if additionally the input graph is connected. The latter time complexity is tight under the Exponential-Time Hypothesis. We also show that n can be replaced with the vertex cover number of the input graph by providing a linear (resp. polynomial) kernel for the variable-embedding (resp. fixed-embedding) case; these results contrast the NP-hardness of the problem on graphs of bounded treewidth (and even on trees). Finally, we complement known lower bounds for the problem by showing that CLUSTERED PLANARITY WITH LINEAR SATURATORS is NP-hard even when the number of clusters is at most 3, thus excluding the algorithmic use of the number of clusters as a parameter.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability; Theory of computation → Computational geometry; Mathematics of computing → Graph algorithms

Keywords and phrases Clustered planarity, independent c-graphs, path saturation, graph drawing

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2024.24

Related Version *Full Version:* <http://arxiv.org/abs/2409.19410> [39]

Funding *Giordano Da Lozzo:* Supported, in part, by MUR of Italy (PRIN Project no. 2022ME9Z78 – NextGRAAL and PRIN Project no. 2022TS4Y3N – EXPAND).

Robert Ganian: Supported by the Austrian Science Fund (FWF, Project 10.55776/Y1329) and the Vienna Science and Technology Fund (WWTF, Project 10.47379/ICT22029).

Siddharth Gupta: Supported, in part, by BITS Pilani New Faculty Seed Grant.

Bojan Mohar: Supported, in part, by the NSERC Discovery Grant R832714 (Canada), by the ERC Synergy grant (European Union, ERC, KARST, project number 101071836), and by the Research Project N1-0218 of ARIS (Slovenia).

Meirav Zehavi: Supported by the European Research Council (ERC) grant titled PARAPATH.

Acknowledgements This research started at the Dagstuhl Seminar: New Frontiers of Parameterized Complexity in Graph Drawing; seminar number: 23162; April 16-21, 2023 [28].



© Giordano Da Lozzo, Robert Ganian, Siddharth Gupta, Bojan Mohar, Sebastian Ordyniak, and Meirav Zehavi;

licensed under Creative Commons License CC-BY 4.0

35th International Symposium on Algorithms and Computation (ISAAC 2024).

Editors: Julián Mestre and Anthony Wirth; Article No. 24; pp. 24:1–24:16



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

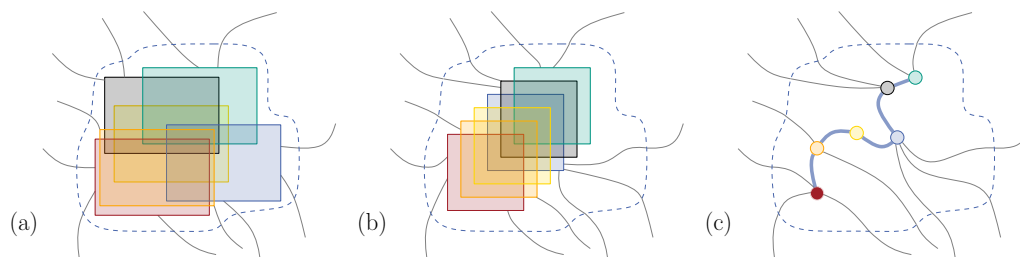
1 Introduction

The representation of graphs with a hierarchical structure has become an increasingly crucial tool in the analysis of networked data across various domains. Indeed, by recursively grouping vertices into clusters exhibiting semantic affinity, modern visualization tools allow for the visualization of massive graphs whose entire visualization would otherwise be impossible. Clustered graphs, where a graph's vertex set is partitioned into distinctive subsets, naturally originate in various and diverse fields such as knowledge representation [35], software visualization [41], visual statistics [10], and data mining [40], only to name a few.

Formally, a *flat clustered graph* (for short, *clustered graph* or *c-graph*) is a pair $\mathcal{C} = (G, \mathcal{V})$ where G is a graph and $\mathcal{V} = \{V_1, \dots, V_k\}$ is a partition of the vertex set of G into sets V_i called *clusters*. The graph G is the *underlying graph* of \mathcal{C} . A pivotal criterion for a coherent visualization of a clustered graph stems from the classical notion of graph planarity. A *c-planar drawing* of a clustered graph $\mathcal{C} = (G, \mathcal{V})$ is defined as a planar drawing of G , accompanied by a representation of each cluster $V_i \in \mathcal{V}$ as a region \mathcal{D}_i homeomorphic to a closed disc, such that regions associated with different clusters are disjoint, the drawing of the subgraph of G induced by the vertices of each cluster V_i lies in the interior of \mathcal{D}_i , and each edge crosses the boundary of a region at most once. The problem of testing for the existence of a c-planar drawing of a clustered graph, called C-PLANARITY TESTING, was introduced by Lengauer [37] (in an entirely different context) and then rediscovered by Feng, Cohen, and Eades [23]. Determining the complexity of the problem has occupied the agenda of the Graph Drawing community for almost three decades [2, 6, 7, 9, 11, 14–16, 18, 19, 21, 26, 29, 30, 32–34, 43]. The seemingly elusive goal of settling the question regarding the computational complexity of this problem has only been recently addressed by Fulek and Tóth [27], who presented a polynomial-time algorithm running in $\mathcal{O}(n^8)$ time (and in $\mathcal{O}(n^{16})$ time for the version of the problem in which a recursive clustering of the vertices is allowed. It is worth pointing out that, even before a polynomial-time solution for the C-PLANARITY problem was presented, Cortese and Patrignani [17] established that the problem retains the same complexity on both flat and general (i.e., recursively clustered) instances. Subsequently, a more efficient algorithm running in quadratic time has been presented by Bläsius, Fink, and Rutter [8].

Patrignani and Cortese studied *independent c-graphs*, i.e., c-graphs where each of the clusters induces an independent set [17]. A characterization given by Di Battista and Frati [21] implies that an independent c-graph is c-planar if and only if the underlying graph can be augmented by adding extra edges, called *saturating edges*, in such a way that the resulting graph has a planar embedding and each cluster induces a tree in the resulting graph. Angelini et al. [4] considered a constrained version of c-planarity, called CLUSTERED PLANARITY WITH LINEAR SATURATORS (for short, CPLS), which takes as input an independent c-graph and requires that each cluster induces a path (instead of a tree) in the augmented graph. They proved that the CPLS problem is NP-complete for c-graphs with an unbounded number of clusters, regardless of whether the input graph is equipped with an embedding or not. The problem fits the paradigm of augmenting planar graphs with edges in such a way that the resulting graph remains planar, while achieving some other desired property, which is a central question in Algorithmic Graph Theory [1, 12, 13, 24, 36, 42].

Although CPLS is a topological problem, it stems from a geometric setting within *intersection-link representations*, a form of hybrid representations for locally-dense globally-sparse graphs [4]. Specifically, see also [3], given a c-graph $\mathcal{C} = (G, \mathcal{V})$ whose every cluster induces a clique, the CLIQUE PLANARITY problem asks to compute a *clique planar representation* of \mathcal{C} , i.e., a drawing of \mathcal{C} in which each vertex $v \in V(G)$ is represented as a



■ **Figure 1** (a) A partial clique planar representation focused on a cluster. (b) Canonical representation of the cluster in (a). A linear saturation of the vertices of the cluster corresponding to (b).

translate $R(v)$ of a given rectangle R , each intra-cluster edge (u, v) is represented by an intersection between $R(u)$ and $R(v)$, and each inter-cluster edge (u, v) is represented by a Jordan arc connecting the boundaries of $R(u)$ and $R(v)$ that intersects neither the interior of any rectangle nor the representation of any other inter-cluster edge. Notably, the authors showed that a c-graph whose every cluster induces a clique admits a clique planar representation if and only if it admits a so-called *canonical representation*, where the vertices are squares arranged in a “linear fashion”; see Fig. 1. This allowed them to establish the equivalence between CPLS and CLIQUE PLANARITY. In particular, they proved that a c-graph \mathcal{C} whose every cluster induces a clique is a yes-instance of CLIQUE PLANARITY if and only if the c-graph obtained by removing all intra-cluster edges from \mathcal{C} is a yes-instance of CPLS.

