Algorithmica (2021) 83:2471–2502
https://doi.org/10.1007/s00453-021-00839-2

C-Planarity Testing of Embedded Clustered Graphs with Bounded Dual Carving-Width

Giordano Da Lozzo, et al. ***[full author details at the end of the article]***

Received: 2 January 2020 / Accepted: 21 May 2021 / Published online: 8 June 2021

Abstract

For a *clustered graph*, i.e, a graph whose vertex set is recursively partitioned into clusters, the C-Planarity Testing problem asks whether it is possible to find a planar embedding of the graph and a representation of each cluster as a region homeomorphic to a closed disk such that (1) the subgraph induced by each cluster is drawn in the interior of the corresponding disk, (2) each edge intersects any disk at most once, and (3) the nesting between clusters is reflected by the representation, i.e., child clusters are properly contained in their parent cluster. The computational complexity of this problem, whose study has been central to the theory of graph visualization since its introduction in 1995 [Feng, Cohen, and Eades, *Planarity for clustered graphs*, ESA'95], has only been recently settled [Fulek and Tóth, *Atomic Embeddability, Clustered Planarity, and Thickenability*, to appear at SODA'20]. Before such a breakthrough, the complexity question was still unsolved even when the graph has a prescribed planar embedding, i.e, for *embedded clustered graphs*. We show that the C-Planarity Testing problem admits a single-exponential single-parameter FPT (resp., XP) algorithm for embedded flat (resp., non-flat) clustered graphs, when parameterized by the carving-width of the dual graph of the input. These are the first FPT and XP algorithms for this long-standing open problem with respect to a single notable graph-width parameter. Moreover, the polynomial dependency of our FPT algorithm is smaller than the one of the algorithm by Fulek and Tóth. In particular, our algorithm runs in quadratic time for flat instances of bounded treewidth and bounded face size. To further strengthen the relevance of this result, we show that an algorithm with running time $O(r(n))$ for flat instances whose underlying graph has pathwidth 1 would result in an algorithm with running time $O(r(n))$ for flat instances and with running time $O(r(n^2) + n^2)$ for general, possibly non-flat, instances.

A preliminary version of this paper appeared in *Proceedings of the 14th International Symposium on Parameterized and Exact Computation (IPEC), Munich, Germany, 2019* [35]. Da Lozzo was supported in part by H2020-MSCA-RISE project 734922-"CONNECT" and by MIUR Project "AHeAD" under PRIN 20174LF3T8. Eppstein was supported in part by the US NSF under Grants CCF-1618301 and CCF-1616248. Goodrich was supported in part by the US NSF under Grant 1815073. Gupta was supported in part by the Zuckerman STEM Leadership Program.

Keywords Clustered planarity · Carving-width · Non-crossing partitions · Fixed-parameter tractability

1 Introduction

Many real-world data exhibit an intrinsic hierarchical structure that can be captured in the form of clustered graphs, i.e., graphs equipped with a recursive clustering of their vertices. This graph model has proved very powerful to represent information at different levels of abstraction and drawings of clustered networks appear in a wide variety of application domains, such as software visualization [65], knowledge representation [60], visual statistics [22], and data mining [63]. More formally, a *clustered graph* (for short, *c-graph*) is a pair $\mathcal{C}(G, \mathcal{T})$, where G is the *underlying graph* and $\mathcal{T}$ is the *inclusion tree* of $\mathcal{C}$, i.e., a rooted tree whose leaves are the vertices of G. Each non-leaf node μ of $\mathcal{T}$ corresponds to a cluster containing the subset V_μ of the vertices of G that are the leaves of the subtree of $\mathcal{T}$ rooted at μ.

A natural and well-established criterion for a readable visualization of a c-graph has been derived from the classical notion of graph planarity. A *c-planar drawing* of a c-graph $\mathcal{C}(G, \mathcal{T})$ (see Fig. 7c) is a planar drawing of G together with a representation of each cluster μ in $\mathcal{T}$ as a region $D(\mu)$ homeomorphic to a closed disc such that: (1) for each cluster μ in $\mathcal{T}$, region $D(\mu)$ contains the drawing of the subgraph $G[V_\mu]$ of G induced by V_μ; (2) for every two clusters μ and η in $\mathcal{T}$, it holds $D(\eta) \subseteq D(\mu)$ if and only if η is a descendant of μ in $\mathcal{T}$; (3) each edge crosses the boundary of any cluster disk at most once; and (4) the boundaries of no two cluster disks intersect. A c-graph is *c-planar* if it admits a c-planar drawing.

The C-Planarity Testing problem, introduced by Feng et al. and Eades more than two decades ago [41], asks for the existence of a c-planar drawing of a c-graph. Despite several algorithms having been presented in the literature to construct c-planar drawings of c-planar c-graphs with nice aesthetic features [10, 38, 54, 62], determining the computational complexity of the C-Planarity Testing problem has been one of the most challenging quests in the graph drawing research area; see, e.g., the survey papers [19, 28, 69]. A generalization of the C-Planarity Testing problem is the problem of testing the planarity of hypergraphs, which was introduced by Johnson and Pollak [59]. Given a hypergraph (V, U), where V is a finite set of vertices and U is a finite set of *hyperedges*, i.e., subsets of V, the Hypergraph Planarity problem asks for the existence of a planar graph $P = (V, E)$ such that the graph $P[X]$ is connected, for every $X \in U$. The Hypergraph Planarity problem was shown to be NP-complete [59] and this negative result holds even if U is the union of two partitions [12]. In contrast, there are polynomial-time algorithms to test hypergraph planarity when P is required to belong to specific families of planar graphs [15, 20, 21, 23, 61, 72] or when U is the union of two partitions and, for every $X \in U$, not only $G[X]$ but also $G[V \setminus X]$ is required to be connected [11]. To shed light on the complexity of the C-Planarity Testing problem, several researches have tried to highlight its connections with other notoriously difficult problems in the area [3, 69], as well as to consider relaxations [5, 7, 8, 11, 33, 40, 73] and more constrained versions [2, 4, 6, 9, 30, 42, 43] of the classical notion of c-planarity. Algebraic approaches

have also been considered [44, 53]. Only recently, Fulek and Tóth settled the question by giving a polynomial-time algorithm for a generalization of the C-Planarity Testing problem, called Atomic Embeddability [47].

A cluster μ is *connected* if $G[V_\mu]$ is connected, and it is *disconnected* otherwise. A c-graph is *c-connected* if every cluster is connected. Efficient algorithms for the c-connected case have been known since the early stages of the research on the problem [29, 36, 41]. Afterwards, polynomial-time algorithms have also been conceived for c-graphs satisfying other, weaker, connectivity requirements [27, 49, 52]. A c-graph is *flat* if each leaf-to-root path in $\mathcal{T}$ consists of exactly two edges, that is, the clustering determines a partition of the vertex set; see, e.g.,Fig. 7a. For flat c-graphs polynomial-time algorithms have been presented for several restricted cases [2, 13, 16, 25, 39, 43, 45, 55, 57, 58].

Motivations and contributions. In this paper, we consider the parameterized complexity of the C-Planarity Testing problem for *embedded c-graphs*, i.e., c-graphs with a prescribed combinatorial embedding, having a connected underlying graph; see also [16, 26, 34, 58] for previous work in this direction. The focus on instances with a fixed embedding is a natural restriction as several graph drawing problems are often easier to solve in this setting. In fact, by our result in [34] there exists a linear-time reduction from C-Planarity Testing of embedded c-graphs to C-Planarity Testing of general c-graphs. Notably, such a reduction only works for instances with a connected underlying graph. We remark that all the existing algorithms for C-Planarity Testing of embedded c-graphs only deal with instances having a connected underlying graph. In fact, whether the C-Planarity Testing problem is polynomial-time solvable for embedded c-graphs having a disconnected underlying graph remains an open problem.

In Sect. 2, we show that the C-Planarity Testing problem retains its complexity when restricted to instances of bounded path-width and to connected instances of bounded tree-width. Such a result, which holds even for general, non-embedded, c-graphs, implies that the goal of devising an algorithm parameterized by graph-width parameters that are within a constant factor from tree-width (e.g., branch-width [66]) or that are bounded by path-width (e.g., tree-width, rank-width [64], boolean-width [1], and clique-width [32]) and with a dependency on the input size which improves upon the one in [47] appears to be a significant algorithmic challenge. The estimate on the running time of the algorithm presented in [47] is $O(n^8)$ for flat c-graphs and $O(n^{16})$ for non-flat c-graphs. However, as noted by the authors [46], these estimates depend on the running time of an algorithm presented by Carmesin [24, Section 6] to test whether a simply connected 2-polyhedron embeds in R^3, and the correctness of the claimed running time of this algorithm has not been confirmed yet.

Remarkably, before the results presented in [47], the computational complexity of the problem was still unsolved even for instances with faces of bounded size, and polynomial-time algorithms were known only for "small" faces and in the flat scenario. Namely, Jelinkova et al. [58] presented a quadratic-time algorithm for 3-connected flat c-graphs with faces of size at most 4. Subsequently, Di Battista and Frati [39] presented a linear-time/linear-space algorithm for embedded flat c-graphs with faces of size at most 5.

Motivated by the discussion above and by the results in Sect. 2, we focus our attention on embedded c-graphs $\mathcal{C}(G,\mathcal{T})$ whose underlying graph G has *both* bounded tree-width and bounded face size, i.e., instances such that the *carving-width* of the dual $\delta(G)$ of G is bounded. Gu and Tamaki [51] showed that, for a planar graph, a carving decomposition of optimal width can be computed in $O(n^3)$ time; further, Thilikos, Serna, and Bodlaender [71] gave an FPT algorithm running in $O(\hat{g}(\omega)n)$ time, where $\hat{g}$ is a computable function, to construct a carving decomposition of width ω of a (not-necessarily planar) graph, if any, or to decide that none exists. In Sect. 3, we present an FPT (resp., XP) algorithm based on a dynamic-programming approach on a special carving-decomposition, namely, a bond-carving decomposition, of $\delta(G)$, to solve the problem for embedded flat (resp., non-flat) c-graphs. The key ingredient of our algorithm lies in the ability of maintaining a succinct description of the internal cluster connectivity via non-crossing partitions. We remark that, to the best of the authors' knowledge, these are the first FPT and XP algorithms for the C-Planarity Testing problem, with respect to a *single* graph-width parameter. More formally, we prove the following results.

Theorem 1.1 *C-Planarity Testing can be solved in $O(2^{4\omega+\log\omega}n+n^2)$ time for any n-vertex embedded flat c-graph $\mathcal{C}(G,\mathcal{T})$, where ω is the carving-width of $\delta(G)$, if a carving decomposition of $\delta(G)$ of width ω is provided, and in $O\big((2^{4\omega+\log\omega}+\hat{g}(\omega))n+n^2\big)$ time, otherwise.*

Theorem 1.2 *C-Planarity Testing can be solved in $O(4^{4\omega+\log\omega}h^{2\omega-1}n+n^2)$ time for any n-vertex embedded non-flat c-graph $\mathcal{C}(G,\mathcal{T})$, where ω is the carving-width of $\delta(G)$ and h is the height of $\mathcal{T}$, if a carving decomposition of $\delta(G)$ of width ω is provided, and in $O\big((4^{4\omega+\log\omega}h^{2\omega-1}+\hat{g}(\omega))n+n^2\big)$ time, otherwise.*

Since $h \leq n$, Theorem 1.2 immediately translates into a slice-wise polynomial (XP) algorithm, parameterized by the dual carving width, for C-Planarity Testing of embedded non-flat c-graphs. We formalize this in the following.

Corollary 1.1 *C-Planarity Testing can be solved in $O(4^{4\omega+\log\omega}n^{2\omega})$ time for any n-vertex embedded non-flat c-graph $\mathcal{C}(G,\mathcal{T})$, where ω is the carving-width of $\delta(G)$, if a carving decomposition of $\delta(G)$ of width ω is provided, and in $O\big((4^{4\omega+\log\omega}+\hat{g}(\omega))n^{2\omega}\big)$ time, otherwise.*

It is well known that the carving-width $\mathrm{cw}(\delta(H))$ of the dual graph $\delta(H)$ of a plane graph H with maximum face size $\ell(H)$ and tree-width $\mathrm{tw}(H)$ satisfies the relationship $\mathrm{cw}(\delta(H)) \leq \ell(H)(\mathrm{tw}(H)+2)$ [14, 18]. Therefore, Theorem 1.1 and Corollary 1.1 provide the first[1] polynomial-time algorithms for instances of bounded face size and bounded tree-width. This also answers an open question posed by Di Battista and Frati [39, Open Problem (ii)] for flat instances of bounded tree-width; also,

[1] The results in this paper were accepted for publication in [35] before the contribution by Fulek and Tóth appeared in [47].

since any n-vertex planar graph has tree-width in $O(\sqrt{n})$, it provides an $2^{O(\sqrt{n})}$ sub-exponential-time algorithm for flat instances of bounded face size, which improves the previous $2^{O(\sqrt{n}\log n)}$ time bound presented in [34] for such instances. Finally, in Sect. 4, we exploit Theorems 1.1 and 1.2 to get single-parameter FPT and XP algorithms with respect to the embedded-width and to the dual cut-width of the underlying graph.

2 Definitions and Preliminaries

In this section, we give definitions and preliminaries that will be useful throughout.

Graphs and connectivity. A *graph* $G = (V, E)$ is a pair, where V is the set of *vertices* of G and E is the set of *edges* of G, i.e., pairs of vertices in V. A *multigraph* is a generalization of a graph that allows the existence of multiple copies of the same edge. The *degree* of a vertex is the number of its incident edges. We denote the *maximum degree* of G by $\Delta(G)$. Also, for any $S \subseteq V$, we denote by $G[S]$ the subgraph of G induced by the vertices in S.

A graph is *connected* if it contains a path between any two vertices. A *cutvertex* is a vertex whose removal disconnects the graph. A connected graph is 2-*connected* if either it is an edge or it does not contain any cutvertices. The *blocks* of a graph are its maximal 2-connected subgraphs.

Planar graphs and embeddings. A drawing of a graph is *planar* if it contains no edge crossings. A graph is *planar* if it admits a planar drawing. Two planar drawings of the same graph are *equivalent* if they determine the same *rotation* at each vertex, i.e, the same circular orderings for the edges around each vertex. A *combinatorial embedding* (for short, *embedding*) is an equivalence class of planar drawings. A planar drawing partitions the plane into topologically connected regions, called *faces*. The bounded faces are the *inner faces*, while the unbounded face is the *outer face*. A combinatorial embedding together with a choice for the outer face defines a *planar embedding*. An *embedded graph* (resp. *plane graph*) G is a planar graph with a fixed combinatorial embedding (resp. fixed planar embedding). The *length* of a face f of G is the number of occurrences of the edges of G encountered in a traversal of the boundary of f. The *maximum face size* $\ell(G)$ of G is the maximum length over all faces of G.

C-Planarity. An *embedded c-graph* $\mathcal{C}(G, \mathcal{T})$ is a c-graph whose underlying graph G has a prescribed combinatorial embedding, and it is *c-planar* if it admits a c-planar drawing that preserves the given embedding. Since we only deal with embedded c-graphs, in the remainder of the paper we will refer to them simply as c-graphs, excepts in the statements of theorems and lemmas where we explicitly specify whether the considered c-graphs are embedded or not. Also, when G and $\mathcal{T}$ are clear from the context, we simply denote $\mathcal{C}(G, \mathcal{T})$ as $\mathcal{C}$. Given a non-root cluster $\mu \in \mathcal{T}$, a vertex $v \in V(G)$ is a *vertex of* μ, if $v \in V_\mu$, and an edge $(u, v) \in E(G)$ is an *intra-cluster edge of* μ, if both u and v are in V_μ, and it is an *inter-cluster edge of* μ, otherwise. Note that, given two clusters $\mu, \nu \in \mathcal{T}$ such that ν is an ancestor of μ, the intra-cluster edges of μ are also intra-cluster edges of ν, but the reverse is not necessarily true.

