Journal of Graph Algorithms and Applications
http://www.cs.brown.edu/publications/jgaa/
vol. 6, no. 1, pp. 27–66 (2002)

Planar Graphs with Topological Constraints

Christoph Dornheim
Institut für Informatik
Albert-Ludwigs-Universität
D-79110 Freiburg, Germany
http://www.informatik.uni-freiburg.de/~dornheim
dornheim@informatik.uni-freiburg.de

Abstract

We address in this paper the problem of constructing embeddings of planar graphs satisfying declarative, user-defined topological constraints. The constraints consist each of a cycle of the given graph and a set of its edges to be embedded inside this cycle and a set of its edges to be embedded outside this cycle. Their practical importance in graph visualization applications is due to the capability of supporting the semantics of graphs. Additionally, embedding algorithms for planar graphs with topological constraints can be combined with planar graph drawing algorithms that transform a given embedding into a topology preserving drawing according to particular drawing conventions and aesthetic criteria.

We obtain the following main results on the planarity problem with topological constraints. Since it turns out to be NP-complete, we develop a polynomial time algorithm for reducing the problem for arbitrary planar graphs to a planarity problem with constraints for biconnected graphs. This allows focussing on biconnected graphs when searching for heuristics or polynomial time subproblems. We then define a particular subproblem by restricting the maximum number of vertices that any two distinct cycles involved in the constraints can have in common. Whereas the problem remains NP-complete if this number exceeds 1, it can otherwise be solved in polynomial time. The embedding algorithm we develop for this purpose is based on the reduction method.

Communicated by H. de Fraysseix and and J. Kratochvíl:
submitted May 2000; revised March 2001.

This research was supported by the *Deutsche Forschungsgemeinschaft* (DFG) as part of the graduate school on *Human and Machine Intelligence*.

1 Introduction

Most research in the graph drawing field has focussed so far on developing algorithms that produce drawings of graphs according to particular drawing conventions and aesthetic criteria (see [5], [6], [11] and [23]). Common drawing conventions require, for instance, the vertices to have integer coordinates, the edges to be polylines or straight lines or to be composed of straight line segments parallel to one of the coordinate axes. Aesthetic criteria are in general optimization goals such as minimizing the number of edge crossings, maximizing the minimum distance between vertices or, in case of polyline drawings, minimizing the number of bends. Both drawing conventions and aesthetic criteria are intended to ensure readability of the drawings. They can be seen as geometric constraints implicit to the drawing algorithms.

Many applications, however, need a visualization of graphs satisfying user-defined constraints to emphasize certain semantical aspects. Visual languages [27], for instance, may require some subgraphs to be drawn with a predefined shape to support their semantics. And in VLSI design [14] it is often necessary to have a graph layout where certain vertices are arranged in a group or cluster [21]. Although "the study of constraints is a strategic research direction for the graph drawing field" [22], there are only few graph drawing approaches allowing the user to define constraints. A survey of these approaches and those drawing algorithms that can be modified to support some sort of constraints is given in [22]. Again, the resulting constraints are primarily geometric and most of them are covered by the approach of [15] which supports the specification of arithmetic linear equality and inequality constraints on the coordinates of the vertices.

These approaches to constrained graph drawing have often the disadvantage of not drawing a planar graph without crossings, i.e. as a planar embedding. For graph planarity without constraints there are many testing and embedding algorithms developed so far: from conceptually simple procedures such as [4] (also briefly described in [1]) to more involved linear time procedures such as the path addition embedding algorithm presented by Mehlhorn and Mutzel [18] based on the planarity testing procedure of Hopcroft and Tarjan [16] or the vertex addition embedding algorithm presented by Chiba et al. [3] based on the planarity testing procedure of Booth and Lueker [2]. However, the problem of constructing a planar embedding according to explicit constraints has obtained only little attention so far, despite its impact to planar graph theory and practice. An interesting approach to constrained graph embedding presented in [19] is the problem of finding a planar embedding such that the user's preferences for facial cycles are satisfied. The preferences, or constraints, are specified by a cost function on the cycles of the graph. In an embedding that satisfies these constraints the overall costs of its facial cycles are minimized. Another approach is the construction of planar embeddings for clustered graphs [12]. A clustered graph is a graph with a user-defined hierarchical structure over the vertices such that each level decomposes the graph into disjoint clusters. In a planar drawing of a clustered graph a cluster is represented by a simple region