Our Contribution. In this paper, we study the CPLS problem from a computational perspective, in both the fixed embedding as well as the variable embedding setting. In the fixed embedding case, the underlying graph of the c-graph comes with a prescribed combinatorial embedding, which must be preserved by the output drawing. Instead, in the variable embedding setting, we are allowed to select the embedding of the underlying graph. To distinguish these two settings, we refer to the former problem (i.e., where a fixed embedding is provided as part of the input) as CPLSF. Our main results are as follows.

(1) In Section 3 we give exact single-exponential and sub-exponential algorithms for the problems. In particular, both CPLS and CPLSF can be solved in $2^{\mathcal{O}(n)}$ time. Moreover, we obtain a subexponential $2^{\mathcal{O}(\sqrt{n} \log n)}$ algorithm for CPLSF when the underlying graph is connected; this result is essentially tight under the Exponential Time Hypothesis [31]. In both cases, the main idea behind the algorithms is to use a divide-and-conquer approach that separates the instance according to a hypothetical separator in the solution graph.

(2) In Section 4 we obtain polynomial kernels (and thus establish fixed-parameter tractability) for both CPLS and CPLSF with respect to the vertex cover number of the underlying graph. Interestingly, while being provided with an embedding allowed us to design a more efficient exact algorithm for CPLSF, in the parameterized setting the situation is reversed: we obtain a linear kernel for CPLS, but for CPLSF the size of the kernel is cubic in the vertex cover number. Combining the former result with our exact algorithm for CPLS allows us to obtain an algorithm that runs in single-exponential time with respect to the vertex cover number.

(3) In Section 5 we observe that the CPLS problem is NP-complete on trees and even a disjoint union of stars. Since stars have treedepth, pathwidth, and treewidth one, this charts an intractability border between the vertex cover number parameterization used in Section 4 and other parameters. Then we prove that the problem is NP-complete even for c-graphs having at most 3 clusters, thus strengthening the previously known hardness result for an

unbounded number of clusters. This result combined with the equivalence between CPLS and CLIQUE PLANARITY shows that CLIQUE PLANARITY is NP-complete for instances with a bounded number of clusters, which solves an open problem posed in [3, OP 4.3].

Full proofs and further details for paragraphs marked with (\star) can be found in the full version of the paper [39].

2 Preliminaries

For a positive integer k , we denote by $[k]$ the set $\{1, \dots, k\}$. We use standard terminology in the context of graph theory [22] and graph drawing [20]. An *embedded graph* $G_{\mathcal{E}}$ is a planar graph G equipped with an embedding \mathcal{E} . A *noose* N of an embedded graph $G_{\mathcal{E}}$ is a simple closed curve in some drawing Γ of $G_{\mathcal{E}}$ that

- (i) intersects G only at vertices and
- (ii) traverses each face of Γ at most once.

Given a subgraph H of G , we denote by $\mathcal{E}(H)$ the embedding of H obtained from \mathcal{E} by restricting it to H . The *vertex cover number* of a graph G is the smallest size of a vertex cover in G . We assume basic familiarity with the parameterized complexity framework, and in particular with the notion of *kernelization* [25].

Clustered Planarity with Linear Saturators. Let \mathcal{C} be a c-graph. We say that \mathcal{C} has a *fixed embedding* if the underlying graph of \mathcal{C} is an embedded graph, and has a *variable embedding* otherwise. We say that \mathcal{C} is an *embedded c-graph* if \mathcal{C} has a fixed embedding.

Let $\mathcal{C} = (G, \{V_1, \dots, V_k\})$ be an independent c-graph, i.e., for every $i \in [k]$, V_i is an independent set of G . We say that G can be *linearly saturated* if there exist sets Z_1, \dots, Z_k of non-edges of G such that

- (i) $H = (V(G), E(G) \cup Z)$ for $Z = \bigcup_{i=1}^k Z_i$ is planar,
- (ii) for every $i \in [k]$, each edge in Z_i connects two vertices of V_i in H , and
- (iii) for every $i \in [k]$, the graph $H[V_i]$ is a path.

For $i \in [k]$, the edges in Z_i are the *saturating edges* of cluster V_i , and H is the *linear saturation* of G via Z_1, \dots, Z_k . We now define the CLUSTERED PLANARITY WITH LINEAR SATURATORS (for short, CPLS) and the FIXED EMBEDDING CLUSTERED PLANARITY WITH LINEAR SATURATORS (for short, CPLSF) problems.

- **CPLS:** Given an independent c-graph (G, \mathcal{V}) , does there exist a linear saturation of G ?
- **CPLSF:** Given an independent embedded c-graph $(G_{\mathcal{E}}, \mathcal{V})$, does there exist a linear saturation H of G that admits an embedding \mathcal{E}' for which $\mathcal{E}'(G)$ coincides with \mathcal{E} ?

To devise exact algorithms for the CPLS and CPLSF problem, it will be useful to consider a more general setting. A c-graph is *paths-independent* if each of its clusters induces a collection of paths; the notion of a linear saturator for a paths-independent c-graph is the same as that of an independent c-graph. We now define the CLUSTERED PLANARITY WITH LINEAR SATURATORS COMPLETION (for short, CPLS-Completion) and the FIXED EMBEDDING CLUSTERED PLANARITY WITH LINEAR SATURATORS COMPLETION (for short, CPLSF-Completion) problem as the generalizations of CPLS and CPLSF, respectively, where the input is a paths-independent c-graph. Observe that in case of CPLS (resp., CPLSF), $H[V_i] = H[Z_i]$ as every cluster induces an independent set in G ; this is, however, not necessarily true in the case of CPLS-Completion (resp., CPLSF-Completion).

3 Exact Algorithms for CPLS and CPLSF

This section details our exact single- and sub-exponential algorithms for CPLS and CPLSF.

Proof Ideas. We aim to solve the problem via a divide-and-conquer approach, where at each iteration, we “split” the current instance of the problem into simpler (and, in particular, substantially smaller) sub-instances of the problem. To understand how to perform the split, consider an (unknown) solution Z , and the graph $H = (V(G), E(G) \cup Z)$. As this graph is planar, there exists a noose N that intersects only $\mathcal{O}(\sqrt{|V(G)|})$ vertices of H and does not intersect any edges of H , such that both the interior and the exterior of N contain a constant fraction (roughly between $1/3$ to $2/3$) of the vertices of G [38]. Thus, naturally, we would like to split our problem instance into two instances, one corresponding to the interior and boundary of N , and the other corresponding to the exterior and boundary of N (so, the boundary is common to both). However, two issues arise, which we describe next.

The first (simpler) issue is that we do not know N since we do not know Z . However, since N intersects only few vertices, we can simply “guess” N by guessing the set U of intersected vertices, and the cyclic order ρ in which N intersects them. By guessing, we mean that we iterate over all possible options, and aim to find at least one that yields a solution (if a solution exists). Having U and ρ at hand, we still do not know the interior and exterior of N , and therefore, we still do not know how to create the two simpler sub-instances. Thus, we perform additional guesses: We guess the set I of vertices drawn (with respect to the planar drawing of H) strictly inside N and thereby also the set O of vertices drawn strictly outside N . Additionally, for the set of edges having both endpoints in U , we guess a partition $\{E_{\text{in}}, E_{\text{out}}\}$ that encodes which of them are drawn inside N and which of them are drawn outside N . Specifically for the fixed embedding case, we non-trivially exploit the given embedding to perform the guesses of I, O and $\{E_{\text{in}}, E_{\text{out}}\}$ in a more sophisticated manner that yields only subexponentially many guesses.

The second (more complicated) issue is that we cannot just create two instances: One for the subgraph of G induced by $I \cup U$ (and without the edges in E_{out}) and the other for the subgraph of G induced by $O \cup U$ (and without the edges in E_{in}) and solve them independently. The two main concerns are the following: First, we need a single planar drawing for the entire (unknown) graph H and so the drawings of the two graphs in the two sub-instances should be “compatible”. Second, we may not need to (and in some cases, in fact, must not) add edges to a graph in any of the two sub-instances to connect all vertices in the same cluster in that graph into a single path. Instead, we need to create a collection of paths, so that the two collections that we get, one for each of the two sub-instances, will together yield a single path.