A *candidate saturating edge* of $\mathcal{C}$ is an edge (u, v) not in G such that u and v are incident to the same face of G and there exists a non-root cluster $\mu \in \mathcal{T}$ for which both u and v are in V_μ; refer to Fig. 7b. A c-graph $\mathcal{C}'(G', \mathcal{T}')$ with $\mathcal{T}' = \mathcal{T}$ obtained by adding to $\mathcal{C}$ a subset E^+ of its candidate saturating edges is a *super c-graph* of $\mathcal{C}$; also, if G' is planar, then the set E^+ is a *planar saturation* and the edges in E^+ are the *saturating edges* of $\mathcal{C}$. Furthermore, c-graph $\mathcal{C}$ is *hole-free* if there exists a face f in G such that, when f is chosen as the outer face, there does not exist any cycle whose vertices are in V_μ that encloses in its interior a vertex not in V_μ, for some non-root cluster $\mu \in \mathcal{T}$. Finally, two c-graphs are *equivalent* if they are both c-planar or they are both not c-planar.

Remark 2.1 In this paper, unless stated otherwise, we only consider c-graphs whose underlying graph is connected.

We will exploit the following characterization presented by Di Battista and Frati [39], which holds true also for non-flat c-graphs although originally only proved for flat c-graphs.

Theorem 2.1 ([39], Theorem 1) *An embedded c-graph $\mathcal{C}(G, \mathcal{T})$ is c-planar if and only if:*

(i) *G is planar,*
(ii) *$\mathcal{C}$ is hole-free, and*
(iii) *there exists an embedded super c-graph $\mathcal{C}^*(G^*, \mathcal{T}^*)$ of $\mathcal{C}$ such that G^* is planar and $\mathcal{C}^*$ is c-connected.*

Condition i of Theorem 2.1 can be tested using any of the known linear-time planarity-testing algorithms. Condition ii of Theorem 2.1 can be verified in linear time as described by Di Battista and Frati [39, Lemma 7], by exploiting the linear-time algorithm for checking if an embedded, possibly non-flat, c-graph is hole-free presented by Dahlhaus [36]. Therefore, in the following we will assume that any c-graph satisfies these conditions and thus view the C-Planarity Testing problem as one of testing Condition iii of Theorem 2.1.

Tree-width. A *tree decomposition* of a graph G is a tree T whose nodes, called *bags*, are labeled by subsets of vertices of G. For each vertex v of G the bags containing v must form a nonempty contiguous subtree of T, and for each edge (u, v) of G at least one bag of T must contain both u and v. The *width* of the decomposition is one less than the maximum cardinality of any bag. The *tree-width* $\mathrm{tw}(G)$ of G is the minimum width of any of its tree decompositions.

Cut-sets and duality. Let $G = (V, E)$ be a connected graph and let S be a subset of V. The partition $\{S, V \setminus S\}$ of V is a *cut* of G and the set $E(S, V \setminus S)$ of edges with an endpoint in S and an endpoint in $V \setminus S$ is a *cut-set* of G. Also, the cut-set $E(S, V \setminus S)$ is a *bond* if $G[S]$ and $G[V \setminus S]$ are both non-null and connected.

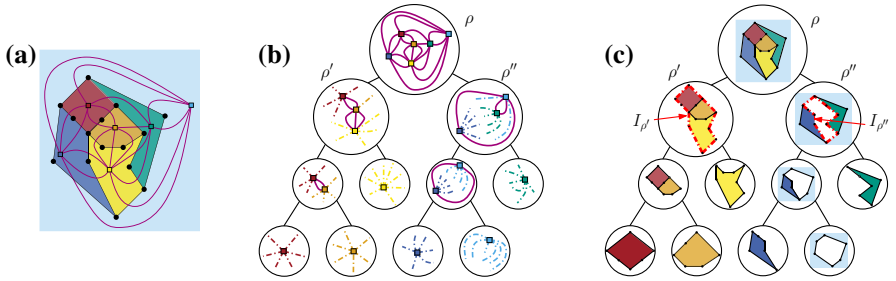

Fig. 1 **a** Running example: an embedded graph G and its dual $\delta(G)$. **b** A bond-carving decomposition (D, γ) of the dual $\delta(G)$ of the graph G in Fig. 1a. **c** The decomposition (D, γ) where the vertices of $\delta(G)$ are replaced by the corresponding faces of G

For an embedded graph G, the *dual* $\delta(G)$ of G is the planar multigraph that has a vertex v_f, for each face f of G, and an edge (v_f, v_g), for each edge e shared by faces f and g; the edge e is the *dual edge* of (v_f, v_g), and vice versa. Also, $\delta(G)$ is 2-connected if and only if G is 2-connected. Figure 1a shows an embedded graph G (black edges) and its dual $\delta(G)$ (purple edges); we will use these graphs as running examples throughout the paper. The following duality is well known.

Lemma 2.1 ([48], Theorem 14.3.1) *If G is an embedded graph, then a set of edges is a cycle of G if and only if their dual edges form a bond in $\delta(G)$.*

Carving-width. A *carving decomposition* of a graph $G = (V, E)$ is a pair (D, γ), where D is a rooted binary tree whose leaves are the vertices of G, and γ is a function that maps the non-root nodes of D, called *bags*, to cut-sets of G as follows. For any non-root bag v, let D_v be the subtree of D rooted at v and let $\mathcal{L}_v$ be the set of leaves of D_v. Then, $\gamma(v) = E(\mathcal{L}_v, V \setminus \mathcal{L}_v)$. The *width* of a carving decomposition (D, γ) is the maximum of $|\gamma(v)|$ over all bags v in D. The *carving-width* $\mathrm{cw}(G)$ of G is the minimum width over all carving decompositions of G. The *dual carving-width* is the carving-width of the dual of G. Note that, the maximum face size $\ell'(G)$ coincides with the maximum degree of $\delta(G)$, which is a lower bound for the $\mathrm{cw}(\delta(G))$. A *bond-carving decomposition* is a special kind of carving decomposition in which each cut-set is a bond of the graph; i.e., in a bond-carving decomposition every cut-set separates the graph into two connected components [67, 70]. In [70], Seymour and Thomas presented a quadratic-time algorithm to convert a carving decomposition into a bond-carving decomposition of the same width.

In this paper, we view a bond-carving decomposition of the vertices of the dual $\delta(G)$ of an embedded graph G as a decomposition of the faces of G; see Fig. 1c. A similar approach was followed in [14]. In particular, due to the duality expressed by Lemma 2.1, the cut-sets $\gamma(v)$ of the bags v of D correspond to cycles that can be used to *recursively partition* the faces of the primal graph, where these cycles are formed by the edges of the primal that are dual to those in each cut-set.

Partitions. Let $\mathcal{Q} = \{q_1, q_2, \ldots, q_n\}$ be a ground set. A *partition* of $\mathcal{Q}$ is a set $\{Q_1, \ldots, Q_k\}$ of non-empty subsets Q_i's of $\mathcal{Q}$, called *parts*, such that $\mathcal{Q} = \bigcup_{i=1}^{k} Q_i$ and $Q_i \cap Q_j = \emptyset$, with $1 \leq i < j \leq k$. Observe that $k \leq |\mathcal{Q}|$.

Let now $\mathcal{S} = (s_1, s_2, \ldots, s_n)$ be a *cyclically-ordered set*, i.e., a set equipped with a circular ordering. Let a, b, and c be three elements of $\mathcal{S}$ such that b appears after a and before c in the circular ordering of $\mathcal{S}$; we write $a \prec_b c$. A partition P of $\mathcal{S}$ is *crossing*, if there exist elements $a, c \in S_i$ and $b, d \in S_j$, with $S_i, S_j \in P$ and $i \neq j$, such that $a \prec_b c$ and $c \prec_d a$; and, it is *non-crossing*, otherwise. We denote the set of all the non-crossing partitions of $\mathcal{S}$ by $\mathcal{NC}(\mathcal{S})$. Note that, $|\mathcal{NC}(\mathcal{S})|$ coincides with the *Catalan number* $\mathcal{CAT}(n)$ of n, which satisfies $\mathcal{CAT}(n) \leq 2^{2n}$ [50, Chapter 19].

2.1 Relationship Between Graph-Width Parameters and Connectivity

In this section, we present reductions that shed light on the effect that the interplay between some notable graph-width parameters and the connectivity of the underlying graph have on the computational complexity of the C-Planarity Testing problem.

We will exploit recent results by Cortese and Patrignani, who proved the following:

(a) Any n-vertex non-flat c-graph $\mathcal{C}(G, \mathcal{T})$ can be transformed into an equivalent $O(n \cdot h)$-vertex flat c-graph in quadratic time [31, Theorem 1], where h is the height of $\mathcal{T}$.
(b) Any n-vertex flat c-graph can be turned into an equivalent $O(n)$-vertex *independent flat c-graph*, i.e., a flat c-graph such that each non-root cluster induces an independent set, in linear time [31, Theorem 2].

We remark that the reductions from [31] preserve the connectivity of the underlying graph.

Theorem 2.2 *Let $\mathcal{C}(G, \mathcal{T})$ be an n-vertex (flat) c-graph and let h be the height of $\mathcal{T}$. In $O(n^2)$ time (in $O(n)$ time), it is possible to construct an $O(n \cdot h)$-vertex ($O(n)$-vertex) independent flat c-graph $\mathcal{C}'(G', \mathcal{T}')$ that is equivalent to $\mathcal{C}$ such that:*

(i) *G' is a collection of stars or*
(ii) *G' is a tree.*

Proof Let $\mathcal{C}(G, \mathcal{T})$ be an n-vertex (flat) c-graph. By the results (a) and (b) above, we can construct an $O(n \cdot h)$-vertex ($O(n)$-vertex) independent flat c-graph $\mathcal{C}^+(G^+, \mathcal{T}^+)$ equivalent to $\mathcal{C}$ in $O(n^2)$ time (in $O(n)$ time). Note that, G^+ only contains inter-cluster edges.

Consider the following transformation defined for a single edge $e = (u, v)$ of G^+, which constructs a new c-graph $\mathcal{C}^1$ starting from $\mathcal{C}^+$. First, subdivide the edge e with two dummy vertices u_e and v_e to create edges (u, u_e), (u_e, v_e), and (v_e, v). Then, delete

the edge (u_e, v_e). Finally, assign u_e and v_e to a new cluster μ_e, and add μ_e as a child of the root of the tree $\mathcal{T}^+$. To construct $\mathcal{C}'$ in case (i), we perform the above transformation for all the edges of G^+. To construct $\mathcal{C}'$ in case (ii), as long as the graph contains a cycle, we perform the above transformation for an edge e of such a cycle. Since the transformation to obtain $\mathcal{C}^1$ from $\mathcal{C}^+$ takes $O(1)$ time, the construction of $\mathcal{C}'$ from $\mathcal{C}^+$ can be done in linear time both in case (i) and (ii), the running time follows. Furthermore, Conditions i, ii of Theorem 2.1 are trivially satisfied by $\mathcal{C}'$ since G' contains no cycles.

To show the correctness of the reduction, it suffices to prove that $\mathcal{C}^1$ is c-planar if and only if $\mathcal{C}^+$ is c-planar. Suppose first that $\mathcal{C}^+$ is c-planar and thus by Theorem 2.1, it has a c-connected super c-graph $\mathcal{C}^+_{con}(G^+_{con}, \mathcal{T}^+_{con})$ with G^+_{con} planar. We now construct a c-connected super c-graph $\mathcal{C}^1_{con}(G^1_{con}, \mathcal{T}^1_{con})$ of $\mathcal{C}^1$ with G^1_{con} planar as follows. First, initialize $G^1_{con} = G^+_{con}$ and $\mathcal{T}^1_{con} = \mathcal{T}^1$. As G^+_{con} is a super graph of G^+, the edge e also belongs to G^+_{con} (and thus to G^1_{con}). Then, subdivide the edge e with the vertices u_e and v_e to create edges (u, u_e), (u_e, v_e), and (v_e, v) in G^1_{con}. Clearly, G^1_{con} is a super graph of G^1; also, since e is an inter-cluster edge, all the clusters of $\mathcal{T}^+_{con}$ are still connected in G^1_{con}. Also, the cluster μ_e is connected in G^1_{con} as the edge (u_e, v_e) is present in G^1_{con}. Therefore, $\mathcal{C}^1_{con}$ is a c-connected super c-graph of $\mathcal{C}^1$ and thus, by Theorem 2.1, $\mathcal{C}^1$ is c-planar.

Suppose now that $\mathcal{C}^1$ is c-planar and thus, by Theorem 2.1, it has a c-connected super c-graph $\mathcal{C}^1_{con}(G^1_{con}, \mathcal{T}^1_{con})$ with G^1_{con} planar. We now construct a c-connected super c-graph $\mathcal{C}^+_{con}(G^+_{con}, \mathcal{T}^+_{con})$ of $\mathcal{C}^+$ with G^+_{con} planar as follows. First, initialize $G^+_{con} = G^1_{con}$ and $\mathcal{T}^+_{con} = \mathcal{T}^+$. As $\mathcal{C}^1_{con}$ is c-connected, the cluster μ_e is also connected, i.e., we have the edge (u_e, v_e) in G^1_{con}. Consider the path (u, u_e, v_e, v) and replace it with the edge (u, v). Clearly, all the clusters of $\mathcal{T}^+_{con}$ are still connected in G^+_{con}, and G^+_{con} is still planar. Therefore, $\mathcal{C}^+_{con}$ is a c-connected super c-graph of $\mathcal{C}^+$ and thus, by Theorem 2.1, $\mathcal{C}^+$ is c-planar. □

We point out that by applying the reduction in the above proof without enforcing a specific embedding, Theorem 2.2 also holds for general instances of the C-Planarity Testing problem, i.e., non-embedded c-graphs. Moreover, since the reduction given in [31] also works for disconnected instances, applying the reduction of Theorem 2.2 for case (i) to a general disconnected instance $\mathcal{C}(G, \mathcal{T})$ would result in an equivalent independent flat c-graph $\mathcal{C}'(G', \mathcal{T}')$ such that G' is a collection of stars. An immediate, yet important, consequence of this discussion is that an algorithm with running time $O(r(n))$ for flat instances whose underlying graph is a collection of stars would result in an algorithm with running time $O(r(n))$ for flat instances and with running time $O(r(n^2) + n^2)$ for general, possibly non-flat, instances, where r is a computable function.

The proof of the following lemma, which will turn useful in the following sections, is based on the duality expressed by Lemma 2.1.

Lemma 2.2 *Given an n-vertex embedded c-graph $\mathcal{C}(G, \mathcal{T})$ and a carving decomposition (D, γ) of $\delta(G)$ of width ω, in $O(n^2)$ time, it is possible to construct an $O(n)$-vertex*

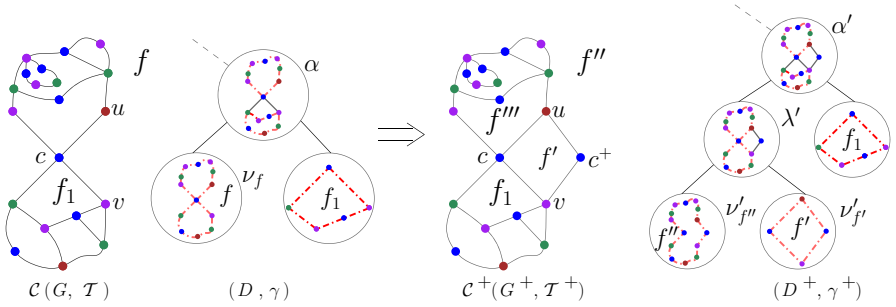

Fig. 2 Reduction of Lemma 2.2 focused on cutvertex c. The transformation of (D, γ) into (D^+, γ^+) is shown. The red dashed edges are dual to those in the cut-set of each bag

embedded c-graph $\mathcal{C}'(G', \mathcal{T}')$ that is equivalent to $\mathcal{C}$ such that G' is 2-connected, and a carving decomposition (D', γ') of $\delta(G')$ of width $\omega' = \max(\omega, 4)$.