containing exactly the vertices that belong to this cluster. Additionally, edges are allowed to intersect the boundaries of the regions only once; in particular, edges connecting vertices of the same cluster must be inside the corresponding region. The embedding algorithm developed in [12] requires the vertices of each cluster to induce a connected graph. As shown in [9] and [10], these embeddings can be further transformed into drawings according to specific drawing conventions while preserving their cluster structure. Since many drawing procedures for planar graphs need a particular embedding (more precisely, a combinatorial representation of an embedding) as input and produce drawings of the same topology, the construction of embeddings satisfying given constraints can be used as a preprocessing step for a wide range of planar graph drawing algorithms.

This paper is devoted to planar graph embedding with topological constraints; it considerably extends the previous work in [8]. We define a topological constraint for a graph to consist of some of its cycle and a set of its edges to be embedded inside this cycle and a set of its edges to be embedded outside this cycle. These constraints are clearly topological since topological transformations of the plane preserve their validity. Note that we build constraints by sets of edges instead of sets of vertices. If some vertex is to be embedded inside or outside a given cycle, the corresponding topological constraint can be defined by using an edge incident to this vertex (if the vertex is isolated, it must be replaced by some new edge for the time of the embedding process). To see that in some situations vertex sets would be insufficient, think of, e.g., Hamilton cycles: the diagonals of a Hamilton cycle can be embedded inside or outside, but there is no vertex left to constitute a topological constraint.

The topological constraints are useful in graph embedding whenever the inner and outer regions of cycles have an intrinsic or user-defined semantics. Graphs whose vertices and edges represent spatial entities such as road maps or subway maps often induce some natural meaning to the faces and regions of their embeddings as well. In this case, topological constraints can be defined to ensure that the embedded graph preserves some important topological relationships. Generally, these constraints can be used to impose some structure on the embedding to be constructed. Consider in Fig. 1 the transition diagram of a finite automaton accepting the language $(\mathtt{a}(\mathtt{cde})^*\mathtt{bz} + \mathtt{f}(\mathtt{hij})^*\mathtt{gz})^*$. The main components of this expression are $\mathtt{a}(\mathtt{cde})^*\mathtt{bz}$ and $\mathtt{f}(\mathtt{hij})^*\mathtt{gz}$, each representing a cyclic walk starting and ending in q_0. Moreover, each component contains a subcomponent $(\mathtt{cde})^*$ and $(\mathtt{hij})^*$ representing cyclic walks starting and ending in q_1 and q_6, respectively. To reflect this structure in the diagram, it makes sense to draw the cycles q_0, q_1, q_7 and q_0, q_6, q_7 representing the main components into disjoint regions, but to draw the cycles q_1, q_3, q_2 and q_6, q_5, q_4 representing the corresponding subcomponents inside q_0, q_1, q_7 and q_0, q_6, q_7, respectively. These requirements can be expressed exactly by topological constraints. Last but not least, a topological constraint allows one to specify a cycle to be an inner or outer facial cycle just by postulating the former or latter set, respectively, to consist of all edges not belonging to that cycle.