To handle the second concern, we perform additional guesses. Specifically, we guess some information on how the (solution) cluster paths in H “behave” when they are restricted to the interior of N – we guess a triple (M, P, D) which encodes, roughly speaking, a pairing M between some vertices in U that are connected by the cluster paths only using the interior of N , the set of vertices P through which the cluster paths enter the interior of N and “do not return”, and the set of vertices D that are incident to two edges in Z drawn in the interior. Now, having such a guess at hand, we handle both concerns by defining a special graph that augments the graph induced by $G[U \cup O]$ (with the edges in E_{in} removed) so that the solutions returned for the corresponding sub-instance will have to be, in some sense, “compatible” with (M, P, D) as well as draw O and E_{out} outside N while preserving the order ρ . The definition of this augmented graph is, perhaps, the most technical definition required for the proof, as it needs to handle both concerns simultaneously. Among other operations performed to

obtain this augmented graph, for the first concern, we add extra edges between vertices in M , attach pendants on P , and treat vertices in D as if they belong to their own clusters, and for the second concern, we triangulate the result in a careful manner.

The Variable-Embedding Case. Towards solving CPLS, we recursively solve the more general problem mentioned earlier, namely, CPLS-Completion. Additionally, we suppose that some of the vertices of the input graph can be marked, and that we are not allowed to add edges incident to marked vertices. To avoid confusion, we will denote this annotated version by CPLS-COMPLETION*. The rest of this section is devoted to the proof of the following.

► **Theorem 1.** *Let $\mathcal{C} = (G, \mathcal{V})$ be an n -vertex paths-independent c -graph. It can be tested whether \mathcal{C} is a positive instance of CPLS-COMPLETION* in $8^{n+\mathcal{O}(\sqrt{n}\log n)} = 2^{\mathcal{O}(n)}$ time.*

We start with the following definition.

► **Definition 2 (Non-Crossing Matchings and the Partition MatPenDel).** *Let $\mathcal{C} = (G, \mathcal{V})$ be a paths-independent c -graph. Let ρ be a cyclic ordering of some subset U of $V(G)$. A matching M is a non-crossing matching of ρ if it is a matching in the graph $H = (U, \{\{a, b\} : a, b \in U\})$ such that for every pair of edges $\{a, b\}, \{c, d\} \in M$, when we traverse U in the cyclic order ρ , starting with a , we either encounter b before both c and d , or we encounter both c and d before b . Denote by $\text{MatPenDel}(\mathcal{V}, \rho)$ (which stands for Matching, Pendants and Deleted) the set of all triples (M, P, D) such that:*

- M is a non-crossing matching of ρ such that each edge of M matches only vertices in the same cluster, and
- $P, D \subseteq U \setminus V(M)$ are disjoint sets, where $V(M)$ is the set of vertices matched by M .

Intuitively, ρ will represent a cyclic balanced separator of G , i.e., a noose in a drawing of the solution graph that separates the solution graph into two almost equally sized subgraphs, M will represent path segments between pairs of vertices on ρ , P (“pendants”) will represent vertices through which the paths leave ρ never to return,¹ and D will represent degree-2 vertices on the aforementioned path segments (which will be, in a sense, deleted when we “complement” the triple). This will be formalized in Definition 5 ahead.

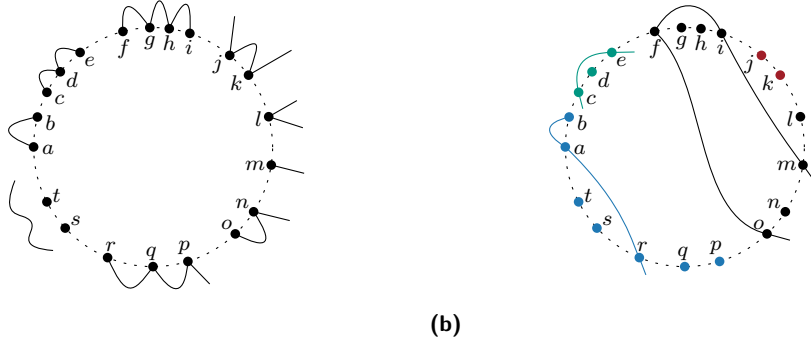
► **Observation 3.** *Let $\mathcal{C} = (G, \mathcal{V})$ be a paths-independent c -graph. Let ρ be a cyclic ordering of some subset U of $V(G)$. Then, $|\text{MatPenDel}(\mathcal{V}, \rho)| = 2^{\mathcal{O}(|U|)}$. Moreover, the set $\text{MatPenDel}(\mathcal{V}, \rho)$ can be computed in time $2^{\mathcal{O}(|U|)}$.*

We formalize the notion of a *partial solution* as follows.

► **Definition 4 (Partial Solution).** *Let $\mathcal{C} = (G, \mathcal{V})$ be a paths-independent c -graph. A cluster path is a path all of whose vertices belong to the same cluster in \mathcal{V} . A partial solution for \mathcal{C} is a set \mathcal{S} of vertex-disjoint cluster paths. Note that \mathcal{S} can contain several cluster paths whose vertices belong to the same cluster. We say that \mathcal{S} is properly marked if every edge in \mathcal{S} incident to a marked vertex also belongs to G .*

Next, we formalize how a triple in MatPenDel captures information on a partial solution (see Fig. 2a). For intuition, think of \mathcal{S} as if it consisted of paths that contain vertices only from the exterior (or only from the interior) of a cyclic separator.

¹ Thus, for a (non-partial) solution, P contains at most 2 vertices per cluster, though we do not need to formally demand this already in Definition 5.



■ **Figure 2** (a) Example for Definition 5: The paths in \mathcal{S} are drawn as black curves, and the vertices in U are marked by disks. We have $M = \{\{a, b\}, \{c, e\}, \{f, i\}\}$, $P = \{m, o, r\}$, $D = \{d, g, h, j, k, l, n, p, q\}$ and $U \setminus (V(M) \cup P \cup D) = \{s, t\}$. (b) Example for Definition 7: The vertices in U are marked by disks, and their association with the clusters is indicated by colors. Suppose $M_{\text{in}} = \{\{a, b\}, \{c, e\}, \{f, i\}\}$, $P_{\text{in}} = \{m, o, r\}$, $D_{\text{in}} = \{d, g, h, j, k, l, n, p, q\}$ and $U \setminus (V(M_{\text{in}}) \cup P_{\text{in}} \cup D_{\text{in}}) = \{s, t\}$, and $M_{\text{out}} = \{\{f, o\}, \{i, m\}, \{r, a\}\}$, $P_{\text{out}} = \{c, e\}$, $D_{\text{out}} = \{s, t\}$ and $U \setminus (V(M_{\text{out}}) \cup P_{\text{out}} \cup D_{\text{out}}) = \{b, d, g, h, j, k, l, n, p, q\}$. Then, the triples $T_{\text{in}} = (M_{\text{in}}, P_{\text{in}}, D_{\text{in}})$ and $T_{\text{out}} = (M_{\text{out}}, P_{\text{out}}, D_{\text{out}})$ are complementary, and $G_{T_{\text{in}}, T_{\text{out}}}$ is a subgraph of the illustrated graph induced by $\{a, b, c, e, f, i, m, o, r\}$. The pendants are the endpoints of the edges going inside or outside the cycle (the vertex b , for example, is not adjacent to a pendant, while the vertex c is).

► **Definition 5 (Extracting a Triple from a Partial Solution).** Let $\mathcal{C} = (G, \mathcal{V})$ be a paths-independent c -graph. Let $U \subseteq V(G)$ and let \mathcal{S} be a partial solution. Then, $\text{ExtractTriple}(U, \mathcal{S}) = (M, P, D)$ is defined as follows:

- M has an edge between the endpoints of every path in \mathcal{S} that has both endpoints in U ,
- $P \subseteq U$ consists of the vertices of degree 1 in \mathcal{S} belonging to U that are not in $V(M)$,² and
- $D \subseteq U$ consists of the vertices of degree 2 in \mathcal{S} belonging to U .