Proof Let $B(H)$ and $X(H)$ be the set of blocks and cut-vertices of a graph H respectively. For each cut-vertex $c \in X(H)$, let $i_c(H)$ denotes the number of blocks incident to c and let $i_H = \sum_{c \in X(H)} i_c(H)$. Since G is planar, we have that $i_G \in O(n)$. We construct $\mathcal{C}'(G', \mathcal{T}')$ as follows.

Consider the following operation defined for a cut-vertex $c \in X(G)$. Let μ be the cluster that is the parent of c in $\mathcal{T}$, and let (u, c) and (v, c) be two edges belonging to different blocks of G that are clockwise consecutive in the list of edges incident to c. Let f be the face of G that is to the left of the edge (u, c) when traversing this edge from u to c (and thus also to the left of the edge (v, c) when traversing this edge from c to v). Refer to Fig. 2. Consider the c-graph $\mathcal{C}^+(G^+, \mathcal{T}^+)$ obtained as follows. Initialize $\mathcal{C}^+$ to $\mathcal{C}$. Then, embed a path (u, c^+, v) inside the face f of G^+, where c^+ is a new vertex that we add as a child of μ in $\mathcal{T}^+$; see Fig. 2. We denote by f' the face of G^+ bounded by the cycle (u, c^+, v, c) and by f'' the other face of G^+ incident to c^+. We have that $\mathcal{C}$ and $\mathcal{C}^+$ are equivalent. This is due to the fact that any saturating edge (c, x) incident to c and lying in f (of a c-connected c-planar super c-graph of $\mathcal{C}$) can be replaced by two saturating edges (x, c^+) lying in f'' and (c^+, c) lying in f' (of a c-connected c-planar super c-graph of $\mathcal{C}^+$), and vice versa.

We now show that $\delta(G^+)$ admits a carving decomposition (D^+, γ^+) of width $\omega^+ = \max(\omega, 4)$. In order to do so, we show how to modify the carving decomposition (D, γ) of $\delta(G)$ of width ω into (D^+, γ^+). Consider the leaf bag ν_f of D corresponding to face f and let α be the parent of ν_f in D. We construct D^+ from D as follows. First, we initialize $(D^+, \gamma^+) = (D, \gamma)$; in the following, we denote by ν' the bag of D^+ corresponding to the bag ν of D. We remove ν'_f from D^+, add a new non-leaf bag λ' as a child of α' and two leaf bags $\nu'_{f'}$, corresponding to face f', and $\nu'_{f''}$, corresponding to face f'', as children of λ'; refer to Fig. 2. Let $e_1 = (f', f''')$, $e_2 = (f', f'')$, $e_3 = (f_1, f')$, and $e_4 = (f', f'')$ be the edges of $\delta(G^+)$ that are dual to the edges (u, c), (u, c^+), (v, c), and (v, c^+) of G^+, respectively, where f''' is the face of G^+ incident to (u, c) different from f'. We have $\gamma^+(\nu'_{f'}) = \{e_1, e_2, e_3, e_4\}$,

$\gamma^+(\nu'_{f''}) = (\gamma(\nu'_f) \setminus \{e_1, e_3\}) \cup \{e_2, e_4\}$, $\gamma^+(\lambda') = \gamma(\nu'_f)$, and $\gamma^+(\nu') = \gamma(\nu)$, for any other bag ν belonging to both D^+ and D. In particular, the size of the edge-cuts defined by all the bags different from $\nu'_{f'}$ stays the same, while the size of the edge-cut of $\nu'_{f'}$ is 4. Therefore, (D^+, γ^+) is a carving decomposition of $\delta(G^+)$ of width $\omega^+ = \max(\omega, 4)$.

Let c be a cut-vertex in $X(G)$. We perform the operation described above for each pair of clockwise consecutive edges incident to c in G which belong to different blocks of G. This results in a c-graph $\mathcal{C}_c(G_c, \mathcal{T}_c)$ equivalent to $\mathcal{C}(G, \mathcal{T})$ such that $X(G_c) = X(G) \setminus \{c\}$, $|V(G_c)| = |V(G)| + i_c(G)$, and $i_{G_c} = i_G - i_c(G)$. Moreover, as discussed above, $\delta(G_c)$ admits a carving decomposition (D_c, γ_c) of width $\omega_c = \max(\omega, 4)$. The construction of the c-graph $\mathcal{C}_c$ and of the carving decomposition (D_c, γ_c) can be achieved in linear time by exploiting standard doubly-linked lists with pointers stored with the elements, which support the following two operations in constant time (according to a prescribed orientation of the edges that allows to define when a face leaves an edge to its left/right): (1) Given two clockwise consecutive edges incident to a common vertex, retrieve the face that leaves both such edges to its right/left, and (2) given an edge (u, c) and a face that leaves such an edge to its right/left, retrieve the edge (v, c) that leaves such a face to its left/right. Note that, such data structures can be constructed in linear time from the edge-face incidence graph of G; for implementation details see, e.g., [37].

Repeating the above procedure as long as no cutvertex is left, eventually yields a 2-connected c-graph $\mathcal{C}'(G', \mathcal{T}')$, with $|V(G')| = n + i_G = O(n)$, that is equivalent to $\mathcal{C}$ and a carving decomposition (D', γ') of $\delta(G')$ of width $\omega' = \max(\omega, 4)$. Since, for each cutvertex c, the execution of the above procedure takes $O(|V(G')|)$ time and since the cutvertices and the blocks of graph G_c can be computed in $O(|V(G')|)$ time [56], we have that $\mathcal{C}'$ and (D', γ') can be constructed in $O(n^2)$ time. This concludes the proof. □

3 A Dynamic-Programming Algorithm for C-Planarity Testing

In this section, we present an FPT algorithm for the C-Planarity Testing problem of c-graphs parameterized by the dual carving-width. We first describe a dynamic-programming algorithm to test whether a 2-connected c-graph $\mathcal{C}$ is c-planar, by verifying whether $\mathcal{C}$ satisfies Condition iii of Theorem 2.1. Then, by combining this result and Lemma 2.2, we extend the algorithm to simply-connected instances.

A key ingredient of our algorithm lies in the ability of maintaining a succinct description of the internal cluster connectivity of a subgraph of a given c-graph. This is done by exploiting non-crossing partitions of the vertex set of a c-graph. In the following, we provide definitions and operations that allow us to handle these partitions in an efficient way. Hereafter we first give concepts which are common to both flat and non-flat c-graphs. Then, in Sects. 3.1 and 3.2 we present concepts which are tailored for flat and non-flat c-graphs, respectively. Finally, in Sect. 3.3, we present our algorithm to test whether a 2-connected c-graph $\mathcal{C}$ is c-planar.

Annotated Sets in Clustered Graphs. Let $\mathcal{C}(G, \mathcal{T})$ be a c-graph. A set $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ such that $S_i \subset V(G)$, for every $i \in \{1, 2, \ldots, k\}$, is a *clustered-annotated set* of $\mathcal{C}$ if each element $S_i \in \mathcal{S}$ is *associated with* a non-root cluster $\mu_i \in \mathcal{T}$. Next we define how the association between the elements S_i of a clustered-annotated set and clusters of $\mathcal{T}$ are altered by performing union and intersection operations.

Definition 3.1 Let $\mathcal{S}$ and $\mathcal{S}'$ be two clustered-annotated sets of a c-graph $\mathcal{C}(G, \mathcal{T})$. Let $S \in \mathcal{S}$ and $S' \in \mathcal{S}'$, and let μ and μ' be the clusters associated with S and S', respectively. Also, let Q be a subset of $V(G)$. We have the following:

(a) The cluster $\mu_\cup$ *associated with* $S_\cup = S \cup S'$ is the lowest common ancestor of μ and μ' in $\mathcal{T}$.
(b) The cluster $\mu_\cap$ *associated with* $S_\cap = S \cap Q$ is μ.

Condition a of Definition 3.1 has a simple, yet important, consequence which we formally state below.

Remark 3.1 The intra-cluster edges of both μ and μ' are also the intra-cluster edges of $\mu_\cup$, while the opposite is not necessarily true.

3.1 Flat C-Graphs

Let $\mathcal{C}(G, \mathcal{T})$ be a flat c-graph. A *flat partition* of a set $V' \subseteq V(G)$ is a partition of V' that is also a clustered-annotated set of $\mathcal{C}$. A flat partition $\{S_1, \ldots, S_k\}$ is *good* if all the vertices of S_i are in V_{μ_i} where μ_i is the cluster associated with S_i, for each part S_i. As $\mathcal{C}$ is a flat c-graph, every vertex of G belongs to a unique non-root cluster, therefore the association between parts and non-root clusters in a good flat partition is uniquely defined. Furthermore, a flat partition of a cyclically-ordered set $V' \subseteq V(G)$ is *admissible* if it is both good and non-crossing.

Let P be a good flat partition of a set $\mathcal{Q} \subseteq V(G)$ and let $\mathcal{Q}' \subset \mathcal{Q}$. The *projection* of P *onto* $\mathcal{Q}'$, denoted as $P|_{\mathcal{Q}'}$, is the flat partition of $\mathcal{Q}'$ obtained from P by first replacing each part $S_i \in P$ with $S_i \cap \mathcal{Q}'$ and then removing empty parts, if any. Since P is good and by Condition b of Definition 3.1, we have that each part of $P|_{\mathcal{Q}'}$ is associated with a non-root cluster such that all its vertices belong to that cluster. Therefore, $P|_{\mathcal{Q}'}$ is good.

We define the binary operator $\uplus_F$, called *generalized union*, that given two good flat partitions P' and P'' of sets $\mathcal{Q}' \subseteq V(G)$ and $\mathcal{Q}'' \subseteq V(G)$, respectively, returns a good flat partition $P^* = P' \uplus_F P''$ of $\mathcal{Q}' \cup \mathcal{Q}''$ obtained as follows. Initialize $P^* = P' \cup P''$. Then, as long as there exist $Q_i, Q_j \in P^*$ such that $Q_i \cap Q_j \neq \emptyset$, replace sets Q_i and Q_j with their union $Q_i \cup Q_j$ in P^*. We now argue that P^* is good. First, since P' and P'' are both good, Q_i and Q_j are associated with non-root clusters, say μ and μ'. Second, since $\mathcal{C}$ is a flat c-graph, $Q_i \cap Q_j \neq \emptyset$ implies that $\mu = \mu'$. Finally, by Condition a of Definition 3.1, $Q_i \cup Q_j$ is associated with μ. We have the following.

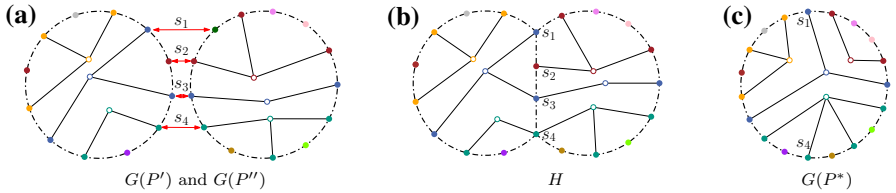

Fig. 3 Illustrations for Definition 3.2

Lemma 3.1 *The generalized union $P^* = P' \uplus_F P''$ of two good flat partitions P' and P'' can be computed in $O(|\mathcal{Q}'| + |\mathcal{Q}''|)$ time.*

Proof We show how to compute P^* in $O(|\mathcal{Q}'| + |\mathcal{Q}''|)$ time. Let $P' = \{Q'_1, \ldots, Q'_{k'}\}$ and $P'' = \{Q''_1, \ldots, Q''_{k''}\}$. We construct an undirected graph G^* as follows. First, for each element $a \in \mathcal{Q}' \cup \mathcal{Q}''$, we add a vertex v_a to $V(G^*)$. Then, for each part $Q'_i \in P'$ (resp. $Q''_i \in P''$), we add a vertex $v_{Q'_i}$ (resp. $v_{Q''_i}$) to $V(G^*)$. Finally, for each element $x \in Q'_i$ (resp. $x \in Q''_i$), we add an edge $(v_x, v_{Q'_i})$ (resp. $(v_x, v_{Q''_i})$) to $E(G^*)$. As $k' \leq |\mathcal{Q}'|$ and $k'' \leq |\mathcal{Q}''|$, $|V(G^*)| \leq |\mathcal{Q}'| + |\mathcal{Q}''| + k' + k'' \leq 2(|\mathcal{Q}'| + |\mathcal{Q}''|)$ and $E(G^*) \leq |\mathcal{Q}'| + |\mathcal{Q}''|$.

Observe that, for every part $Q'_i \in P'$ (resp. $Q''_i \in P''$), the set $V_{Q'_i} = \{v_{Q'_i}\} \cup \{v_x | x \in Q'_i\}$ (resp. $V_{Q''_i} = \{v_{Q''_i}\} \cup \{v_x | x \in Q''_i\}$) induces a star in G^*. Therefore, there exist $Q'_i \in P'$ and $Q''_j \in P''$ such that $Q'_i \cap Q''_j \neq \emptyset$ if and only if $V_{Q'_i \cup Q''_j} = \{v_{Q'_i}\} \cup \{v_{Q''_j}\} \cup \{v_x | x \in Q'_i \cup Q''_j\}$ induces a connected subgraph of G^*. We now perform a breadth-first search (BFS) in G^* to find all the connected components of G^*. Let $C = \{C_1, C_2, \ldots, C_l\}$ be the set of all the connected components. For every $C_i \in C$, we construct a new part Q^*_i by removing from C_i the vertices corresponding to parts in P' and P''. The set $\{Q^*_1, \ldots, Q^*_l\}$ is our required P^*. Since the BFS runs in $O(|V(G^*)| + |E(G^*)|)$ time, the lemma follows. □

An admissible flat partition P of a cyclically-ordered set $\mathcal{S} \subseteq V(G)$ can be naturally associated with a 2-connected plane graph $G(P)$ as follows; see, e.g., Fig. 3c. The outer face of $G(P)$ is a cycle $C(P)$ whose vertices are the elements in $\mathcal{S}$ and the clockwise order in which they appear along $C(P)$ is the same as in $\mathcal{S}$. Also, for each part $S_i \in P$ such that $|S_i| \geq 2$, graph $G(P)$ contains a vertex v_i in the interior of $C(P)$ that is adjacent to all the elements in S_i, i.e., removing all the edges of $C(P)$ yields a collection of stars, whose central vertices are the v_i's, and isolated vertices. To show that $G(P)$ is planarly embedded, we only need to prove that the edges incident to the central vertices of any two different stars do not cross. In particular, let v_1 and v_2 be the central vertices of two different stars and let (v_1, a), (v_1, b), (v_2, c), and (v_2, d) be edges of $G(P)$. Since $a, b \in S_1$ and $c, d \in S_2$ and since P is non-crossing, we have that both c and d are encountered when traversing the cycle C from a to b either clockwise or counterclockwise. Therefore, either v_1 and all its incident edges are embedded in the interior of the cycle composed of the edges (v_2, c), (v_2, d), and the edges of the subpath of the cycle C

encountered while traversing C from c to d either clockwise or counterclockwise. We say that $G(P)$ is the *cycle-star* associated with P.

We also extend the definitions of *generalized union* and *projection* to admissible flat partitions by regarding the corresponding cyclically-ordered sets as unordered. We remark that the projection of an admissible flat partition always yields an admissible flat partition. On the other hand, the generalized union of two admissible flat partitions may produce a flat partition which is good but crossing. However, in the following we define an operator, central to our algorithm, which given two admissible flat partitions (with some additional properties) returns an admissible flat partition.

Definition 3.2 Let P' and P'' be two admissible flat partitions of cyclically-ordered sets $\mathcal{S}' \subseteq V(G)$ and $\mathcal{S}'' \subseteq V(G)$, respectively, with the following properties (where $\mathcal{S}_\cap = \{s_1, s_2, \ldots, s_k\}$ denotes the set of elements that are common to $\mathcal{S}'$ and $\mathcal{S}''$): (i) $|\mathcal{S}_\cap| \geq 2$ and $(\mathcal{S}' \cup \mathcal{S}'') \setminus \mathcal{S}_\cap \neq \emptyset$, (ii) the elements of $\mathcal{S}_\cap$ appear consecutively both in $\mathcal{S}'$ and $\mathcal{S}''$, and (iii) the cyclic ordering of the elements in $\mathcal{S}_\cap$ determined by $\mathcal{S}'$ is the reverse of the cyclic ordering of these elements determined by $\mathcal{S}''$.