The outline of the paper is as follows. Recalling in Sect. 2 the required defi-

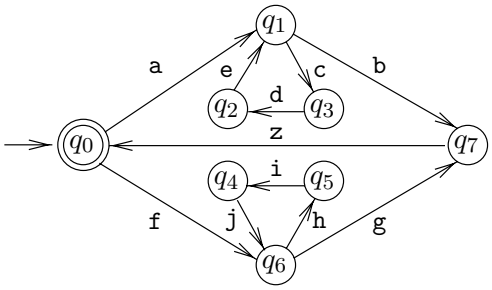

Figure 1: A transition diagram of a finite automaton with the language $(\mathtt{a}(\mathtt{cde})^*\mathtt{bz} + \mathtt{f}(\mathtt{hij})^*\mathtt{gz})^*$. The structure of this expression is reflected in the drawing by the topological relationships of the cycles.

nitions and results of planar graph theory, we show in Sect. 3 NP-completeness of the problem of deciding whether a graph has an embedding satisfying a given set of topological constraints. In contrast, we present a linear time embedding algorithm for triconnected graphs. Section 4 is concerned with developing a polynomial time procedure for reducing the embedding problem with topological constraints to biconnected graphs. This can be particularly useful in searching for tractable subproblems. We introduce in Sect. 5 a particular subproblem defined by the maximum number of vertices two distinct cycles involved in the constraints have in common. Whereas this problem remains NP-complete if this number exceeds 1, we develop a polynomial time algorithm for generating a required embedding if this number is 0 or 1. Finally, we conclude with Sect. 6.

2 Preliminaries

Assuming familiarity with basic graph terminology (see [1] and [7]), we recall in this section the required fundamentals on planar graphs. For a thorough presentation of planar graph theory we refer the reader to [24], [7] and [20].

Unless otherwise stated, all graphs are finite, simple and undirected. The set of *vertices* and *edges* of a graph G are denoted $V(G)$ and $E(G)$, respectively. Paths and cycles in a graph are always simple, i.e. their inner vertices are pairwise distinct. A cycle C in G is *induced* if any edge of G joining two vertices of C is already an edge in C. Moreover, C is *non-separating* if the number of components of G does not increase when deleting C from G. A *block* means a maximal connected subgraph without a cutvertex, i.e. a maximal 2-connected subgraph or a bridge or an isolated vertex. If A denotes the set of cutvertices of G and $\mathcal{B}$ denotes the set of its blocks, then the *block graph* of G is defined on $A \cup \mathcal{B}$ by the edges aB with $a \in B$ for $a \in A$ and $B \in \mathcal{B}$. Note that the block graph of a connected graph is always a tree.

A *plane graph* (V, E) consists of a finite set $V \subseteq \mathbb{R}^2$ and a finite set E of Jordan curves connecting two points of V such that any two points are connected

by at most one curve and the interior of a curve contains no point of V and no point of any other curve. These points and curves are again called *vertices* and *edges*, respectively, since all graph notions can also be applied to plane graphs. If there is an isomorphism between an abstract graph G and a plane graph $\widetilde{G}$, we call $\widetilde{G}$ an *embedding* of G in the plane. A graph is *planar* if it has an embedding in the plane. In the sequel, $\widetilde{G}$ always denotes an embedding of the graph G and for subgraphs $H \subset G$ we often denote the restriction of $\widetilde{G}$ to H simply by $\widetilde{H}$.

Every plane graph G divides $\mathbb{R}^2 \setminus G$ into disjoint open arcwise connected regions called *faces* of G; the unbounded one is the *outer face* and the bounded ones are the *inner faces*. If a cycle C of G is the boundary of some face, we call C a *facial cycle*; depending on whether C is the boundary of an inner or outer face, we call C an *inner* or *outer facial cycle*, respectively. Any cycle C divides $\mathbb{R}^2 \setminus C$ into two open arcwise connected regions; the bounded one is the *interior* of C, denoted by *int* C, and the unbounded one is the *exterior* of C, denoted by *ext* C. The closure of int C and ext C is denoted by *Int* C and *Ext* C, respectively.