Further, \mathcal{S} is compatible with a cyclic ordering ρ of U if M is a non-crossing matching of ρ .

We have the following immediate observation, connecting Definitions 2 and 5.

► **Observation 6.** Let $\mathcal{C} = (G, \mathcal{V})$ be a paths-independent c -graph. Let ρ be a cyclic ordering of some subset U of $V(G)$. Let \mathcal{S} be a partial solution compatible with ρ . Then, $\text{ExtractTriple}(U, \mathcal{S}) \in \text{MatPenDel}(\mathcal{V}, \rho)$.

We will be interested, in particular, in triples which are complementary (see Fig. 2b), which intuitively means that partial solutions for the inside and the outside described by the two triples can be combined to a solution for the whole graph:

► **Definition 7 (Complementary Triples in MatPenDel).** Let $\mathcal{C} = (G, \mathcal{V})$ be a paths-independent c -graph. Let ρ be a cyclic ordering of some subset U of $V(G)$. Then, $T_{\text{in}} = (M_{\text{in}}, P_{\text{in}}, D_{\text{in}})$, $T_{\text{out}} = (M_{\text{out}}, P_{\text{out}}, D_{\text{out}}) \in \text{MatPenDel}(\mathcal{V}, \rho)$ are complementary if:

1. $D_{\text{in}} \subseteq U \setminus (V(M_{\text{out}}) \cup P_{\text{out}} \cup D_{\text{out}})$, and $D_{\text{out}} \subseteq U \setminus (V(M_{\text{in}}) \cup P_{\text{in}} \cup D_{\text{in}})$.³
2. Let $G_{T_{\text{in}}, T_{\text{out}}}$ denote the graph on vertex set $V(M_{\text{out}}) \cup P_{\text{out}} \cup V(M_{\text{in}}) \cup P_{\text{in}}$ and edge set $M_{\text{in}} \cup M_{\text{out}}$, such that for every vertex in P_{in} , and similarly, for every vertex in P_{out} , we

² In other words, for every path in \mathcal{S} having precisely one endpoint in U , P contains the endpoint in U .

³ The reason why we write \subseteq rather than $=$ is that some vertices in, e.g., $U \setminus (V(M_{\text{out}}) \cup P_{\text{out}} \cup D_{\text{out}})$ can be the endpoints of solution paths. For example, consider the vertex b in Fig. 2b.

add a new vertex attached to it and belonging to the same cluster.⁴ Then, this graph is a collection of paths, such that all vertices in its vertex set that belong to the same cluster in \mathcal{V} also belong to the same (single) path, and all vertices that belong to the same path also belong to the same cluster.

The utility of complementary triples is in the following definition, which specifies when a partial solution \mathcal{S} for the inner part satisfying $\text{ExtractTriple}(U, \mathcal{S}) = T_{\text{out}}$ can be combined with any partial solution \mathcal{S}' for the outer part that satisfies $\text{ExtractTriple}(U, \mathcal{S}') = T_{\text{in}}$.

► **Definition 8 (Partial Solution Compatible with $(T_{\text{in}}, I, E_{\text{in}})$).** Consider (G, \mathcal{V}) , a cyclic ordering ρ of some $U \subseteq V(G)$, $T_{\text{in}} = (M_{\text{in}}, P_{\text{in}}, D_{\text{in}}) \in \text{MatPenDel}(\mathcal{V}, \rho)$, $I \subseteq V(G) \setminus U$, and $E_{\text{in}} \subseteq E(G[U])$. Then, a partial solution \mathcal{S} is compatible with $(T_{\text{in}}, I, E_{\text{in}})$ if:

1. $\text{ExtractTriple}(U, \mathcal{S}) = T_{\text{out}}$ and T_{in} are complementary.
2. Let $G'_{\text{in}} = G[I \cup U] - (E(G[U]) \setminus E_{\text{in}})$ and $G_{\text{in}} = G'_{\text{in}} - D_{\text{in}}$. We have that \mathcal{S} contains all and only the vertices in G_{in} , and all (but not necessarily only)⁵ edges in G_{in} between vertices in the same cluster.
3. There exists a planar drawing φ_{in} of $G'_{\text{in}} \cup E(\mathcal{S})$ with an inner-face whose boundary contains U (with, possibly, other vertices) ordered as by ρ .
4. Each path in \mathcal{S} satisfies one of the following conditions: (a) it consists of all vertices of a cluster in \mathcal{V} that belong to G_{in} , and has no endpoint in U , or (b) it has an endpoint in U .

Towards the statement that will show the utility of compatibility (in Lemma 10 ahead), we need one more definition, which intuitively provides necessary conditions for obtaining a solution for (G, \mathcal{V}) from a partial solution that is compatible with $(T_{\text{in}}, I, E_{\text{in}})$.

► **Definition 9 (Sensibility of $(T_{\text{in}}, I, E_{\text{in}})$).** Consider (G, \mathcal{V}) , a cyclic ordering ρ of some $U \subseteq V(G)$, $T_{\text{in}} = (M_{\text{in}}, P_{\text{in}}, D_{\text{in}}) \in \text{MatPenDel}(\mathcal{V}, \rho)$, $I \subseteq V(G) \setminus U$, and $E_{\text{in}} \subseteq E(G[U])$. Then, $(T_{\text{in}}, I, E_{\text{in}})$ is sensible if:

1. There is no edge $\{u, v\} \in E(G)$ with $v \in I$ and $u \in O$ for $O = V(G) \setminus (I \cup U)$.
2. No vertex in D_{in} is adjacent in $G'_{\text{in}} = G[I \cup U] - (E(G[U]) \setminus E_{\text{in}})$ to a vertex in the same cluster in \mathcal{V} . Additionally, no vertex in $U \setminus (V(M_{\text{in}}) \cup P_{\text{in}} \cup D_{\text{in}})$ is adjacent in $G'_{\text{out}} = G[O \cup U] - E_{\text{in}}$ for $O = V(G) \setminus (I \cup U)$ to a vertex in the same cluster in \mathcal{V} .
3. No cluster in \mathcal{V} has non-empty intersection with both $I \cup (U \setminus D_{\text{in}})$ and $O \cup D_{\text{in}}$ but not with $V(M_{\text{in}}) \cup P_{\text{in}}$.

We show that compatible solutions yield solutions to CPLS-COMPLETION* in MatPenDel .

► **Lemma 10.** Consider (G, \mathcal{V}) , a cyclic ordering ρ of some $U \subseteq V(G)$, $T_{\text{in}} = (M_{\text{in}}, P_{\text{in}}, D_{\text{in}}) \in \text{MatPenDel}(\mathcal{V}, \rho)$, $I \subseteq V(G) \setminus U$, and $E_{\text{in}} \subseteq E(G[U])$. Suppose that $(T_{\text{in}}, I, E_{\text{in}})$ is sensible. Additionally, consider

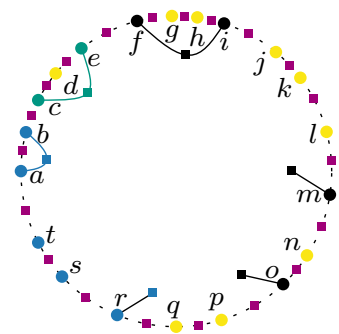
- a properly marked partial solution \mathcal{S}_{in} compatible with $(T_{\text{in}}, I, E_{\text{in}})$, and
- a properly marked partial solution \mathcal{S}_{out} compatible with $\text{ExtractTriple}(U, \mathcal{S}_{\text{in}}) = (T_{\text{out}}, O, E_{\text{out}})$ where $O = V(G) \setminus (I \cup U)$ and $E_{\text{out}} = E(G[U]) \setminus E_{\text{in}}$.

Then, $Z = E(\mathcal{S}_{\text{in}} \cup \mathcal{S}_{\text{out}}) \setminus E(G)$ is a solution to (G, \mathcal{V}) as an instance of CPLS-COMPLETION*.