The binary operator$\textcircled{\cap}$, called *bubble merge*, returns an admissible flat partition $\textcircled{\cap}$ obtained as follows; refer to Fig. 3. Consider the cycle-stars $G(P')$ and $G(P'')$ associated with P' and P'', respectively.

- First, identify the vertices corresponding to the same element of $\mathcal{S}_\cap$ in both $C(P')$ and $C(P'')$ (see Fig. 3a) in such a way that no vertex of $C(P')$ lies inside $C(P'')$, and vice-versa. Since the elements in $\mathcal{S}_\cap$ are consecutive along both $C(P')$ and $C(P'')$, by item (ii), and since the ordering of such elements along $C(P')$ is the reverse of their ordering along $C(P'')$, by item (iii), this yields a new plane graph H (see Fig. 3b). Note that H is 2-connected since $G(P')$ and $G(P'')$ are 2-connected and since $|\mathcal{S}_\cap| \geq 2$; therefore, the outer face f_H of H is a simple cycle.
- Second, traverse f_H clockwise to construct a cyclically-ordered set $\mathcal{S}^* \subseteq \mathcal{S}' \cup \mathcal{S}''$ on the vertices of f_H.
- Finally, set $P^* = (P' \uplus_F P'')|_{\mathcal{S}^*}$. P^* is good by the definition of generalized union and projection. The fact that P^* is a non-crossing partition of $\mathcal{S}^*$ follows immediately from the planarity of H (see Fig. 3c).

Lemma 3.1 and the fact that the graph H in Definition 3.2 can be easily constructed from P' and P'' in linear time imply the following.

Lemma 3.2 *The bubble merge*$\textcircled{\cap}$*of two admissible flat partitions P' and P'' can be computed in $O(|\mathcal{S}'| + |\mathcal{S}''|)$ time.*

3.2 Non-Flat C-Graphs

In this subsection, we show how the concepts of flat partition, good flat partition, admissible flat partition, generalized union, projection, and bubble merge need to

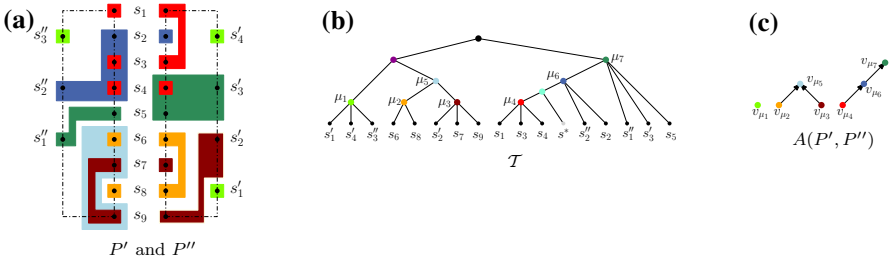

Fig. 4 Illustration for the construction of the auxiliary graph $A(P', P'')$ of two good non-flat partitions P' and P'' defined on the vertices of a c-graph $\mathcal{C}(G, \mathcal{T})$. The non-flat partitions P' and P'' are, in fact, also non-crossing, and thus admissible

be naturally extended to non-flat c-graphs. To this aim, recall that an intra-cluster edge is an edge that connects two vertices of the same cluster. Therefore, in a non-flat c-graph $\mathcal{C}(G, \mathcal{T})$, an intra-cluster edge for a cluster $\mu \in \mathcal{T}$ is also an intra-cluster edge for all the clusters that are encountered on the path from μ to the root of $\mathcal{T}$.

In the context of non-flat c-graphs, we are going to use an extension of the notion of non-crossing partition that takes into account the inclusion between clusters. Let $\mathcal{Q} = \{q_1, q_2, \ldots, q_n\}$ be a ground set. In a *recursive partition* $P = \{Q_1, \ldots, Q_k\}$ of $\mathcal{Q}$, it holds that either $Q_i \cap Q_j = \emptyset$ or $Q_i \subset Q_j$, with $1 \leq i < j \leq k$. Observe that $k \leq 2|\mathcal{Q}| - 1$, as it is possible to represent the hierarchy of P by means of a rooted tree whose leaves are the elements in $\mathcal{Q}$ and whose internal vertices have degree greater than 2, except possibly for the root which may have degree 2.

Let now $\mathcal{S} = (s_1, s_2, \ldots, s_n)$ be a cyclically-ordered set. A recursive partition P of $\mathcal{S}$ is *crossing*, if there exist elements $a, c \in S_i$ and $b, d \in S_j$, with $S_i, S_j \in P$ and $S_i \cap S_j = \emptyset$ (i.e. neither S_i nor S_j are contained in one another), such that $a \prec_b c$ and $c \prec_d a$; and, it is *non-crossing*, otherwise. We observe that the size of the set $\mathcal{RNC}(\mathcal{S})$ of all the non-crossing recursive partitions of $\mathcal{S}$ coincides with $\mathcal{CAT}(2n-1)$, as each non-crossing recursive partition in $\mathcal{RNC}(\mathcal{S})$ can be paired with exactly one rooted ordered tree on $2n-1$ vertices whose leaves are the elements of $\mathcal{S}$ [50, Chapter 32].

Let $\mathcal{C}(G, \mathcal{T})$ be a c-graph. The *depth* of a cluster $\mu \in \mathcal{T}$ is the length of the path between μ and the root cluster in $\mathcal{T}$. Observe that, the root cluster has depth 0. A cluster $\nu \in \mathcal{T}$ is an *ancestor* (resp., *successor*) of a cluster $\mu \in \mathcal{T}$, with $\mu \neq \nu$, if μ belongs to the subtree of $\mathcal{T}$ rooted at ν (resp., ν belongs to the subtree of $\mathcal{T}$ rooted at μ). Observe that, a cluster is neither an ancestor nor a successor of itself. Given a set $S \subseteq V(G)$, the *lowest common ancestor* of S *in* $\mathcal{T}$ is the deepest cluster $\mu \in \mathcal{T}$ such that all the vertices of S are in V_μ. A *non-flat partition* of a set $V' \subseteq V(G)$ is a recursive partition of V' that is also a clustered-annotated set of $\mathcal{C}$. A non-flat partition $\mathcal{S} = \{S_1, \ldots, S_k\}$ is *good* if (i) all the vertices of S_i are in V_{μ_i}, where μ_i is the cluster associated with S_i, for each part $S_i \in \mathcal{S}$, and (ii) for any two parts $S_i, S_j \in \mathcal{S}$ such that $S_i \subset S_j$, μ_i should be a successor of μ_j. Let $\mathcal{S} = \{S_1, \ldots, S_k\}$ be a good non-flat partition. Since $\mathcal{C}$ is a non-flat c-graph and $\mathcal{S}$ is good, for each part $S_i \in \mathcal{S}$, the cluster associated with S_i must be either the lowest common ancestor μ_i of S_i in $\mathcal{T}$ or a non-root ancestor of μ_i in $\mathcal{T}$. The details of how this association is defined are given in Sect. 3.3. Furthermore, a non-flat partition of a cyclically-ordered set $V' \subseteq V(G)$ is

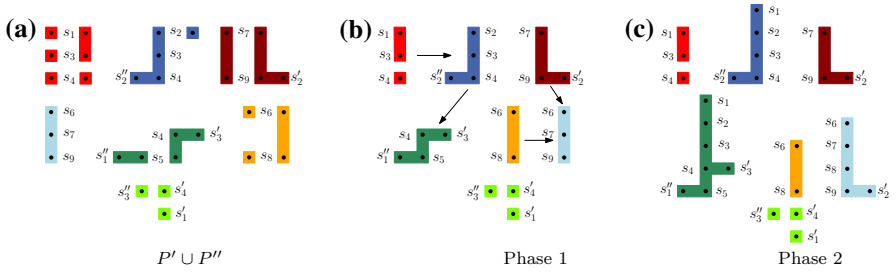

Fig. 5 Illustrations for the construction of the generalized union of the two admissible non-flat partitions P' and P'' illustrated in Fig. 4a. Different parts are represented by shaded polygons. The arrows in **b** show how the parts obtained in Phase 1 are merged in Phase 2, according to the auxiliary graph $A(P', P'')$, to obtain **c**

admissible if it is both good and non-crossing. The definition of *projection* of a non-flat partition *onto* a set is identical to the corresponding definition in the flat setting.

In order to the extend the definition of generalized union to be applicable to pairs of good non-flat partitions, we introduce the following auxiliary directed graph; refer to Fig. 4. Given a pair (P', P'') of good non-flat partitions of sets $\mathcal{Q}' \subset V(G)$ and $\mathcal{Q}'' \subset V(G)$, respectively, we construct a graph $A(P', P'')$ as follows. The vertex set of $A(P', P'')$ contains a vertex v_μ if there exists a part $S \in P' \cup P''$ that is associated with the cluster μ. The edge set of $A(P', P'')$ contains an edge $v_\mu v_\nu$ directed from v_μ to v_ν, if ν is an ancestor of μ in $\mathcal{T}$ and there exists no vertex v_τ in the vertex set of $A(P', P'')$ such that the cluster τ is in the path connecting μ and ν in $\mathcal{T}$. Observe that, by construction, the underlying undirected graph of $A(P', P'')$ is a forest. Also, two parts $S', S'' \in P' \cup P''$ intersect only if there exists a directed path in $A(P', P'')$, possibly of length 0, between the vertices corresponding to the clusters to which S' and S'' are associated with. This holds since there exists a directed path between two vertices of $A(P', P'')$ only if the corresponding clusters are one ancestor of the other in $\mathcal{T}$.

We are now ready to define the *generalized union* $\uplus_R$ of P' and P'', which returns a non-flat partition $P^* = P' \uplus_R P''$ of $\mathcal{Q}' \cup \mathcal{Q}''$ obtained as follows; refer to Fig. 5. We initialize $P^* = P' \cup P''$. Then, we perform the following procedure consisting of two phases.

- **Phase 1.** We visit the vertices of $A(P', P'')$. When we are at a vertex v_μ, as long as there exist $Q_i, Q_j \in P^*$ such that Q_i and Q_j are associated with the same cluster μ and $Q_i \cap Q_j \neq \emptyset$, we remove sets Q_i and Q_j from P^* and add the set $Q_i \cup Q_j$ to P^*. By Condition a of Definition 3.1, the set $Q_i \cup Q_j$ is associated with the cluster μ. Observe that, after this phase is completed, any two parts $Q_i, Q_j \in P^*$ intersect only if Q_i and Q_j are associated with different clusters.
- **Phase 2.** We again visit the vertices of $A(P', P'')$, but according to a topological ordering of such vertices from sources to sinks. When we are at a vertex v_μ, as long as there exist $Q_i, Q_j \in P^*$ such that (a) $Q_i \cap Q_j \neq \emptyset$, and (b) Q_i is associated with the cluster μ, Q_j is associated with the cluster ν, and $v_\mu v_\nu$ is an edge of

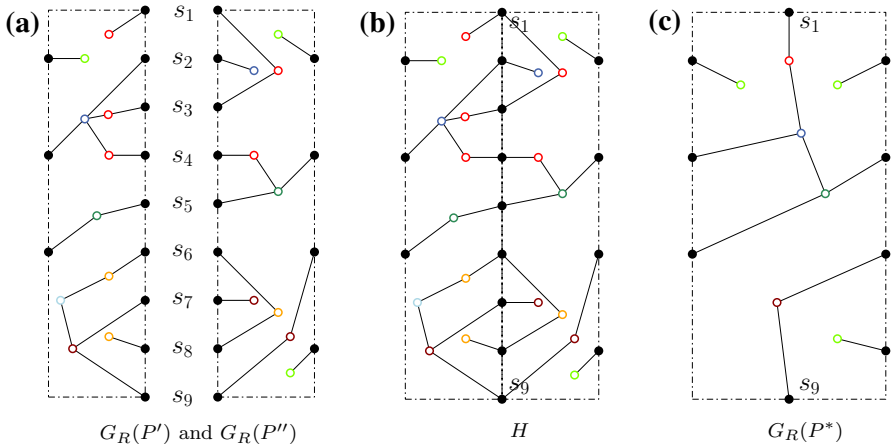

Fig. 6 Illustrations for the construction of the bubble merge of the two admissible non-flat partitions P' and P'' illustrated in Fig. 4a

$A(P', P'')$, we remove the part Q_j from P^* and add the part $Q_i \cup Q_j$ to P^*. By Condition a of Definition 3.1, the set $Q_i \cup Q_j$ is associated with the cluster v.

By the above definition, the generalized union of P' and P'' can be easily computed in quadratic time. We formalize this fact in the following.

Lemma 3.3 *The generalized union $P^* = P' \uplus P''$ of two good non-flat partitions P' and P'' can be computed in $O\big((|\mathcal{Q}'| + |\mathcal{Q}''|)^2\big)$ time.*

In the same way as an admissible flat partition P can be naturally associated with a cycle-star $G(P)$, an admissible non-flat partition P' of a cyclically-ordered set $\mathcal{S} \subset V(G)$ can be naturally associated with a 2-connected plane graph $G_R(P')$, as follows. The outer face of $G_R(P')$ is a cycle $C(P')$ whose vertices are the elements in $\mathcal{S}$ and the clockwise order in which they appear along $C(P')$ is the same as in $\mathcal{S}$. Also, for each part $S_i \in P'$, graph $G_R(P')$ contains a vertex v_i in the interior of $C(P')$. We further add the following edges to the graph $G_R(P')$. First, we add an edge connecting two vertices v_i and v_j if and only if S_j is the smallest part in P' such that $S_i \subset S_j$. Then, we add an edge connecting a vertex $v \in \mathcal{S}$ with a vertex v_i if and only if S_i is the smallest part in P' containing v. We say that $G_R(P')$ is the *cycle-tree* associated with P'; see Fig. 6c for an example.

The definition of *bubble merge*() of two admissible non-flat partition is identical to the corresponding definition in the flat setting where, however, the operator $\uplus_R$ is used instead of the operator $\uplus$, and the role played by cycle-stars is now assumed by the cycle-trees associated with the input admissible non-flat partitions; refer to Fig. 6. Since the cycle-tree $G_R(P)$ associated with an admissible non-flat partition on a cyclically-ordered set $\mathcal{S} \subseteq V(G)$ can be constructed in $O(|\mathcal{S}|^2)$ time and it has size linear in $|\mathcal{S}|$, the bubble merge of two admissible

non-flat partitions P' and P'' of cyclically-ordered sets $\mathcal{S}'$ and $\mathcal{S}''$, respectively, can be easily computed in quadratic time. We formalize this fact in the following.

Lemma 3.4 *The bubble merge of two admissible non-flat partitions P' and P'' can be computed in $O\big((|\mathcal{S}'| + |\mathcal{S}''|)^2\big)$ time.*

3.3 Algorithm

Let $\mathcal{C}(G, \mathcal{T})$ be a 2-connected, possibly non-flat, c-graph. Let (D, γ) be a bond-carving decomposition of $\delta(G)$ of width at most ω and let v be a non-root bag of D. We denote by F_v the set of faces of G that are dual to the vertices of $\delta(G)$ that are leaves of the subtree D_v of D rooted at v. Also, let G_v be the embedded subgraph of G induced by the edges incident to the faces in F_v. Since (D, γ) is a bond-carving decomposition of $\delta(G)$, there exists a *unique* face of G_v not in F_v, which we denote as f_v^∞. The *interface graph* I_v of v is the subgraph of G_v induced by the edges that are incident to f_v^∞. The *boundary* B_v of v is the vertex set of I_v. Note that, the edges of I_v are dual to those in $\gamma(v)$. By Lemma 2.1 and by the definition of bond-carving decomposition, we derive the next observation about I_v.

Observation 3.1 *The interface graph I_v of v is a cycle of length at most ω.*

Since G is 2-connected, by Observation 3.1 the vertices in B_v have a natural (clockwise) circular ordering defined by cycle I_v. Therefore, from now on, we regard B_v as a cyclically-ordered set.