Given a graph $G = (V, E)$ the *edge space* $\mathcal{E}(G)$ of G is the vector space over the 2-element field $\mathbb{F}_2 = \{0, 1\}$ formed by the set of all subsets of E and the sum $A \oplus B = (A \setminus B) \cup (B \setminus A)$ for $A, B \subseteq E$. The *cycle space* $\mathcal{C}(G)$ is the subspace of $\mathcal{E}(G)$ generated by the cycles of G – more precisely, by their edge sets. The elements of $\mathcal{C}(G)$ are unions of edge-disjoint cycles of G. Suppose $\widetilde{G}$ is an embedding of a 2-connected planar graph G and $\mathcal{F}'$ is the set of all facial cycles of $\widetilde{G}$. Then, as follows from MacLane's planarity criterion [17], $\mathcal{F} = \mathcal{F}' \setminus \{F\}$ forms a basis of $\mathcal{C}(G)$ for any $F \in \mathcal{F}'$ and F is the sum of all cycles in $\mathcal{F}$. There is always an embedding of G where $\mathcal{F}$ is the set of inner facial cycles and F the outer facial cycle. We often make use of the fact that the edges of $\widetilde{G}$ contained in $\operatorname{Int} \widetilde{C}$ for some cycle $C \subset G$ are exactly the edges of those inner facial cycles of $\widetilde{G}$ whose sum is C. Moreover, any two embeddings of G are topologically equivalent if they coincide in their outer facial cycle and their set of inner facial cycles (see [7]). We can thus represent a particular 2-connected embedding up to topological equivalence just by its outer facial cycle and the set of its inner facial cycles. If in addition G is 3-connected, the facial cycles of $\widetilde{G}$ can be identified combinatorially as the induced non-separating cycles in G, as proved by Tutte [26]. This allows representing any embedding of G just by its outer facial cycle because all these embeddings have the same facial cycles.

Finally, we introduce a notion that plays a fundamental role in graph embedding algorithms following the principle of path addition. Suppose H is a subgraph of G. Then an *H-component* of G is either an edge (together with its ends) in $E(G) \setminus E(H)$ joining two vertices of H or it is a connected component of $G - H$ together with all edges (and their ends) of G joining this component to H.[1] The vertices of an H-component in H are its *vertices of attachment*. Suppose C is a cycle in G. Vertices of attachment of a C-component, provided that there are at least two, divide C into edge-disjoint paths, called *segments*. We say

[1] This definition follows [24]. One can find elsewhere in the literature various notions for H-components, e.g. *bridges* in [1].

that two C-components *avoid* each other if either one of them has fewer than two vertices of attachment or all vertices of attachment of one C-component lie in a single segment of the other C-component; otherwise they *overlap*. The *overlap graph* G_C of C in G is defined as the graph whose vertices are the C-components of G such that two vertices are adjacent iff the corresponding C-components overlap. Overlap graphs can be used even for characterizing planar graphs: by Tutte's theorem [25] a graph is planar iff the overlap graph of each of its cycle is bipartite.

3 Topological Constraints

In this section we introduce topological constraints and present a criterion for graphs having an embedding that satisfies such constraints. We also show some basic properties of the computational problem of finding these embeddings: whereas this problem is in general NP-complete, it can be solved in linear time for 3-connected graphs.

Definition 1 *Let G be a graph and $\mathcal{T}$ be a set of triples $T_i = \langle C_i, In_i, Out_i\rangle$ such that C_i is a cycle in G, $In_i, Out_i \subset E(G) \setminus E(C_i)$, $In_i \cup Out_i \neq \emptyset$ and $In_i \cap Out_i = \emptyset$. These triples are called* topological constraints *for G.*

An embedding $\widetilde{G}$ of G satisfies $\mathcal{T}$ if for each constraint $T_i \in \mathcal{T}$ all edges of In_i are contained in int $\widetilde{C_i}$ and all edges of Out_i are contained in ext $\widetilde{C_i}$; such an embedding is called a $\mathcal{T}$-embedding. *Moreover, G is* $\mathcal{T}$-planar *if G has a $\mathcal{T}$-embedding.*