The crucial tool used to partition an instance of CPLS-COMPLETION* into two smaller instances is the *augmented graph* ($\text{AUGMENTGRAPH}((G, \mathcal{V}), \rho, (M, P, D))$), which is illustrated in Fig. 3 and intuitively ensures that a solution for the given graph is compatible with the triple (T, P, D) . The central statement about augmented graphs is the following.

⁴ So, if a vertex belongs to both P_{in} and P_{out} , we attach two new vertices to it.

⁵ As the paths in \mathcal{S} can contain edges that are not edges in G .



■ **Figure 3** Example of an augmented graph. The vertices in U are marked by disks, and the clusters in \mathcal{V} are $\{\{a, b, p, q, r, s, t\}, \{c, d, e\}, \{f, g, h, i, l, m, n, o\}, \{j, k\}\}$. Suppose $M = \{\{a, b\}, \{c, e\}, \{f, i\}\}$, $P = \{m, o, r\}$, $D = \{d, g, h, j, k, l, n, p, q\}$ and $U \setminus (V(M) \cup P \cup D) = \{s, t\}$. Newly added vertices to the augmented graph are marked by squares, vertices belonging to their own singleton clusters are yellow or purple, and the other clusters are: (i) a, b, r, s, t and the two neighboring squares drawn inside the cycle (blue); (ii) $\{c, e\}$ and the neighboring square drawn inside the cycle (green); (iii) $\{f, i, m, o\}$ and the three neighboring squares drawn inside the cycle (black).

► **Lemma 11.** Consider (G, \mathcal{V}) , a cyclic ordering ρ of some $U \subseteq V(G)$, $T_{\text{in}} = (M_{\text{in}}, P_{\text{in}}, D_{\text{in}}) \in \text{MatPenDel}(\mathcal{V}, \rho)$, $I \subseteq V(G) \setminus U$, and $E_{\text{in}} \subseteq E(G[U])$. Let $Z_{\text{in}} \neq \text{NULL}^6$ be a solution to the instance $(G_A, \mathcal{V}_A) = \text{AUGMENTGRAPH}(G'_{\text{in}}, \rho, T_{\text{in}})$ of CPLS-COMPLETION^* (where $G'_{\text{in}} = G[I \cup U] - (E(G[U]) \setminus E_{\text{in}})$). Then, in polynomial time, we can compute a partial solution compatible with $(T_{\text{in}}, I, E_{\text{in}})$ that is marked properly.

Using the ideas outlined at the beginning of the section, we can now use Lemmas 10 and 11 to show Theorem 1.

Fixed Embedding. By building on the ideas for the variable-embedding case, we also provide a single-exponential algorithm for CPLSF, which becomes subexponential if additionally the input graph is connected. While the single-exponential algorithm is almost identical to our algorithm for CPLS, the subexponential algorithm uses connectivity together with the fixed embedding to reduce the number of guesses during the initial phase of the algorithm.

► **Theorem 12.** CPLSF-COMPLETION (and thus also CPLSF) can be solved in time $2^{\mathcal{O}(n)}$. Moreover, it can be solved in time $2^{\mathcal{O}(\sqrt{n} \log n)}$ if the input graph is connected.

4 The Kernels

In this section we provide kernelization algorithms for CPLS and CPLSF parameterized by the vertex cover number k of the input graph G . Assume that X is a vertex cover of G of size k ; we will deal with computing a suitable vertex cover in the proofs of the main theorems of this section. As our first step, we construct the set Z consisting of the union of X with all vertices of degree at least 3 in G . Since G can be assumed to be planar, we have:

► **Lemma 13** ([25, Lemma 13.3]). $|Z| \leq 3k$.

⁶ We use NULL to algorithmically represent the non-existence of a solution (particularly in pseudocode).

Note that each vertex in $V(G) \setminus Z$ now has 0, 1 or 2 neighbors in Z . For each subset $Q \subseteq Z$ of size at most 2, let the *neighborhood type* T_Q consist of all vertices in $V(G) \setminus Z$ whose neighborhood in Z is precisely Q . Moreover, for $i \in \{0, 1, 2\}$ we let $T_i = \bigcup_{Q \subseteq Z, |Q|=i} T_Q$ contain all vertices outside of Z with degree i . At this point, our approach for dealing with CPLS and CPLSF will diverge.

The Fixed-Embedding Case. We will begin by obtaining a handle on vertices with precisely two neighbors in Z . However, to do so we first need some specialized terminology. To make our arguments easier to present, we assume w.l.o.g. that the input instance \mathcal{I} is equipped with a drawing D of G that respects the given embedding.

Let G_2 be the subgraph of G induced on $Z \cup T_2$, where T_2 is the set of all vertices in $V(G) \setminus Z$ with precisely two neighbors in Z . Let D_2 be the restriction of D to G_2 , and observe that D_2 only differs from D by omitting some pendant and isolated vertices. Let a face in D_2 be *special* if it is incident to more than 2 vertices of Z , and *clean* otherwise; notice that the boundary of a clean face must be a C_4 which has precisely two vertices of Z that lie on opposite sides of the C_4 and which contains only vertices in $T_0 \cup T_1$ in its interior.

► **Lemma 14.** *The number of special faces in D_2 is upper-bounded by $9k$, and the number of vertices in T_2 that are incident to at least one special face is upper-bounded by $36k$.*

For a pair $\{a, b\} \subseteq Z$, we say that a set P of clean faces is an *(ab)-brick* if (1) the only vertices of Z they are incident to are a and b , and (2) P forms a connected region in D_2 , and (3) P is maximal with the above properties. Since the boundaries of every clean face in P consists of a , b , and two degree-2 vertices, this in particular implies that the clean faces in P form a sequence where each pair of consecutive faces shares a single degree-2 vertex; this sequence may either be cyclical (in the case of a and b not being incident to any special faces; we call such bricks *degenerate*), or the first and last clean face in P are adjacent to special faces. Observe that a degenerate brick may only occur if $|Z| = 2$.

► **Observation 15.** *The total number of bricks in D_2 is upper-bounded by $24k$.*

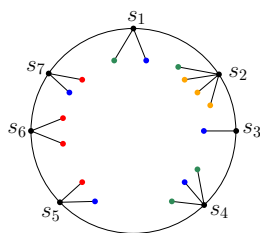
In the next lemma, we use Observation 15 to guarantee the existence of a brick with sufficiently many clean faces to support a safe reduction rule.

► **Lemma 16.** *Assume $|T_2| \geq 420k + 1$, where k is the size of a provided vertex cover of G . Then we can, in polynomial time, either correctly determine that \mathcal{I} is a no-instance or find a vertex $v \in T_2$ with the following property: \mathcal{I} is a yes-instance if and only if so is the instance \mathcal{I}' obtained from \mathcal{I} by removing v .*

Our next goal will be to reduce the size of T_1 . To do so, we will first reduce the total number of clusters occurring in the instance – in particular, while by now we have the tools to reduce the size of G_2 (and hence also the number of clusters intersecting $V(G_2) = T_2 \cup Z$), there may be many other clusters that contain only vertices in T_1 and T_0 . Let V_i be a cluster which does not intersect $V(G_2) = T_2 \cup Z$. Observe that if V_i contains vertices in more than a single face of D_2 , then \mathcal{I} must be a no-instance; for the following, we shall hence assume that this is not the case. In particular, for a cluster V_i such that all of its vertices are contained in a face f of D , we define its *type* $t(i)$ as follows. t_i is the set which contains f as well as all vertices v on the boundary of f fulfilling the following condition: each $v \in t(i)$ is adjacent to a pendant vertex $a \in V_i$ where a is drawn in f . An illustration of types is provided in Fig. 4.

For the next lemma, let τ be the set of all types occurring in \mathcal{I} .