In the following, for the sake of simplicity, we refer to flat partitions and non-flat partitions simply as *partitions*, and assume that in the context of non-flat and flat c-graphs such a term refers to the appropriate notion. Similarly, we denote the generalized union of good partitions by $\uplus$, and assume that in the context of non-flat and flat c-graphs such a term refers to the operation $\uplus_F$ and $\uplus_R$, respectively. We will exploit the following useful observation concerning the generalized union.

Observation 3.2 *Let P' and P'' be two good partitions of a set $V' \subset V(G)$, let $P^* = P' \uplus P''$, and let S be a part in P'. Then, there exists a part $S^* \in P^*$ such that $S \subseteq S^*$ and S^* is associated with the cluster S is associated with.*

Let $P \in \mathcal{NC}(B_v)$ be an admissible partition and let $\mathcal{C}_v(G_v, \mathcal{T}_v)$ be the c-graph obtained by restricting $\mathcal{C}$ to G_v. Also, let $\mathcal{C}_v^\diamond(G_v^\diamond, \mathcal{T}_v^\diamond)$ be a super c-graph of $\mathcal{C}_v$ containing no saturating edges in the interior of f_v^∞ and such that $G_v^\diamond$ is planar. We have the following definition.

Definition 3.3 The c-graph $\mathcal{C}_v^\diamond$ *realizes* P if (refer to Fig. 7):

(a) for every two vertices $u, v \in B_v$, we have that if u and v are in the same part $S_i \in P$, then they are connected in $G_v^\diamond$ by paths of intra-cluster edges of the cluster S_i is associated with,
(b) for every two vertices $u, v \in B_v$ and for every path p_{uv} connecting u and v in $G_v^\diamond$, consider the lowest common ancestor μ of $V(p_{uv})$ in $\mathcal{T}$. If μ is a non-root cluster, then there exists a part $S_i \in P$ such that $u, v \in S_i$ and the cluster associated with S_i is either μ or a non-leaf successor of μ,
(c) for each cluster μ in $\mathcal{T}$ such that $V_\mu \cap B_v \neq \emptyset$, each vertex of μ in G_v is connected to some vertex of μ in B_v by paths of intra-cluster edges of μ in $G_v^\diamond$,
(d) for each cluster μ in $\mathcal{T}$ such that $V_\mu \cap B_v \neq \emptyset$ and for every vertex $v \in V(\mu) \cap B_v$, there exists a part in P that contains v which is associated with μ or with a non-leaf successor of μ, and
(e) for each cluster μ in $\mathcal{T}$ such that $V_\mu \cap B_v = \emptyset$ and $V_\mu \cap V(G_v) \neq \emptyset$, all the vertices of μ are in G_v and are connected by paths of intra-cluster edges of μ in $G_v^\diamond$.

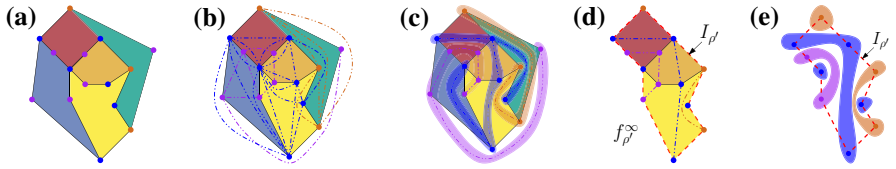

Fig. 7 **a** A flat c-graph $\mathcal{C}(G, \mathcal{T})$. **b** A super c-graph of $\mathcal{C}$ containing all the candidate saturating edges. **c** A c-planar drawing of $\mathcal{C}$ and the corresponding planar saturation. **d** A planar saturation of a c-graph, whose underlying graph is the graph $G_{\rho'}$ of the decomposition in Fig. 1, where no saturating edge lies in the interior of $f_{\rho'}^\infty$. **e** The admissible flat partition P determined by the planar saturation in **d**; sets of vertices of $I_{\rho'}$ belonging to the same cluster and connected by saturating edges in **d** form distinct parts in P (enclosed by shaded regions)

Conditions a and b of Definition 3.3 imply that each part S_i of P is in a one-to-one correspondence with a connected component of $G_v^\diamond[V_\mu]$, where μ is the cluster S_i is associated with. Conditions c and e of Definition 3.3 do not depend on partition P. However, they are necessary for the existence of a super c-graph $\mathcal{C}^*$ of $\mathcal{C}$ satisfying Theorem 2.1 whose restriction to G_v coincides with $\mathcal{C}_v^\diamond$. In fact, Condition c of Definition 3.3 reflects the possibility for the clusters which intersect B_v and that are not yet connected by means of intra-cluster edges of $\mathcal{C}_v^\diamond$ to reach connectivity by means of intra-cluster edges of $\mathcal{C}^*$ that lie outside I_v. Let μ' be a cluster such that $V_{\mu'} \cap B_v \neq \emptyset$. Observe that, by Condition c of Definition 3.3, a connected component of $G_v^\diamond[V_{\mu'}]$ is in one-to-one correspondence with a partition of the set $V_{\mu'} \cap B_v$ such that all the vertices belonging to the same connected component of $G_v^\diamond[V_{\mu'}]$ are in the same part. This observation, together with Conditions b and d of Definition 3.3, ensures that each connected component of $G_v^\diamond[V_{\mu'}]$ is represented by a part of P. Moreover, Condition e of Definition 3.3 ensures that cycle I_v does not form a *cluster separator*, that is, a cycle of G such that the

vertices of some cluster μ appear both in its interior and in its exterior, but not in it. Thus, partition P *"represents"* the internal-cluster connectivity in $\mathcal{C}_\nu^\diamond$ of the clusters whose vertices appear in B_ν *in a potentially positive instance*. Also, P is *realizable* by $\mathcal{C}_\nu$ if there exists a super c-graph $\mathcal{C}_\nu^\diamond(G_\nu^\diamond, \mathcal{T}_\nu^\diamond)$ of $\mathcal{C}_\nu$ that realizes P containing no saturating edges in the interior of f_ν^∞ and such that $G_\nu^\diamond$ is planar.

We are going to exploit the next lemma, which holds for any bond-carving decomposition.

Lemma 3.5 *Let ρ' and ρ'' be the two children of the root ρ of D. Then , $I_{\rho'} = I_{\rho''}$.*

Proof Let $\mathcal{L}_{\rho'}$ and $\mathcal{L}_{\rho''}$ be the set of leaves of the subtree of D rooted at ρ' and ρ'', respectively. Recall that, $\gamma(\rho') = E(\mathcal{L}_{\rho'}, V \setminus \mathcal{L}_{\rho'})$ and $\gamma(\rho'') = E(\mathcal{L}_{\rho''}, V \setminus \mathcal{L}_{\rho''})$. Since $\mathcal{L}_{\rho'} = V \setminus \mathcal{L}_{\rho''}$ and $\mathcal{L}_{\rho''} = V \setminus \mathcal{L}_{\rho'}$, we have $\gamma(\rho') = \gamma(\rho'')$. Thus, cycles $I_{\rho'}$ and $I_{\rho''}$ are composed of the edges of G that are dual to the same edges of $\delta(G)$. □

Lemma 3.5 allows us to derive the following useful characterization.

Theorem 3.1 (Characterization) *The 2-connected embedded c-graph $\mathcal{C}(G, \mathcal{T})$ is c-planar if and only if there exist admissible partitions $P' \in \mathcal{NC}(B_{\rho'})$ and $P'' \in \mathcal{NC}(B_{\rho''})$, where ρ' and ρ'' are the two children of the root ρ of D, such that:*

(i) *P' and P'' are realizable by $\mathcal{C}_{\rho'}(G_{\rho'}, \mathcal{T}_{\rho'})$ and by $\mathcal{C}_{\rho''}(G_{\rho''}, \mathcal{T}_{\rho''})$, respectively, and*
(ii) *for each cluster μ in $\mathcal{T}$ such that $V_\mu \cap B_{\rho'} \neq \emptyset$, there exists exactly one part $S \in P^*$, with $P^* = P' \uplus P''$, such that $S = V_\mu \cap B_{\rho'}$ and S is associated with either μ or with a non-leaf successor of μ.*

Proof We first prove the *only if* part. Let $\mathcal{C}^\diamond(G^\diamond, \mathcal{T}^\diamond)$ be a c-connected super c-graph of $\mathcal{C}$ with $G^\diamond$ planar, which exists by Theorem 2.1. Let $\mathcal{C}_{\rho'}^\diamond(G_{\rho'}^\diamond, \mathcal{T}_{\rho'}^\diamond)$ and $\mathcal{C}_{\rho''}^\diamond(G_{\rho''}^\diamond, \mathcal{T}_{\rho''}^\diamond)$ be the super c-graphs of $\mathcal{C}_{\rho'}$ and $\mathcal{C}_{\rho''}$ induced by $\mathcal{C}^\diamond$, respectively. We first define two admissible partitions $P' \in \mathcal{NC}(B_{\rho'})$ and $P'' \in \mathcal{NC}(B_{\rho''})$ as follows. We describe the construction of P', the construction of P'' being analogous. For each cluster $\mu \in \mathcal{T}_{\rho'}$, consider each connected component H of the cluster μ restricted to $G_{\rho'}^\diamond$. Let $S = V(H) \cap B_{\rho'}$. If $S \notin P'$, then we associate S with μ and add it to P'. Otherwise, S already belongs to P'; let ν be the cluster associated with S. Observe that, ν is either a successor or an ancestor of μ. We associate S with the deepest cluster between ν and μ. Clearly, by construction, the partitions P' and P'' are realizable by $\mathcal{C}_{\rho'}(G_{\rho'}, \mathcal{T}_{\rho'})$ and $\mathcal{C}_{\rho''}(G_{\rho''}, \mathcal{T}_{\rho''})$, respectively. Observe that, by the above construction, for each cluster μ in $\mathcal{T}$ such that $V_\mu \cap B_{\rho'} \neq \emptyset$, each vertex in $V_\mu \cap B_{\rho'}$ belongs to at least one part in P' (and to at least one part in P'') that is associated with either μ or with a non-leaf successor of μ. Let μ be a cluster such that $V_\mu \cap B_{\rho'} \neq \emptyset$. Since $\mathcal{C}^\diamond(G^\diamond, \mathcal{T}^\diamond)$ is a c-connected c-graph, the cluster μ induces a single connected component. Therefore, by the definition of generalized union, all the parts in P' and in P'' whose vertices belong to V_μ contribute to generating a part $S \in P^*$, with $P^* = P' \uplus P''$, such that

$S = V_\mu \cap B_{\rho'}$. In particular, by Condition a of Definition 3.1, S is associated with either μ or with a non-leaf successor of μ.

We now prove the *if part*. By Lemma 3.5, it holds $G = G_{\rho'} \cup G_{\rho''}$ and $I_{\rho'} = I_{\rho''} = G_{\rho'} \cap G_{\rho''}$. Let $\mathcal{C}^\diamond_{\rho'}$ be a super c-graph of $\mathcal{C}_{\rho'}$ realizing P' and let $\mathcal{C}^\diamond_{\rho''}$ be a super c-graph of $\mathcal{C}_{\rho''}$ realizing P''; these c-graphs exist since Condition i holds. Let $\mathcal{C}^\diamond$ be the super c-graph of $\mathcal{C}$ obtained by augmenting $\mathcal{C}$ with the saturating edges of both $\mathcal{C}^\diamond_{\rho'}$ and $\mathcal{C}^\diamond_{\rho''}$. We have that $G^\diamond$ is planar since both $G^\diamond_{\rho'}$ and $G^\diamond_{\rho''}$ are planar, since $G^\diamond_{\rho'}$ and $G^\diamond_{\rho''}$ share the common interface $I_{\rho'} = I_{\rho''}$, by Lemma 3.5, and since there are no saturating edges of $\mathcal{C}^\diamond_{\rho'}$ (resp. $\mathcal{C}^\diamond_{\rho''}$) that lie in the interior of $f^\infty_{\rho'}$ (resp. $f^\infty_{\rho''}$).

We show that every cluster μ is connected in $\mathcal{C}^\diamond$, provided that Condition ii holds. This proves that $\mathcal{C}^\diamond$ is a c-connected super c-graph of $\mathcal{C}$, thus by Condition iii of Theorem 2.1, c-graph $\mathcal{C}$ is c-planar. We let $B = B_{\rho'} = B_{\rho''}$ and distinguish two cases, based on whether some vertices of μ appear along cycle $I_{\rho'} = I_{\rho''}$ or not.

Consider first a cluster μ containing vertices in B. By Condition c of Definition 3.3, we have that every vertex in μ either belongs to B or it is connected by paths of intra-cluster edges of μ in either $\mathcal{C}^\diamond_{\rho'}$ or $\mathcal{C}^\diamond_{\rho''}$ to a vertex in B. Since, by Condition ii of the statement, there exists only one part $S_\mu \in P^*$ such that $S_\mu = V_\mu \cap B_{\rho'}$, we have that the different parts of P' and of P'' containing vertices of μ are joined together by the vertices of μ in B. Therefore, the cluster μ is connected in $\mathcal{C}^\diamond$. Finally, consider a cluster μ such that no vertex of μ belongs to B. Then, all the vertices of cluster μ only belong to either $\mathcal{C}_{\rho'}$ or $\mathcal{C}_{\rho''}$, by Condition e of Definition 3.3. Suppose that μ only belongs to $\mathcal{C}_{\rho'}$, the case when μ only belongs to $\mathcal{C}_{\rho''}$ is analogous. Since $\mathcal{C}^\diamond_{\rho'}$ realizes P', by Condition e of Definition 3.3, all the vertices of μ in G'_ρ are connected by paths of intra-cluster edges of μ. Thus, cluster μ is connected in $\mathcal{C}^\diamond$, since it is connected in $\mathcal{C}^\diamond_{\rho'}$. This concludes the proof. □

Theorem 3.1 implies that the algorithmic core of our problem lies in the computation of the set A_ν of all admissible partitions of B_ν that are realizable by $\mathcal{C}_\nu$, for every non-root bag ν of D. Observe that the size of A_ν depends on both the size of B_ν and the height h of $\mathcal{T}$ as any part in a partition can be associated with at most $h-2$ clusters, namely the non-root clusters in the path between the lowest common ancestor of the vertices in the part and the root of $\mathcal{T}$. Therefore, we can upper bound $|A_\nu|$ in the flat and in the non-flat case by $\mathcal{NC}(B_\nu)$ and by $h^{2|B_\nu|-1} \cdot \mathcal{RNC}(B_\nu)$, respectively.

We now present our main algorithmic tool.

Algorithm 1. Let (D, γ) be a bond-carving decomposition of $\delta(G)$ of width ω. We process the bags of D bottom-up and compute the following *relevant information*, for each non-root bag ν of D: (1) the set A_ν, and (2) for each admissible partition $P \in A_\nu$, for each part $S_i \in P$, and for each cluster μ that is either the cluster associated with S_i or a non-root ancestor of such cluster, the number $count(S_i, \mu)$ of vertices of cluster μ belonging to G_ν which are either in S_i or are connected to some vertex of S_i by paths of intra-cluster edges of μ in $\mathcal{C}^\diamond_\nu$. Since there may exist parts $S_1, S_2 \in P$ such that $S_1 \supset S_2$, for any cluster μ which is either the cluster associated with S_1 or a non-root ancestor of such cluster, we have that the vertices of cluster μ belonging to G_ν which are either in S_2 or are connected to some vertex of S_2 by paths

of intra-cluster edges of μ in $\mathcal{C}_\nu^\diamond$ are also counted in $count(S_1, \mu)$. In that case, to avoid double counting of these vertices of μ, the value of $count(S_2, \mu)$ is set to NULL.