► **Lemma 17.** *If $|V(G_2)| = \alpha$ and D_2 has β faces, then $|\tau| \leq 7\alpha + 8\beta$ or \mathcal{I} is a no-instance.*



■ **Figure 4** An illustration of cluster types in a depicted face f . In this example, individual clusters are marked by colors and none of the vertices s_1, \dots, s_7 belong to any of the colored clusters. The types of the red, blue, yellow and green clusters are $\{f, s_5, s_6, s_7\}$, $\{f, s_1, s_3, s_4, s_5, s_7\}$, $\{f, s_2\}$ and $\{f, s_1, s_2, s_4\}$, respectively. Note that the depicted example cannot occur in a yes-instance since the curves for, e.g., the blue and red clusters would need to cross each other.

Lemma 17 allows us to bound the total number of cluster types in the instance via Lemma 18 below. The proof relies on a careful case analysis that depends on the size of the cluster types of the considered clusters; for cluster types of size at least 4 we directly obtain a contradiction with planarity, but for cluster types of size 2 or 3 we identify “rainbow patterns” that must be present and allow us to simplify the instance.

► **Lemma 18.** *Assume there are three distinct clusters, say V_1, V_2 and V_3 , which do not intersect $V(G_2)$ and all have the same type of size at least 2. Then we can, in polynomial time, either correctly identify that \mathcal{I} is a no-instance or find a non-empty set $A \subseteq T_1$ with the following property: \mathcal{I} is a yes-instance iff so is the instance \mathcal{I}' obtained from \mathcal{I} by removing A .*

Having bounded the number of cluster types (Lemma 17) and the number of clusters of each type (Lemma 18), it remains to bound the number of vertices in each of the clusters.

► **Lemma 19.** *Let $a \in V(G_2)$ and f be a face of D_2 incident to a , and let k be the size of a provided vertex cover of G . Assume a cluster V_i contains at least $2k + 3$ vertices in T_1 that are adjacent to a and lie in f . Then we can, in polynomial time, either correctly identify that \mathcal{I} is a no-instance, or find a vertex $q \in V_i \cap T_1$ such that \mathcal{I} is a yes-instance if and only if so is the instance \mathcal{I}' obtained by deleting q .*

We now have all the ingredients required to obtain our polynomial kernel:

► **Theorem 20.** *CPLSF has a cubic kernel when parameterized by the vertex cover number.*

The Variable-Embedding Case. For CPLS, we establish the following result:

► **Theorem 21** (\star). *CPLS has a linear kernel when parameterized by the vertex cover number.*

Proof Sketch. One can immediately observe that the vertices in T_0 are irrelevant. On a high level, we can then proceed by devising reduction rules that result in a new instance that does not contain large clusters; however it may still happen that there are many clusters occurring in the instance. To deal with this, we group all the clusters together depending on the “neighborhood types” of the vertices they contain. A second level of analysis and reduction then allows us to deal with each group of clusters; there, the most difficult case surprisingly arises when dealing with instances containing many clusters, each consisting of precisely two pendant vertices. ◀

As an immediate consequence of Theorem 21 and Theorem 1, we can obtain an improved asymptotic running time for solving CPLS:

► **Corollary 22.** *CPLS can be solved in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$, where k and n are the vertex cover number and the number of vertices of the input graph, respectively.*

5 NP-completeness

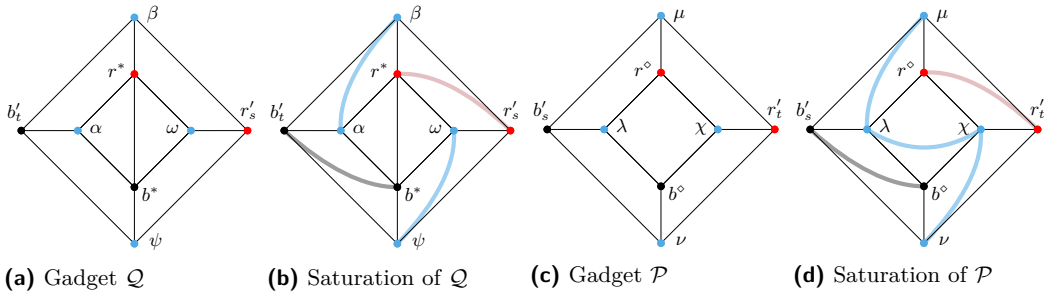
The CPLS problem was shown NP-complete for instances with an *unbounded* number of clusters, even when the underlying graph is a subdivision of a 3-connected planar graph [4]. It is a simple exercise to see that CPLS is NP-complete for trees and for forests of stars (see the Full Version). We establish the NP-completeness of CPLS and CPLSF when restricted to instances with up to three clusters, which we refer to as CPLS-3 and CPLSF-3, respectively.

The proof is based on a reduction from a specialized and still NP-complete version of the BIPARTITE 2-PAGE BOOK EMBEDDING (B2PBE) problem [5]. Let $G = (U_b, U_r, E)$ be a bipartite planar graph, where the vertices in U_b and U_r are called *black* and *red*, respectively. A *bipartite 2-page book embedding* of G is a planar embedding \mathcal{E} of G in which the vertices are placed along a Jordan curve $\ell_{\mathcal{E}}$, called *spine* of \mathcal{E} , the black vertices appear consecutively along $\ell_{\mathcal{E}}$, and each edge lies entirely in one of the two regions of the plane, called *pages*, bounded by $\ell_{\mathcal{E}}$. The B2PBE problem asks whether a given bipartite graph admits a bipartite 2-page book embedding. We will reduce from the B2PBE WITH PRESCRIBED END-VERTICES (B2PBE-PE) problem. Given a bipartite planar graph $G = (U_b, U_r, E)$ and a quadruple $(b_s, b_t, r_s, r_t) \in U_b \times U_b \times U_r \times U_r$, B2PBE-PE asks whether G admits a bipartite 2-page book embedding \mathcal{E} in which the vertices $b_s, b_t, r_s,$ and r_t appear in this counter-clockwise order along the spine of \mathcal{E} . Let G^+ be a planar supergraph of G whose vertex set is $U_b \cup U_r$ and whose edge set is $E \cup E(\sigma)$, where σ is a cycle that traverses all the vertices of U_b (and, thus, also of U_r) consecutively. A cycle σ exhibiting the above properties is a *connector* of G . By [5, Lemma 2.1] and the definition of connector, we have the following.

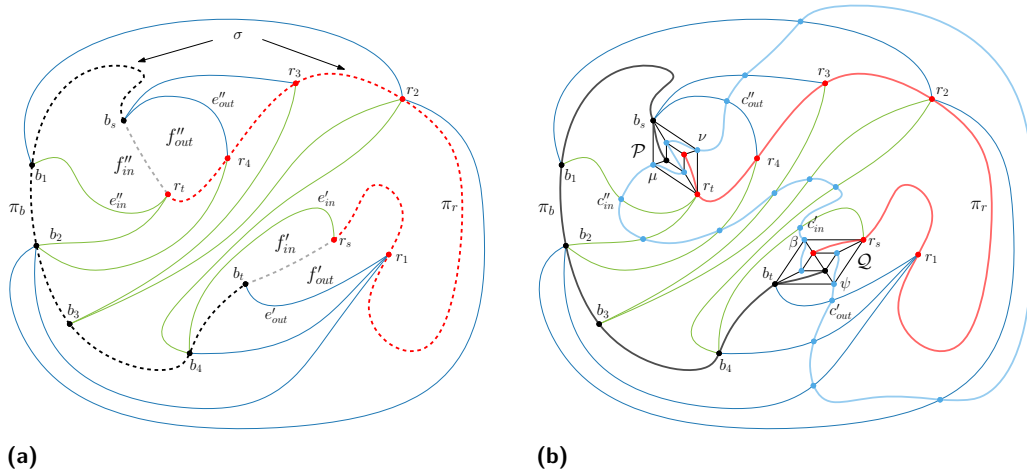
► **Lemma 23.** *A bipartite graph $G = (U_b, U_r, E)$ with distinct vertices $b_s, b_t \in U_b$ and $r_s, r_t \in U_r$ is a positive instance of B2PBE-PE iff it admits a connector σ in which b_t and r_t immediately precede r_s and b_s , respectively, counter-clockwise along σ .*

Next, we sketch a reduction from B2PBE-PE, which is NP-complete even for 2-connected graphs, to CPLS-3. As B2PBE-PE remains NP-complete even for instances with a *fixed embedding* (see [5] for details), the reduction also works if the target problem is CPLSF-3.