- If ν is a leaf bag of D, then $G_\nu = I_\nu$ consists of the vertices and edges of a single face of G. Further, by Observation 3.1, graph G_ν is a cycle of length at most ω. In this case, A_ν simply coincides with the set of all the admissible partitions of B_ν for which each part is associated with its lowest common ancestor. In fact, since G_ν is a simple cycle I_ν, for any admissible partition P of B_ν we can construct a super c-graph $\mathcal{C}_\nu^\diamond(G_\nu^\diamond, \mathcal{T}_\nu^\diamond)$ of $\mathcal{C}_\nu$ that realizes P containing no saturating edges in the interior of f_ν^∞ and such that $G_\nu^\diamond$ is planar as follows: Let $G_R(P)$ (resp. $G(P)$) be the cycle-tree (resp. cycle-star) associated with the partition P if $\mathcal{C}$ is non-flat (resp. flat). Since the proof is identical in both the cases, in the following we denote both $G_R(P)$ and $G(P)$ by G_P. Recall that each vertex $u \in V(G_P) \setminus B_\nu$ corresponds to a part S_u in P and that such a part is associated with the cluster μ_u. Consider an edge $(u, v) \in E(G_P)$ such that $v \in B_\nu$, $u \in V(G_P) \setminus B_\nu$, and such that there exists no other vertex $w \in V(G_P) \setminus B_\nu$ for which μ_w is a descendant of μ_u in $\mathcal{T}$. Contracting the edge (u, v) to v results in a plane graph such that $G_P[S_u]$ is connected. We repeatedly identify such an edge and contract it to its endpoint in B_ν. Observe that, at the end of above recursive procedure, we get that $V(G_P) = B_\nu$, that G_P is a super graph of I_ν, and that, for every part $S \in P$, the graph $G_P[S]$ is connected. By considering $G_\nu^\diamond = G_P$ and $\mathcal{T}_\nu^\diamond = \mathcal{T}_\nu$, we obtain the desired super c-graph $\mathcal{C}_\nu^\diamond$ of $\mathcal{C}_\nu$ that realizes P. Therefore, we can construct A_ν by enumerating all the possible non-crossing partitions of B_ν and by testing whether each such partition is good in $O(\omega)$ time. Note that, the total number of non-crossing partition of B_ν is at most $\mathcal{CAT}(\omega) \leq 2^{2\omega}$ when $\mathcal{C}$ is flat, and it is at most $\mathcal{CAT}(2\omega - 1) \leq 2^{4\omega-2}$ when $\mathcal{C}$ is non-flat. Further, for each $P \in A_\nu$, we can compute all counters $count(S_i, \mu)$ for every $S_i \in P$ and for every cluster μ that is either the cluster associated with S_i or a non-root ancestor of such cluster, in total $O(\omega + h)$ time, by visiting G_P.
- If ν is a non-leaf non-root bag of D, we have already computed the relevant information for the two children ν' and ν'' of ν. In the following way, we either detect that $\mathcal{C}$ does not satisfy Condition iii of Theorem 2.1 or construct the relevant information for ν:

  (1) Initialize $A_\nu = \emptyset$;
  (2) For every pair of realizable admissible partitions $P' \in A_{\nu'}$ and $P'' \in A_{\nu''}$, perform the following operations:

    (2a) Compute $P^* = P' \uplus P''$ and compute the counters $count(S_i, \mu)$ (as described in the proof of Lemma 3.6), for each $S_i \in P^*$ and for each cluster μ that is either the cluster associated with S_i or a non-root ancestor of such cluster, from the counters of the parts in $P' \cup P''$ whose union is S_i.
    (2b) If there exists some $S_i \in P^*$ such that $S_i \cap B_\nu = \emptyset$ and some cluster μ that is either the cluster associated with S_i or a non-root ancestor of

such cluster for which $count(S_i, \mu) \neq \texttt{NULL}$ and $count(S_i, \mu)$ is smaller than the number of vertices in μ, then **reject** the instance.

(2c) Compute ▯ and add P to A_ν.

(3) Remove duplicates from A_ν, if any.

Remark 3.2 Algorithm 1 rejects the instance at step (2b), if I_ν forms a cluster separator. This property is independent of the specific generalized union P^* considered at this step and implies that no P^* (and, thus, no P at step (2c)) can satisfy Condition e of Definition 3.3.

The next lemma is concerned with the correctness and the time complexity of Algorithm 1.

Lemma 3.6 *For each non-root bag ν of D, Algorithm 1 computes the relevant information for ν in $O(2^{4\omega+\log\omega})$ time if $\mathcal{C}$ is flat and in $O(4^{4\omega+\log\omega}n)$ time if $\mathcal{C}$ is non-flat, given the relevant information for its children.*

Proof Let ν' and ν'' be the two children of ν in D. We will first show the correctness of the algorithm and then argue about the running time.

Let A_ν be the set of all the admissible partitions of B_ν that are realizable by $\mathcal{C}_\nu$ and let A^*_ν be the set of all the admissible partitions of B_ν computed by Algorithm 1. We show $A^*_\nu = A_\nu$.

We first prove $A_\nu \subseteq A^*_\nu$. Let P_ν be a realizable admissible partition in A_ν. Since P_ν is realizable by $\mathcal{C}_\nu$, there exists a super c-graph $\mathcal{C}^*_\nu(G^*_\nu, \mathcal{T}^*_\nu)$ of $\mathcal{C}_\nu$ that realizes P_ν containing no saturating edges in the interior of f^∞_ν and such that G^*_ν is planar. Let $\mathcal{C}^*_{\nu'}(G^*_{\nu'}, \mathcal{T}^*_{\nu'})$ (resp. $\mathcal{C}^*_{\nu''}(G^*_{\nu''}, \mathcal{T}^*_{\nu''})$) be the super c-graph of $\mathcal{C}_{\nu'}(G_{\nu'}, \mathcal{T}_{\nu'})$ (resp. of $\mathcal{C}_{\nu''}(G_{\nu''}, \mathcal{T}_{\nu''})$) obtained by adding to $\mathcal{C}_{\nu'}$ (resp. to $\mathcal{C}_{\nu''}$) all the saturating edges in G^*_ν laying in the interior of the faces of $G_{\nu'}$ (resp. of $G_{\nu''}$) that are also faces of G_ν. Clearly, c-graph $\mathcal{C}^*_{\nu'}(G^*_{\nu'}, \mathcal{T}^*_{\nu'})$ (resp. c-graph $\mathcal{C}^*_{\nu''}(G^*_{\nu''}, \mathcal{T}^*_{\nu''})$) contains no saturating edges in the interior of $f^\infty_{\nu'}$ (resp. in the interior of $f^\infty_{\nu''}$), since such a face does not belong to G_ν. We first define two admissible partitions $P' \in \mathcal{NC}(B_{\nu'})$ and $P'' \in \mathcal{NC}(B_{\nu''})$ as follows. We describe the construction of P', the construction of P'' being analogous. For each cluster $\mu \in \mathcal{T}_{\nu'}$, consider each connected component H of the cluster μ restricted to $G^*_{\nu'}$. Let $S = V(H) \cap B_{\nu'}$. If $S \notin P'$, then we associate S with μ and add it to P'. Otherwise, S already belongs to P'; let α be the cluster associated with S. Observe that, α is either a successor or an ancestor of μ. We associate S with the deepest cluster between α and μ. Clearly, by construction, the partitions P' and P'' are realizable by $\mathcal{C}_{\nu'}(G_{\nu'}, \mathcal{T}_{\nu'})$ and $\mathcal{C}_{\nu''}(G_{\nu''}, \mathcal{T}_{\nu''})$, respectively. By hypothesis, we have $P' \in A_{\nu'}$ and $P'' \in A_{\nu''}$. We show that when step (2) of Algorithm 1 considers partitions P' and P'', it successfully adds P_ν to the set A^*_ν. By the construction of $\mathcal{C}^*_{\nu'}$ and $\mathcal{C}^*_{\nu''}$ and the subsequent definitions of P' and of P'', we have that ▯. Therefore, we only need to show that when the algorithm considers the pair (P', P''), it does not reject the instance at step (2b), and thus P_ν is added to A^*_ν at step (2c).

Let $P^* = P' \uplus P''$, which is constructed at step (2a) of the algorithm. Suppose, for a contradiction, that $\mathcal{C}$ is rejected at step (2b). Then, there exists a part S_i of P^* and a cluster μthat is either the cluster associated withS_i or a non-root ancestor of such cluster such that $S_i \cap B_v = \emptyset$ and $count(S_i, \mu)$ is smaller than the number of vertices in the cluster μ. Therefore, the cluster μ contains vertices that belong to $G \setminus G_v$, which implies that P_v cannot satisfy Condition e of Definition 3.3, a contradiction. This concludes the proof of this direction.

We now prove $A_v^* \subseteq A_v$. Let P be a partition in A_v^* obtained from the partitions $P' \in A_{v'}$ and $P'' \in A_{v''}$ (selected at step (2) of the algorithm). Next, we show that P is realizable by $\mathcal{C}_v$. This implies that $A_v^* \subseteq A_v$.

By the definition of realizable partition, there exists a super c-graph $\mathcal{C}^*_{v'}(G^*_{v'}, \mathcal{T}^*_{v'})$ (resp. $\mathcal{C}^*_{v''}(G^*_{v''}, \mathcal{T}^*_{v''})$) of $\mathcal{C}_{v'}(G_{v'}, \mathcal{T}_{v'})$ (resp. of $\mathcal{C}_{v''}(G_{v''}, \mathcal{T}_{v''})$) that realizes P' (resp. P'') containing no saturating edges in the interior of $f^\infty_{v'}$ (resp. of $f^\infty_{v''}$) and such that $G^*_{v'}$ (resp. $G^*_{v''}$) is planar. Let $\mathcal{C}^*_v(G^*_v, \mathcal{T}^*_v)$ be the super c-graph of $\mathcal{C}_v(G_v, \mathcal{T}_v)$ constructed by adding to $\mathcal{C}_v$ the saturating edges in $\mathcal{C}^*_{v'}$ and $\mathcal{C}^*_{v''}$. We show that the c-graph $\mathcal{C}^*_v(G^*_v, \mathcal{T}^*_v)$ realizes P, contains no saturating edges in the interior of f^∞_v, and G^*_v is planar.

First, we have that G^*_v is planar, since $G^*_{v'}$ and $G^*_{v''}$ are planar and do not contain saturating edges in the interior of $f^\infty_{v'}$ and of $f^\infty_{v''}$, respectively. By the previous arguments, we also have that f^∞_v contains no saturating edges.

We show that Condition a of Definition 3.3 holds. Recall that $\emptyset$. We prove that all the vertices in the same part S of P are connected in $\mathcal{C}^*_v$ by means of intra-cluster edges of the cluster S is associated with. Let S_i be a part of P that also belongs to P' or to P''. Then, since $\mathcal{C}^*_{v'}$ and $\mathcal{C}^*_{v''}$ realize P' and P'', respectively, the vertices of S_i are connected in C^*_v by paths of intra-cluster edges of the cluster S_i is associated with, as they are connected by paths of intra-cluster edges of the cluster S_i is associated with, in either $\mathcal{C}^*_{v'}$ or $\mathcal{C}^*_{v''}$, by Condition a of Definition 3.3. Otherwise, let S_i be a part of P that does not belong to either P' or P''. Then, by Definition 3.2, the part S_i is obtained by intersecting B_v and a part S^*_i of the generalized union $P^* = P' \uplus P''$. Thus, by Condition b of Definition 3.1, S_i and S^*_i are associated with the same cluster μ. Also, the vertices in each of the parts of P' and of P'' contributing to the creation of S^*_i are connected by paths of intra-cluster edges of μ in $\mathcal{C}^*_{v'}$ and $\mathcal{C}^*_{v''}$, respectively, by Remark 3.1 and by Condition a of Definition 3.3. Therefore, we have that the connectivity of such sets implies the connectivity of the elements of S_i by paths of intra-cluster edges of μ that connect at their shared vertices in $B_{v'} \cap B_{v''}$.

Next, we show that Condition b of Definition 3.3 holds. We prove that for any two vertices u and v of B_v, if they are connected by a path L in G^*_v for which the lowest common ancestor of $V(L)$ in $\mathcal{T}$ is a non-root cluster μ', then they belong to a part of $P^* = P' \uplus P''$ associated with a cluster μ'', which is either μ' or a non-leaf successor of μ' in $\mathcal{T}$. By Definition 3.2 and by Condition b of Definition 3.1, this in turn implies that u and v belong to a part of P associated with μ''. Let $L_1, L_2, \cdots, L_k$ be the maximal subpaths of L composed of intra-cluster edges belonging to either $\mathcal{C}^*_{v'}$ or $\mathcal{C}^*_{v''}$ such that L_i and L_{i+1} share an endpoint, for $i \in \{1, \ldots, k-1\}$. Note that, the endpoints of each path L_i both belong to $B_{v'}$ or $B_{v''}$. Therefore, by Condition a of Definition 3.3, the endpoints of L_i belong to a part S_i of P' or P'', and the cluster associated with S_i is either μ' or a successor of μ'. As L_i and L_{i+1} share an endpoint, by the definition of generalized union, we have that P^* contains a part $S^* \supseteq S_1 \cup S_2 \cdots \cup S_k$.

Since $u \in S_1$ and $v \in S_k$, S^* contains both u and v. Moreover, by Condition a of Definition 3.1 and by Observation 3.2, the cluster associated with S^* is either μ' or a non-leaf successor of μ' in $\mathcal{T}$.

We now show that Condition c of Definition 3.3 holds. Suppose, for a contradiction, that there exists some cluster μ whose vertices appear in B_v such that there is at least one vertex x of μ in G_v that is not connected by a path of intra-cluster edges of μ to some vertex of μ in B_v. Since P' and P'' are realizable by $\mathcal{C}_{v'}$ and by $\mathcal{C}_{v''}$, by Condition c of Definition 3.3, there exists a vertex $t \in B_{v'} \cup B_{v''}$ of μ such that either $t = x$ or x is connected to t by means of intra-cluster edges of μ. If $t \in B_v$, we get a contradiction. Otherwise, consider the part $S_i \in P^*$ which contains t such that the cluster μ' associated with S_i is either μ or a non-leaf successor of μ. Note that, such a part exists by Condition d of Definition 3.3. We have that $S_i \cap B_v = \emptyset$ as otherwise there would exist a path of intra-cluster edges of μ connecting x to a vertex in B_v. Therefore, $count(S_i, \mu)$ is smaller than the number of vertices of μ since $V_\mu \cap B_v \neq \emptyset$. Thus, step (2b) would reject the instance, and P would not be added to A^*_v, a contradiction.

We now show that Condition d of Definition 3.3 holds. Let μ be a cluster in $\mathcal{T}$ such that $V_\mu \cap B_v \neq \emptyset$. As $B_v \subseteq B_{v'} \cup B_{v''}$, every vertex $v \in V_\mu \cap B_v$ belongs to $B_{v'} \cap B_v$ or $B_{v''} \cap B_v$. Since P' and P'' are realizable by $\mathcal{C}_{v'}$ and $\mathcal{C}_{v''}$, respectively, by Condition d of Definition 3.3, there exists a part S in P' or P'' such that $v \in S$ and the cluster associated with S is either μ or a non-leaf successor of μ. By Obsecrvation 3.2 and Condition d of Definition 3.1, we have a part in P containing v and it is associated with either μ or a non-leaf successor of μ.

Finally, we show that Condition e of Definition 3.3 holds. Suppose, for a contradiction, that there exists some cluster μ whose vertices only belong to $V(G_v) \setminus B_v$ and that there exist two vertices u and v of μ in G_v that are not connected by a path of intra-cluster edges of μ in G^*_v. First consider the case that $V_\mu \cap (B_{v'} \cup B_{v''} = \emptyset)$. Then, since P' and P'' are realizable by $\mathcal{C}_{v'}$ and $\mathcal{C}_{v''}$, by Condition e of Definition 3.3, we have that all the vertices of μ are either in $G_{v'}$ and are connected by paths of intra-cluster edges of μ in $G^*_{v'}$ or all the vertices of μ are either in $G_{v''}$ and are connected by paths of intra-cluster edges of μ in $G^*_{v''}$, a contradiction. Otherwise, by Condition c of Definition 3.3 there exists a vertex $t \in B_{v'} \cup B_{v''}$ of μ such that either $t = u$ or u is connected to t by means of intra-cluster edges of μ in either $\mathcal{C}_{v'}$ or $\mathcal{C}_{v''}$. If $t \in B_v$, we get a contradiction. Otherwise, consider the part $S_i \in P^*$ that contains t. By Condition d of Definition 3.3, the cluster μ' associated with S_i is either μ or a non-leaf successor of μ. We have that $S_i \cap B_v = \emptyset$ as $V_\mu \cap B_v = \emptyset$. Therefore, $count(S_i, \mu)$ is smaller than the number of vertices of μ, as $v \notin S_i$. Thus, step (2b) would reject the instance, and P would not be added to A^*_v, a contradiction.

We conclude by analyzing the running time.

First, we show that Step (2a) can be performed in $O(|B_{v'}| + |B_{v''}|)$ time if $\mathcal{C}$ is flat, and in $O((|B_{v'}| + |B_{v''}|)^2 \cdot h)$ time if $\mathcal{C}$ is non-flat. Since the generalized union P^* can be computed in $O(|B_{v'}| + |B_{v''}|)$ time, by Lemma 3.1 in the flat case, and in $O((|B_{v'}| + |B_{v''}|)^2)$ time, by Lemma 3.3 in the non-flat case, we only need to show that whenever a new part $S \in P^*$ stems from the union of two parts S_1 and S_2, we can compute the counters for S in $O(h)$ time. Clearly, the part S can be labeled with the two parts S_1 and S_2 generating it in the course of the generalized union

operation. Let μ_1 and μ_2 be the clusters associated with S_1 and S_2, respectively. By definition of generalized union, we have that $|S_1 \cap S_2| > 0$ and either $\mu_1 = \mu_2$ or one of μ_1 and μ_2, say μ_1, is an ancestor of the other. The counters associated with S are computed as follows. We first consider the case that $\mu_1 = \mu_2$. In this case, we set $count(S, \alpha) = count(S_1, \alpha) + count(S_2, \alpha) - k_\alpha$, where $k_\alpha = |(S_1 \cap S_2) \cap V_\alpha|$, for every cluster α that is either μ_1 or a non-root ancestor of μ_1. We now consider the case that μ_1 is an ancestor of μ_2. Similarly to the previous case, we set $count(S, \alpha) = count(S_1, \alpha) + count(S_2, \alpha) - k_\alpha$, for every cluster α that is either μ_1 or a non-root ancestor of μ_1. Recall that, in this case, we replace S_1 with S, but S_2 stays in P^* after the union of S_1 and S_2. Therefore, since the connectivity of the vertices of a cluster α, which is either μ_1 or an ancestor of μ_1, through the vertices in S_2 by means of intra-cluster edges of α is now reached through the vertices in S, we update $count(S_2, \alpha) = \texttt{NULL}$ to avoid double counting. Since, there can be at most $h - 2$ counters associated with any part, we obtain the claimed running time.

Step (2b) can be performed in $O(|B_{\nu'}| + |B_{\nu''}|)$ time if $\mathcal{C}$ is flat, and in $O((|B_{\nu'}| + |B_{\nu''}|) \cdot h)$ time if $\mathcal{C}$ is non-flat. This is due to the fact that the number of parts in P^* is $O(|B_{\nu'}| + |B_{\nu''}|)$, and that for each part which has an empty intersection with B_ν, we need to check all its at most $h - 2$ counters, if $\mathcal{C}$ is non-flat.

Step (2c) can be performed in $O(|B_{\nu'}| + |B_{\nu''}|)$ time if $\mathcal{C}$ is flat, and in $O((|B_{\nu'}| + |B_{\nu''}|)^2)$ time if $\mathcal{C}$ is non-flat as the bubble merge P can be computed in $O(|B_{\nu'}| + |B_{\nu''}|)$ time, by Lemma 3.2 in the flat case, and in $O((|B_{\nu'}| + |B_{\nu''}|)^2)$ time, by Lemma 3.4 in the non-flat case.

Step (3) can be performed in $O\big(|B_\nu| \cdot |\mathcal{NC}(B'_\nu)| \cdot |\mathcal{NC}(B''_\nu)|\big)$ time if $\mathcal{C}$ is flat, and in $O\big(|B_\nu| h^{2|B'_\nu|+2|B'_\nu|-2} \cdot |\mathcal{RNC}(B'_\nu)| \cdot |\mathcal{RNC}(B''_\nu)|\big)$ time if $\mathcal{C}$ is non-flat. In fact, the set A_ν computed at step (2) has size in $O(|A'_\nu| \cdot |A''_\nu|)$. In turn, $|A'_\nu|$ (resp. $|A''_\nu|$) is upper bounded in the flat and in the non-flat case by $|\mathcal{NC}(B'_\nu)|$ (resp. $|\mathcal{NC}(B''_\nu)|$) and by $h^{2|B'_\nu|-1} \cdot |\mathcal{RNC}(B'_\nu)|$ (resp. $h^{2|B''_\nu|-1} \cdot |\mathcal{RNC}(B''_\nu)|$), respectively. Therefore, since each partition in A_ν can be injectively mapped to a string whose length is in $O(|B_\nu|)$, in order to remove duplicates from A_ν, we can sort the partitions in A_ν in $O(|B_\nu| \cdot |A_\nu|)$ using radix sort, and then scan the obtained sorted list to remove consecutive duplicated partitions. This yields the stated running time for Step 3.

We are now the ready to provide the overall running time of Algorithm 1. The number of pairs of realizable partitions considered at Step (2) is bounded by $|\mathcal{NC}(B_{\nu'})| \cdot |\mathcal{NC}(B_{\nu''})|$, which is bounded by $2^{2(|B_{\nu'}|+|B_{\nu''}|)}$ if $\mathcal{C}$ is flat, and it is bounded by $h^{2|B'_\nu|+2|B'_\nu|-2} \cdot |\mathcal{RNC}(B'_\nu)| \cdot |\mathcal{RNC}(B''_\nu)|$, which is bounded by $h^{2|B'_\nu|+2|B'_\nu|-2} \cdot 2^{4(|B_{\nu'}|+|B_{\nu''}|)}$ if $\mathcal{C}$ is non-flat. Moreover, the running time of Step (2) is upper bounded by Step 2a. Since, $|B_{\nu'}| \leq \omega$ and $|B_{\nu''}| \leq \omega$, we therefore have that the overall running time of Step (2) is $O(\omega \cdot 2^{4\omega}) = O(2^{4\omega+\log\omega})$ time if $\mathcal{C}$ is flat, and in $O(\omega^2 \cdot h \cdot h^{2\omega-2} \cdot 2^{8\omega}) = O(4^{4\omega+\log\omega} h^{2\omega-1})$ time if $\mathcal{C}$ is non-flat. Observe that, the running time of Step (2) dominates the running time of Step (3). Thus, Algorithm 1 runs in $O(2^{4\omega+\log\omega})$ time, if $\mathcal{C}$ is flat, and in $O(4^{4\omega+\log\omega} h^{2\omega-1})$ if $\mathcal{C}$ is non-flat. □

By Lemma 3.6 and since D contains $O(n)$ bags, we have the following.

Lemma 3.7 *Sets $A_{\rho'}$ and $A_{\rho''}$ can be computed in $O(2^{4\omega+\log\omega}n)$ time, if $\mathcal{C}$ is flat, and in $O(4^{4\omega+\log\omega}h^{2\omega-1}n)$ if $\mathcal{C}$ is non-flat.*

We obtain the next theorems by combining Lemma 3.7 and Theorem 3.1, where the additive $O(n^2)$ factor in the running time derives from the time needed to convert a carving decomposition of $\delta(G)$ into a bond-carving decomposition of the same width [70].

Theorem 3.2 *C-Planarity Testing can be solved in $O(2^{4\omega+\log\omega}n + n^2)$ time for any 2-connected n-vertex embedded flat c-graph $\mathcal{C}(G,\mathcal{T})$, if a carving decomposition of $\delta(G)$ of width ω is provided.*

Theorem 3.3 *C-Planarity Testing can be solved in $O(4^{4\omega+\log\omega}h^{2\omega-1}n + n^2)$ time for any 2-connected n-vertex embedded non-flat c-graph $\mathcal{C}(G,\mathcal{T})$, if a carving decomposition of $\delta(G)$ of width ω is provided.*

We are finally ready to prove our main result.

Proof of Theorems 1.1 and 1.2 Let (D,γ) be a carving decomposition of $\delta(G)$ of optimal width $\omega = \mathrm{cw}(\delta(G))$. First, we apply Lemma 2.2 to $\mathcal{C}$ to obtain, in $O(n^2)$ time, a 2-connected $O(n)$-vertex flat c-graph $\mathcal{C}'(G',\mathcal{T}')$ equivalent to $\mathcal{C}$ and a corresponding carving decomposition (D',γ') of width $\omega' \leq \max(\omega,4)$. Then, we apply either Theorem 3.2 or Theorem 3.3 to test whether $\mathcal{C}'$ (and thus $\mathcal{C}$) is c-planar depending on whether $\mathcal{C}$ is flat or non-flat, respectively. The running time follows from the running time of Theorem 3.2 and 3.3, from the fact that $\omega' = O(\omega)$, $|V(G')| \in O(n)$, and that a carving decomposition of $\delta(G)$ of optimal width ω can be computed in $O(\hat{g}(\omega)n)$ time [71], where $\hat{g}$ is a computable function. This concludes the proof of the theorems. □

4 Graph-Width Parameters Related to the Dual Carving-Width

In this section, we discuss implications of our algorithm for instances of bounded embedded-width and of bounded dual cut-width.

Embedded-width. A tree decomposition of an embedded graph G *respects* the embedding of G if, for every face f of G, at least one bag contains all the vertices of f [17]. The *embedded-width* $\mathrm{emw}(G)$ of G is the minimum width of any of its tree decompositions that respect the embedding of G. For consistency with other graph-width parameters, in the original definition of this width measure [17] the vertices of the outer face are not required to be in some bag. Here, we adopt the variant presented in [34], where the tree decomposition must also include a bag containing the outer face. We have the following.

Lemma 4.1 *Let G be an embedded graph. Then,* $\mathrm{cw}(\delta(G)) \leq \mathrm{emw}^2(G) + 2\,\mathrm{emw}(G)$.

Proof Recall that the dual $\delta(G)$ of G has maximum degree $\ell(G)$ and maximum face size $\Delta(G)$, where $\ell(G)$ and $\Delta(G)$ are the maximum face size and the maximum degree of the graph G, respectively. It is well-known that the tree-width $\mathrm{tw}(G)$ of G and the tree-width $\mathrm{tw}(\delta(G))$ of $\delta(G)$ satisfy the relation: $\mathrm{tw}(\delta(G)) \leq \mathrm{tw}(G) + 1$ [18]. Also, for any graph H, the carving-width $\mathrm{cw}(H)$ of H satisfies the relation: $\mathrm{cw}(H) \leq \Delta(H)(\mathrm{tw}(H) + 1)$ [14]. Therefore, we have $\mathrm{cw}(\delta(G)) \leq \ell(G)(\mathrm{tw}(G) + 2)$. Finally, the embedded-width $\mathrm{emw}(G)$ of G satisfies the relations: $\mathrm{emw}(G) \geq \ell(G)$ and $\mathrm{emw}(G) \geq \mathrm{tw}(G)$, by definition [17, 34]. Combining the above inequalities, we get the stated bound for the carving-width of $\delta(G)$. □

Cut-width. Let π be a linear order of the vertex set of a graph $G = (V, E)$. By splitting π into two linear orders π_1 and π_2 such that π is the concatenation of π_1 and π_2, we define a *cut* of π. The *width* of this cut is the number of edges between a vertex in π_1 and a vertex in π_2. The *width* of π is the maximum width over all its possible cuts. Finally, the *cut-width* of G is the minimum width over all the possible linear orders of V. The *dual cut-width* is the cut-width of the dual of G.

The following relationship between cut-width and carving-width has been proved in [68].

Theorem 4.1 (Theorem 4.3, [68]) *The carving-width of G is at most twice its cut-width.*

By Lemma 4.1 and Theorem 4.1, we have that single-parameter FPT and XP algorithms also exist with respect to the embedded-width and to the dual cut-width of the underlying graph for flat and non-flat c-graphs, respectively.

5 Conclusions

In this paper, we studied the C-Planarity Testing problem for c-graphs with a prescribed combinatorial embedding. We showed that the problem is polynomial-time solvable when the dual carving-width of the underlying graph of the input c-graph is bounded. In particular, we provided a fixed-parameter tractable and a slice-wise polynomial algorithm for the C-Planarity Testing problem for embedded flat and non-flat c-graphs, respectively. This also addresses a question we posed in [34], regarding the existence of notable graph-width parameters such that the C-Planarity Testing problem for embedded flat c-graphs is fixed-parameter tractable with respect to a single one of them. Namely, we answer this question in the affirmative when the parameters are the embedded-width of the underlying graph, and the carving-width and cut-width of its planar dual.

Funding Open access funding provided by Università degli Studi Roma Tre within the CRUI-CARE Agreement.

References

1. Adler, I., Bui-Xuan, B., Rabinovich, Y., Renault, G., Telle, J.A., Vatshelle, M.: On the boolean-width of a graph: structure and applications. In: D.M. Thilikos (ed.) WG 2010, LNCS, vol. 6410, pp. 159–170 (2010). https://doi.org/10.1007/978-3-642-16926-7_16
2. Akitaya, H.A., Fulek, R., Tóth, C.D.: Recognizing weak embeddings of graphs. In: A. Czumaj (ed.) SODA '18, pp. 274–292. SIAM (2018). https://doi.org/10.1137/1.9781611975031.20
3. Angelini, P., Da Lozzo, G.: SEFE = C-planarity? Comput. J. **59**(12), 1831–1838 (2016). https://doi.org/10.1093/comjnl/bxw035
4. Angelini, P., Da Lozzo, G.: Clustered planarity with pipes. Algorithmica **81**(6), 2484–2526 (2019). https://doi.org/10.1007/s00453-018-00541-w
5. Angelini, P., Da Lozzo, G.: Beyond clustered planar graphs. In: S. Hong, T. Tokuyama (eds.) Beyond Planar Graphs, Communications of NII Shonan Meetings, pp. 211–235. Springer (2020). https://doi.org/10.1007/978-981-15-6533-5_12
6. Angelini, P., Da Lozzo, G., Di Battista, G., Frati, F.: Strip planarity testing for embedded planar graphs. Algorithmica **77**(4), 1022–1059 (2017). https://doi.org/10.1007/s00453-016-0128-9
7. Angelini, P., Da Lozzo, G., Di Battista, G., Frati, F., Patrignani, M., Roselli, V.: Relaxing the constraints of clustered planarity. Comput. Geom. **48**(2), 42–75 (2015). https://doi.org/10.1016/j.comgeo.2014.08.001
8. Angelini, P., Da Lozzo, G., Di Battista, G., Frati, F., Patrignani, M., Rutter, I.: Intersection-link representations of graphs. J. Graph Algorithms Appl. **21**(4), 731–755 (2017). https://doi.org/10.7155/jgaa.00437
9. Angelini, P., Da Lozzo, G., Di Battista, G., Frati, F., Roselli, V.: The importance of being proper: (in clustered-level planarity and T-level planarity). Theor. Comput. Sci. **571**, 1–9 (2015). https://doi.org/10.1016/j.tcs.2014.12.019
10. Angelini, P., Frati, F., Kaufmann, M.: Straight-line rectangular drawings of clustered graphs. Discrete Comput. Geom. **45**(1), 88–140 (2011). https://doi.org/10.1007/s00454-010-9302-z
11. Athenstädt, J.C., Cornelsen, S.: Planarity of overlapping clusterings including unions of two partitions. J. Graph Algorithms Appl. **21**(6), 1057–1089 (2017). https://doi.org/10.7155/jgaa.00450
12. Athenstädt, J.C., Hartmann, T., Nöllenburg, M.: Simultaneous embeddability of two partitions. In: C.A. Duncan, A. Symvonis (eds.) Graph Drawing—22nd International Symposium, GD 2014, Würzburg, Germany, September 24–26, 2014, Revised Selected Papers, Lecture Notes in Computer Science, vol. 8871, pp. 64–75. Springer (2014). https://doi.org/10.1007/978-3-662-45803-7_6
13. Biedl, T.: Drawing planar partitions III: Two Constrained Embedding Problems. Tech. Report RRR 13-98, Rutcor Research Report (1998)
14. Biedl, T.C., Vatshelle, M.: The point-set embeddability problem for plane graphs. Int. J. Comput. Geometry Appl. **23**(4–5), 357–396 (2013). https://doi.org/10.1142/S0218195913600091
15. Bixby, R.E., Wagner, D.K.: An almost linear-time algorithm for graph realization. Math. Oper. Res. **13**(1), 99–123 (1988). https://doi.org/10.1287/moor.13.1.99
16. Bläsius, T., Rutter, I.: A new perspective on clustered planarity as a combinatorial embedding problem. Theor. Comput. Sci. **609**, 306–315 (2016). https://doi.org/10.1016/j.tcs.2015.10.011
17. Borradaile, G., Erickson, J., Le, H., Weber, R.: Embedded-width: a variant of treewidth for plane graphs (2017). arXiv:1703.07532
18. Bouchitté, V., Mazoit, F., Todinca, I.: Treewidth of planar graphs: connections with duality. ENDM **10**, 34–38 (2001). https://doi.org/10.1016/S1571-0653(04)00353-1