► **Lemma 24.** $\text{B2PBE-PE} \leq_m^P \text{CPLS-3}$.



■ **Figure 5** Illustrations for the gadget Q and P .



■ **Figure 6** Illustrations for the proof of Lemma 24. (a) A planar embedding \mathcal{E}_σ of a bipartite graph G together with a connector σ of G . The edges of σ are dashed. (b) A planar embedding of the linear saturation of the underlying graph of the independent c-graph \mathcal{C} , obtained from \mathcal{E}_σ .

Proof Sketch. In our reduction from B2PBE-PE to CPLS-3, we will use the *originating gadget* \mathcal{Q} in Fig. 5a and the *traversing gadget* \mathcal{P} in Fig. 5c. These are independent c-graphs with three clusters: red, blue, and black. Let W be a graph with 4 labeled vertices b_s, b_t, r_b , and r_t . The \mathcal{PQ} -merge of W is the graph obtained by taking the union of $W, G_{\mathcal{Q}}$, and $G_{\mathcal{P}}$, identifying the vertices b_t and r_s of W with the vertices b'_t and r'_s of $G_{\mathcal{Q}}$, respectively, and identifying the vertices b_s and r_t of W with the vertices b'_s and r'_t of $G_{\mathcal{P}}$, respectively.

Given a connected bipartite planar graph $G = (U_b, U_r, E)$ and an ordered quadruple $(b_s, b_t, r_s, r_t) \in U_b \times U_b \times U_r \times U_r$, we construct an independent c-graph $\mathcal{C} = (H, \{V_b, V_r, V_c\})$ with three clusters (red, black, and blue) as follows. First, we initialize the underlying graph H of \mathcal{C} to be the \mathcal{PQ} -merge of G (which is well-defined, since G contains the 4 distinct vertices b_s, b_t, r_s , and r_t). Also, we initialize $V_b = U_b \cup \{b^*, b^\diamond\}$, $V_r = U_r \cup \{r^*, r^\diamond\}$, and $V_c = \{\alpha, \beta, \lambda, \mu, \nu, \chi, \psi, \omega\}$. Second, we subdivide, in H , each edge e of $E(G)$ with a dummy vertex c_e and assign the vertex c_e to V_c .

Clearly, the above reduction can be carried out in polynomial time in the size of G . In the Full Version, we show that \mathcal{C} is a positive instance of CPLS-3 if and only if (G, b_s, b_t, r_s, r_t) is a positive instance of B2PBE-PE. The crux of the proof relies on the property that, in *any* planar embedding of a linear saturation of the underlying graph of \mathcal{C} , the end-vertices of the path saturating the red cluster must be r^* and r^\diamond , the end-vertices of the path saturating the blue cluster must be α and ω , and the end-vertices of the path saturating the black cluster must be b^* and b^\diamond . This allows us to turn an embedding of G together with its connector (Fig. 6a) into a linear saturation of the underlying graph of \mathcal{C} (Fig. 6b), and vice versa. ◀

Lemma 24 and the fact that CPLS and CPLSF clearly lie in NP imply the main result of the section, which is summarized in the following and rules out the existence of FPT algorithms for CPLS and CPLSF parameterized by the number of clusters, unless $P = NP$.

► **Theorem 25.** CPLS and CPLSF are NP-complete even when restricted to instances with at most three clusters.

6 Conclusions

This paper established upper and lower bounds that significantly expand our understanding of the limits of tractability for finding linear saturators in the context of clustered planarity.

We remark that, prior to this research, the problem was not known to be NP-complete for instances with $\mathcal{O}(1)$ clusters. Our NP-hardness result for instances of CPLS with three clusters narrows the complexity gap to its extreme (while also solving the open problem posed in [3, OP 4.3] about the complexity of CLIQUE PLANARITY for instances with a bounded number of clusters), and animates the interest for the remaining two cluster case.

References

- 1 Greg Aloupis, Luis Barba, Paz Carmi, Vida Dujmovic, Fabrizio Frati, and Pat Morin. Compatible connectivity augmentation of planar disconnected graphs. *Discret. Comput. Geom.*, 54(2):459–480, 2015. doi:10.1007/S00454-015-9716-8.
- 2 Patrizio Angelini and Giordano Da Lozzo. Clustered planarity with pipes. *Algorithmica*, 81(6):2484–2526, 2019. doi:10.1007/S00453-018-00541-W.
- 3 Patrizio Angelini and Giordano Da Lozzo. Beyond clustered planar graphs. In Seok-Hee Hong and Takeshi Tokuyama, editors, *Beyond Planar Graphs, Communications of NII Shonan Meetings*, pages 211–235. Springer, 2020. doi:10.1007/978-981-15-6533-5_12.
- 4 Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Ignaz Rutter. Intersection-link representations of graphs. *J. Graph Algorithms Appl.*, 21(4):731–755, 2017. doi:10.7155/JGAA.00437.
- 5 Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, and Maurizio Patrignani. 2-Level quasi-planarity or how caterpillars climb (SPQR-)trees. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2779–2798. SIAM, 2021. doi:10.1137/1.9781611976465.165.
- 6 Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Vincenzo Roselli. Relaxing the constraints of clustered planarity. *Comput. Geom.*, 48(2):42–75, 2015. doi:10.1016/J.COMGEO.2014.08.001.
- 7 Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, and Vincenzo Roselli. The importance of being proper: (in clustered-level planarity and T-level planarity). *Theor. Comput. Sci.*, 571:1–9, 2015. doi:10.1016/J.TCS.2014.12.019.
- 8 Thomas Bläsius, Simon D. Fink, and Ignaz Rutter. Synchronized planarity with applications to constrained planarity problems. *ACM Trans. Algorithms*, 19(4):34:1–34:23, 2023. doi:10.1145/3607474.
- 9 Thomas Bläsius and Ignaz Rutter. A new perspective on clustered planarity as a combinatorial embedding problem. *Theor. Comput. Sci.*, 609:306–315, 2016. doi:10.1016/J.TCS.2015.10.011.
- 10 Ulrik Brandes and Jürgen Lerner. Visual analysis of controversy in user-generated encyclopedias. *Inf. Vis.*, 7(1):34–48, 2008. doi:10.1057/PALGRAVE.IVS.9500171.
- 11 Markus Chimani, Giuseppe Di Battista, Fabrizio Frati, and Karsten Klein. Advances on testing c-planarity of embedded flat clustered graphs. *Int. J. Found. Comput. Sci.*, 30(2):197–230, 2019. doi:10.1142/S0129054119500011.
- 12 Markus Chimani, Carsten Gutwenger, Petra Mutzel, and Christian Wolf. Inserting a vertex into a planar graph. In Claire Mathieu, editor, *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 375–383. SIAM, 2009. doi:10.1137/1.9781611973068.42.
- 13 Markus Chimani and Petr Hliněný. Inserting multiple edges into a planar graph. In Sándor P. Fekete and Anna Lubiw, editors, *32nd International Symposium on Computational Geometry, SoCG 2016, June 14-18, 2016, Boston, MA, USA*, volume 51 of *LIPICs*, pages 30:1–30:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICs.SOCG.2016.30.