19. Brandenburg, F., Eppstein, D., Goodrich, M.T., Kobourov, S.G., Liotta, G., Mutzel, P.: Selected open problems in graph drawing. In: G. Liotta (ed.) GD '03, *LNCS*, vol. 2912, pp. 515–539. Springer (2003). https://doi.org/10.1007/978-3-540-24595-7_55
20. Brandes, U., Cornelsen, S., Pampel, B., Sallaberry, A.: Blocks of hypergraphs-applied to hypergraphs and outerplanarity. In: C.S. Iliopoulos, W.F. Smyth (eds.) Combinatorial Algorithms—21st International Workshop, IWOCA 2010, London, UK, July 26–28, 2010, Revised Selected Papers, Lecture Notes in Computer Science, vol. 6460, pp. 201–211. Springer (2010). https://doi.org/10.1007/978-3-642-19222-7_21
21. Brandes, U., Cornelsen, S., Pampel, B., Sallaberry, A.: Path-based supports for hypergraphs. J. Discrete Algorithms **14**, 248–261 (2012). https://doi.org/10.1016/j.jda.2011.12.009
22. Brandes, U., Lerner, J.: Visual analysis of controversy in user-generated encyclopedias. In: IEEE VAST '07, pp. 179–186. IEEE Computer Society (2007). https://doi.org/10.1109/VAST.2007.4389012
23. Buchin, K., van Kreveld, M.J., Meijer, H., Speckmann, B., Verbeek, K.: On planar supports for hypergraphs. J. Graph Algorithms Appl. **15**(4), 533–549 (2011). https://doi.org/10.7155/jgaa.00237
24. Carmesin, J.: Embedding simply connected 2-complexes in 3-space—v. A refined kuratowski-type characterisation (2017)
25. Chimani, M., Di Battista, G., Frati, F., Klein, K.: Advances on testing c-planarity of embedded flat clustered graphs. In: C.A. Duncan, A. Symvonis (eds.) GD '14, LNCS, vol. 8871, pp. 416–427. Springer (2014). https://doi.org/10.1007/978-3-662-45803-7_35
26. Chimani, M., Klein, K.: Shrinking the search space for clustered planarity. In: W. Didimo, M. Patrignani (eds.) GD '12, LNCS, vol. 7704, pp. 90–101. Springer (2012). https://doi.org/10.1007/978-3-642-36763-2_9
27. Cornelsen, S., Wagner, D.: Completely connected clustered graphs. J. Discrete Algorithms **4**(2), 313–323 (2006). https://doi.org/10.1016/j.jda.2005.06.002
28. Cortese, P.F., Di Battista, G.: Clustered planarity. In: J.S.B. Mitchell, G. Rote (eds.) SoCG '05, pp. 32–34. ACM (2005). https://doi.org/10.1145/1064092.1064093
29. Cortese, P.F., Di Battista, G., Frati, F., Patrignani, M., Pizzonia, M.: C-planarity of c-connected clustered graphs. J. Graph Algorithms Appl. **12**(2), 225–262 (2008)
30. Cortese, P.F., Di Battista, G., Patrignani, M., Pizzonia, M.: On embedding a cycle in a plane graph. Discret. Math. **309**(7), 1856–1869 (2009). https://doi.org/10.1016/j.disc.2007.12.090
31. Cortese, P.F., Patrignani, M.: Clustered planarity = flat clustered planarity. In: T.C. Biedl, A. Kerren (eds.) GD 2018, LNCS, vol. 11282, pp. 23–38. Springer (2018). https://doi.org/10.1145/1064092.1064093
32. Courcelle, B., Engelfriet, J., Rozenberg, G.: Handle-rewriting hypergraph grammars. J. Comput. Syst. Sci. **46**(2), 218–270 (1993). https://doi.org/10.1007/978-3-030-04414-5_2
33. Da Lozzo, G., Di Battista, G., Frati, F., Patrignani, M.: Computing nodetrix representations of clustered graphs. J. Graph Algorithms Appl. **22**(2), 139–176 (2018). https://doi.org/10.7155/jgaa.00461
34. Da Lozzo, G., Eppstein, D., Goodrich, M.T., Gupta, S.: Subexponential-time and FPT algorithms for embedded flat clustered planarity. In: A. Brandstädt, E. Köhler, K. Meer (eds.) WG 2018, LNCS, vol. 11159, pp. 111–124. Springer (2018). https://doi.org/10.1007/978-3-030-00256-5_10
35. Da Lozzo, G., Eppstein, D., Goodrich, M.T., Gupta, S.: C-planarity testing of embedded clustered graphs with bounded dual carving-width. In: B.M.P. Jansen, J.A. Telle (eds.) 14th International Symposium on Parameterized and Exact Computation, IPEC 2019, September 11–13, 2019, Munich, Germany, LIPIcs, vol. 148, pp. 9:1–9:17. Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2019). https://doi.org/10.4230/LIPIcs.IPEC.2019.9
36. Dahlhaus, E.: A linear time algorithm to recognize clustered graphs and its parallelization. In: C.L. Lucchesi, A.V. Moura (eds.) LATIN '98, LNCS, vol. 1380, pp. 239–248. Springer (1998). https://doi.org/10.1007/BFb0054325
37. Di Battista, G., Didimo, W.: Gdtoolkit. In: Tamassia, R. (ed.) Handbook on Graph Drawing and Visualization, pp. 571–597. Chapman and Hall/CRC, London (2013)
38. Di Battista, G., Didimo, W., Marcandalli, A.: Planarization of clustered graphs. In: P. Mutzel, M. Jünger, S. Leipert (eds.) GD '01, LNCS, vol. 2265, pp. 60–74. Springer (2001). https://doi.org/10.1007/3-540-45848-4_5
39. Di Battista, G., Frati, F.: Efficient c-planarity testing for embedded flat clustered graphs with small faces. J. Graph Algorithms Appl. **13**(3), 349–378 (2009)

40. Didimo, W., Giordano, F., Liotta, G.: Overlapping cluster planarity. J. Graph Algorithms Appl. **12**(3), 267–291 (2008)
41. Feng, Q., Cohen, R.F., Eades, P.: Planarity for clustered graphs. In: P.G. Spirakis (ed.) ESA'95, LNCS, vol. 979, pp. 213–226. Springer (1995). https://doi.org/10.1007/3-540-60313-1_145
42. Forster, M., Bachmaier, C.: Clustered level planarity. In: P. van Emde Boas, J. Pokorný, M. Bieliková, J. Stuller (eds.) SOFSEM '04, LNCS, vol. 2932, pp. 218–228. Springer (2004). https://doi.org/10.1007/978-3-540-24618-3_18
43. Fulek, R., Kyncl, J.: Hanani-tutte for approximating maps of graphs. In: B. Speckmann, C.D. Tóth (eds.) SoCG '18, LIPIcs, vol. 99, pp. 39:1–39:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2018). https://doi.org/10.4230/LIPIcs.SoCG.2018.39
44. Fulek, R., Kyncl, J., Malinovic, I., Pálvölgyi, D.: Efficient c-planarity testing algebraically. CoRR **abs/1305.4519** (2013). arXiv:1305.4519
45. Fulek, R., Kyncl, J., Malinovic, I., Pálvölgyi, D.: Clustered planarity testing revisited. Electr. J. Comb. **22**(4), P4.24 (2015)
46. Fulek, R., Tóth, C.D.: Atomic embeddability, clustered planarity, and thickenability. CoRR **abs/1907.13086** (2019). arXiv:1907.13086
47. Fulek, R., Tóth, C.D.: Atomic embeddability, clustered planarity, and thickenability. In: S. Chawla (ed.) Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5–8, 2020, pp. 2876–2895. SIAM (2020). https://doi.org/10.1137/1.9781611975994.175
48. Godsil, C.D., Royle, G.F.: Algebraic Graph Theory. Graduate texts in Mathematics. Springer, Berlin (2001). https://doi.org/10.1007/978-1-4613-0163-9
49. Goodrich, M.T., Lueker, G.S., Sun, J.Z.: C-planarity of extrovert clustered graphs. In: P. Healy, N.S. Nikolov (eds.) GD '05, LNCS, vol. 3843, pp. 211–222. Springer (2005). https://doi.org/10.1007/11618058_20
50. Grimaldi, R.: Fibonacci and Catalan Numbers: An Introduction. Wiley, London (2012)
51. Gu, Q., Tamaki, H.: Optimal branch-decomposition of planar graphs in O(n^3) time. ACM Trans. Algorithms **4**(3), 30:1-30:13 (2008). https://doi.org/10.1145/1367064.1367070
52. Gutwenger, C., Jünger, M., Leipert, S., Mutzel, P., Percan, M., Weiskircher, R.: Advances in c-planarity testing of clustered graphs. In: S.G. Kobourov, M.T. Goodrich (eds.) GD '02, LNCS, vol. 2528, pp. 220–235. Springer (2002). https://doi.org/10.1007/3-540-36151-0_21
53. Gutwenger, C., Mutzel, P., Schaefer, M.: Practical experience with hanani-tutte for testing c-planarity. In: C.C. McGeoch, U. Meyer (eds.) ALENEX '14, pp. 86–97. SIAM (2014). https://doi.org/10.1137/1.9781611973198.9
54. Hong, S., Nagamochi, H.: Convex drawings of hierarchical planar graphs and clustered planar graphs. J. Discrete Algorithms **8**(3), 282–295 (2010). https://doi.org/10.1016/j.jda.2009.05.003
55. Hong, S.H., Nagamochi, H.: Simpler algorithms for testing two-page book embedding of partitioned graphs. Theoretical Computer Science (2016)
56. Hopcroft, J.E., Tarjan, R.E.: Efficient algorithms for graph manipulation [H] (algorithm 447). Commun. ACM **16**(6), 372–378 (1973). https://doi.org/10.1145/362248.362272
57. Jelínek, V., Jelínková, E., Kratochvíl, J., Lidický, B.: Clustered planarity: embedded clustered graphs with two-component clusters. In: I.G. Tollis, M. Patrignani (eds.) GD '08, LNCS, vol. 5417, pp. 121–132. Springer (2008). https://doi.org/10.1007/978-3-642-00219-9_13
58. Jelínková, E., Kára, J., Kratochvíl, J., Pergel, M., Suchý, O., Vyskocil, T.: Clustered planarity: small clusters in cycles and Eulerian graphs. J. Graph Algorithms Appl. **13**(3), 379–422 (2009)
59. Johnson, D.S., Pollak, H.O.: Hypergraph planarity and the complexity of drawing venn diagrams. J. Graph Theory **11**(3), 309–325 (1987). https://doi.org/10.1002/jgt.3190110306
60. Kamada, T., Kawai, S.: A general framework for visualizing abstract objects and relations. ACM Trans. Graph. **10**(1), 1–39 (1991). https://doi.org/10.1145/99902.99903
61. Kaufmann, M., van Kreveld, M.J., Speckmann, B.: Subdivision drawings of hypergraphs. In: I.G. Tollis, M. Patrignani (eds.) Graph Drawing, 16th International Symposium, GD 2008, Heraklion, Crete, Greece, September 21–24, 2008. Revised Papers, Lecture Notes in Computer Science, vol. 5417, pp. 396–407. Springer (2008). https://doi.org/10.1007/978-3-642-00219-9_39
62. Nagamochi, H., Kuroya, K.: Drawing c-planar biconnected clustered graphs. Discret. Appl. Math. **155**(9), 1155–1174 (2007). https://doi.org/10.1016/j.dam.2006.04.044
63. Niggemann, O.: Visual data mining of graph based data. Ph.D. thesis, University of Paderborn, Germany (2001). http://ubdata.uni-paderborn.de/ediss/17/2001/niggeman/disserta.pdf

64. Oum, S., Seymour, P.D.: Approximating clique-width and branch-width. J. Comb. Theory Ser. B **96**(4), 514–528 (2006). https://doi.org/10.1016/j.jctb.2005.10.006
65. Paiva, R., Rodrigues, G.N., Bonifácio, R., Ladeira, M.: Exploring the combination of software visualization and data clustering in the software architecture recovery process. In: S. Ossowski (ed.) Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy, April 4–8, 2016, pp. 1309–1314. ACM (2016). https://doi.org/10.1145/2851613.2851765
66. Robertson, N., Seymour, P.D.: Graph minors. X. Obstructions to tree-decomposition. J. Comb. Theory. Ser. B **52**(2), 153–190 (1991). https://doi.org/10.1016/0095-8956(91)90061-N
67. Rué, J., Sau, I., Thilikos, D.M.: Dynamic programming for graphs on surfaces. ACM Trans. Algorithms **10**(2), 8:1-8:26 (2014). https://doi.org/10.1145/2556952
68. Sasák, R.: Comparing 17 graph parameters. Master's thesis, Department of Informatics, University of Bergen, Bergen, Norway (2010)
69. Schaefer, M.: Toward a theory of planarity: hanani–tutte and planarity variants. J. Graph Algorithms Appl. **17**(4), 367–440 (2013). https://doi.org/10.7155/jgaa.00298
70. Seymour, P.D., Thomas, R.: Call routing and the ratcatcher. Combinatorica **14**(2), 217–241 (1994). https://doi.org/10.1007/BF01215352
71. Thilikos, D.M., Serna, M.J., Bodlaender, H.L.: Constructive linear time algorithms for small cutwidth and carving-width. In: D.T. Lee, S. Teng (eds.) ISAAC '00, LNCS, vol. 1969, pp. 192–203. Springer (2000). https://doi.org/10.1007/3-540-40996-3_17
72. van Bevern, R., Kanj, I.A., Komusiewicz, C., Niedermeier, R., Sorge, M.: Twins in subdivision drawings of hypergraphs. In: Y. Hu, M. Nöllenburg (eds.) Graph Drawing and Network Visualization—24th International Symposium, GD 2016, Athens, Greece, September 19–21, 2016, Revised Selected Papers, Lecture Notes in Computer Science, vol. 9801, pp. 67–80. Springer (2016). https://doi.org/10.1007/978-3-319-50106-2_6
73. Vial, J.J.B., Da Lozzo, G., Goodrich, M.T.: Computing k-modal embeddings of planar digraphs. In: M.A. Bender, O. Svensson, G. Herman (eds.) ESA 2019, LIPIcs, vol. 144, pp. 17:1–17:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2019). https://doi.org/10.4230/LIPIcs.ESA.2019.17

Authors and Affiliations

Giordano Da Lozzo[1] · David Eppstein[2] · Michael T. Goodrich[2] · Siddharth Gupta[3]

✉ Giordano Da Lozzo
giordano.dalozzo@uniroma3.it

David Eppstein
eppstein@uci.edu

Michael T. Goodrich
goodrich@uci.edu

Siddharth Gupta
siddhart@post.bgu.ac.il

[1] Roma Tre University, Rome, Italy

[2] University of California, Irvine, USA

[3] Ben-Gurion University of the Negev, Beersheba, Israel