- 14 Markus Chimani and Karsten Klein. Shrinking the search space for clustered planarity. In Walter Didimo and Maurizio Patrignani, editors, *Graph Drawing - 20th International Symposium, GD 2012, Redmond, WA, USA, September 19-21, 2012, Revised Selected Papers*, volume 7704 of *Lecture Notes in Computer Science*, pages 90–101. Springer, 2012. doi: 10.1007/978-3-642-36763-2_9.
- 15 Pier Francesco Cortese and Giuseppe Di Battista. Clustered planarity. In Joseph S. B. Mitchell and Günter Rote, editors, *Proceedings of the 21st ACM Symposium on Computational Geometry, Pisa, Italy, June 6-8, 2005*, pages 32–34. ACM, 2005. doi:10.1145/1064092.1064093.
- 16 Pier Francesco Cortese, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Maurizio Pizzonia. C-planarity of c-connected clustered graphs. *J. Graph Algorithms Appl.*, 12(2):225–262, 2008. doi:10.7155/JGAA.00165.
- 17 Pier Francesco Cortese and Maurizio Patrignani. Clustered planarity = flat clustered planarity. In Therese Biedl and Andreas Kerren, editors, *Graph Drawing and Network Visualization - 26th International Symposium, GD 2018, Barcelona, Spain, September 26-28, 2018, Proceedings*, volume 11282 of *Lecture Notes in Computer Science*, pages 23–38. Springer, 2018. doi: 10.1007/978-3-030-04414-5_2.
- 18 Giordano Da Lozzo, David Eppstein, Michael T. Goodrich, and Siddharth Gupta. Subexponential-time and FPT algorithms for embedded flat clustered planarity. In Andreas Brandstädt, Ekkehard Köhler, and Klaus Meer, editors, *Graph-Theoretic Concepts in Computer Science - 44th International Workshop, WG 2018, Cottbus, Germany, June 27-29, 2018, Proceedings*, volume 11159 of *Lecture Notes in Computer Science*, pages 111–124. Springer, 2018. doi:10.1007/978-3-030-00256-5_10.
- 19 Giordano Da Lozzo, David Eppstein, Michael T. Goodrich, and Siddharth Gupta. C-planarity testing of embedded clustered graphs with bounded dual carving-width. *Algorithmica*, 83(8):2471–2502, 2021. doi:10.1007/S00453-021-00839-2.
- 20 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.
- 21 Giuseppe Di Battista and Fabrizio Frati. Efficient c-planarity testing for embedded flat clustered graphs with small faces. *J. Graph Algorithms Appl.*, 13(3):349–378, 2009. doi: 10.7155/JGAA.00191.
- 22 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.
- 23 Qing-Wen Feng, Robert F. Cohen, and Peter Eades. Planarity for clustered graphs. In Paul G. Spirakis, editor, *Algorithms - ESA '95, Third Annual European Symposium, Corfu, Greece, September 25-27, 1995, Proceedings*, volume 979 of *Lecture Notes in Computer Science*, pages 213–226. Springer, 1995. doi:10.1007/3-540-60313-1_145.
- 24 Sergej Fialko and Petra Mutzel. A new approximation algorithm for the planar augmentation problem. In Howard J. Karloff, editor, *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 25-27 January 1998, San Francisco, California, USA*, pages 260–269. ACM/SIAM, 1998. URL: <http://dl.acm.org/citation.cfm?id=314613.314714>.
- 25 F.V. Fomin, D. Lokshtanov, S. Saurabh, and M. Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019. URL: <https://books.google.at/books?id=s1N-DwAAQBAJ>.
- 26 Radoslav Fulek, Jan Kyncl, Igor Malinovic, and Dömötör Pálvölgyi. Clustered planarity testing revisited. *Electron. J. Comb.*, 22(4):4, 2015. doi:10.37236/5002.
- 27 Radoslav Fulek and Csaba D. Tóth. Atomic embeddability, clustered planarity, and thickenability. *J. ACM*, 69(2):13:1–13:34, 2022. doi:10.1145/3502264.
- 28 Robert Ganian, Fabrizio Montecchiani, Martin Nöllenburg, Meirav Zehavi, and Liana Khazaliya. New frontiers of parameterized complexity in graph drawing (Dagstuhl Seminar 23162). *Dagstuhl Reports*, 13(4):58–97, 2023. doi:10.4230/DAGREP.13.4.58.
- 29 Michael T. Goodrich, George S. Lueker, and Jonathan Z. Sun. C-planarity of extrovert clustered graphs. In Patrick Healy and Nikola S. Nikolov, editors, *Graph Drawing, 13th*

- International Symposium, GD 2005, Limerick, Ireland, September 12-14, 2005, Revised Papers*, volume 3843 of *Lecture Notes in Computer Science*, pages 211–222. Springer, 2005. doi:10.1007/11618058_20.
- 30 Carsten Gutwenger, Michael Jünger, Sebastian Leipert, Petra Mutzel, Merijam Percan, and René Weiskircher. Advances in c-planarity testing of clustered graphs. In Stephen G. Kobourov and Michael T. Goodrich, editors, *Graph Drawing, 10th International Symposium, GD 2002, Irvine, CA, USA, August 26-28, 2002, Revised Papers*, volume 2528 of *Lecture Notes in Computer Science*, pages 220–235. Springer, 2002. doi:10.1007/3-540-36151-0_21.
 - 31 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/JCSS.2001.1774.
 - 32 Vít Jelínek, Eva Jelínková, Jan Kratochvíl, and Bernard Lidický. Clustered planarity: Embedded clustered graphs with two-component clusters. In Ioannis G. Tollis and Maurizio Patrignani, editors, *Graph Drawing, 16th International Symposium, GD 2008, Heraklion, Crete, Greece, September 21-24, 2008. Revised Papers*, volume 5417 of *Lecture Notes in Computer Science*, pages 121–132. Springer, 2008. doi:10.1007/978-3-642-00219-9_13.
 - 33 Vít Jelínek, Ondrej Suchý, Marek Tesar, and Tomáš Vyskocil. Clustered planarity: Clusters with few outgoing edges. In Ioannis G. Tollis and Maurizio Patrignani, editors, *Graph Drawing, 16th International Symposium, GD 2008, Heraklion, Crete, Greece, September 21-24, 2008. Revised Papers*, volume 5417 of *Lecture Notes in Computer Science*, pages 102–113. Springer, 2008. doi:10.1007/978-3-642-00219-9_11.
 - 34 Eva Jelínková, Jan Kára, Jan Kratochvíl, Martin Pergel, Ondrej Suchý, and Tomáš Vyskocil. Clustered planarity: Small clusters in cycles and eulerian graphs. *J. Graph Algorithms Appl.*, 13(3):379–422, 2009. doi:10.7155/JGAA.00192.
 - 35 Tomihisa Kamada and Satoru Kawai. A general framework for visualizing abstract objects and relations. *ACM Trans. Graph.*, 10(1):1–39, 1991. doi:10.1145/99902.99903.
 - 36 Goos Kant and Hans L. Bodlaender. Planar graph augmentation problems (extended abstract). In Frank K. H. A. Dehne, Jörg-Rüdiger Sack, and Nicola Santoro, editors, *Algorithms and Data Structures, 2nd Workshop WADS '91, Ottawa, Canada, August 14-16, 1991, Proceedings*, volume 519 of *Lecture Notes in Computer Science*, pages 286–298. Springer, 1991. doi:10.1007/BFB0028270.
 - 37 Thomas Lengauer. Hierarchical planarity testing algorithms. *J. ACM*, 36(3):474–509, 1989. doi:10.1145/65950.65952.
 - 38 Richard J Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.
 - 39 Giordano Da Lozzo, Robert Ganian, Siddharth Gupta, Bojan Mohar, Sebastian Ordyniak, and Meirav Zehavi. Exact algorithms for clustered planarity with linear saturators, 2024. arXiv:2409.19410.
 - 40 Oliver Niggemann. *Visual data mining of graph based data*. PhD thesis, University of Paderborn, Germany, 2001. URL: <http://ubdata.uni-paderborn.de/ediss/17/2001/niggeman/disserta.pdf>.
 - 41 Renato Paiva, Genáina Nunes Rodrigues, Rodrigo Bonifácio, and Marcelo Ladeira. Exploring the combination of software visualization and data clustering in the software architecture recovery process. In Sascha Ossowski, editor, *Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy, April 4-8, 2016*, pages 1309–1314. ACM, 2016. doi:10.1145/2851613.2851765.
 - 42 Johannes A. La Poutré. Alpha-algorithms for incremental planarity testing (preliminary version). In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 706–715. ACM, 1994. doi:10.1145/195058.195439.
 - 43 Jamie Sneddon and C. Paul Bonnington. A note on obstructions to clustered planarity. *Electron. J. Comb.*, 18(1), 2011. doi:10.37236/646.