Obviously, $\mathcal{T}$-planar graphs are necessarily planar; thus we only consider planar graphs. Given a planar graph G and a set $\mathcal{T}$ of topological constraints for G, n always denotes the number of vertices of G, m denotes the number of edges where $m \leq 3n-6$ and t denotes the number of constraints in $\mathcal{T}$. Note that t can be exponential in n. The relevant parameters of the problem of finding a $\mathcal{T}$-embedding of G are therefore n and t. Since each constraint T_i consists of at most m edges, the input length of any instance can be assumed to be $O(nt)$. For convenience, if C is a cycle C_i occurring in some constraint, we often write In_C and Out_C instead of In_i and Out_i, respectively.

Due to the additional requirements on embeddings, one cannot hope for a simple Kuratowski-like criterion for $\mathcal{T}$-planarity. In contrast to classical planarity, $\mathcal{T}$-planarity of a graph cannot be simply reduced to its components or blocks either; even the required topological relationships between its components can get complex as we will see in the next section. To characterize $\mathcal{T}$-planarity in Theorem 1, we make use of some properties of 3-connected planar graphs given in Sect. 2. This theorem is illustrated in Fig. 2.

Theorem 1 *Let G be a graph and $\mathcal{T}$ be a set of topological constraints. Then, G is $\mathcal{T}$-planar iff there is a 3-connected planar graph G' with $G \subseteq G'$ and $V(G) = V(G')$ and a basis $\mathcal{F}$ of $\mathcal{C}(G')$ consisting of induced non-separating cycles of G' such that for any $\langle C, In_C, Out_C\rangle \in \mathcal{T}$ and $\mathcal{F}_C \subseteq \mathcal{F}$ defined by*

$C = \bigoplus_{F \in \mathcal{F}_C} F$ *it holds that* $x \in \bigcup_{F \in \mathcal{F}_C} F$ *for any* $x \in In_C$ *and* $y \notin \bigcup_{F \in \mathcal{F}_C} F$ *for any* $y \in Out_C$.

Proof: ($\Rightarrow$) Suppose $\widetilde{G}$ is a $\mathcal{T}$-embedding of G. Triangulating $\widetilde{G}$ in the plane leads to an embedding $\widetilde{G'}$ of a maximal planar and thus 3-connected graph G'. The inner facial cycles of $\widetilde{G'}$ are induced non-separating cycles and form a basis $\mathcal{F}$ of $\mathcal{C}(G')$. Let be $\langle C, \mathrm{In}_C, \mathrm{Out}_C \rangle \in \mathcal{T}$ and $\mathcal{F}_C$ be the set of cycles in $\mathcal{F}$ whose sum is C. Since any edge of In_C is contained in Int $\widetilde{C}$ of $\widetilde{G}$, and thus of $\widetilde{G'}$, it belongs to some cycle in $\mathcal{F}_C$. Analogously, any edge of Out_C is contained in Ext $\widetilde{C}$ of $\widetilde{G'}$ and therefore does not belong to some cycle of $\mathcal{F}_C$.

($\Leftarrow$) Let G' be the supergraph of G satisfying the required properties. Then there is an embedding $\widetilde{G'}$ of G' with $\bigoplus_{F \in \mathcal{F}} F$ as the outer facial cycle and all $F \in \mathcal{F}$ as inner facial cycles. For any $\langle C, \mathrm{In}_C, \mathrm{Out}_C \rangle \in \mathcal{T}$ we have that all edges of In_C are contained in Int $\widetilde{C}$ in $\widetilde{G'}$ and all edges of Out_C are contained in Ext $\widetilde{C}$ in $\widetilde{G'}$. Thus, we get a $\mathcal{T}$-embedding $\widetilde{G}$ by restricting $\widetilde{G'}$ to G. □

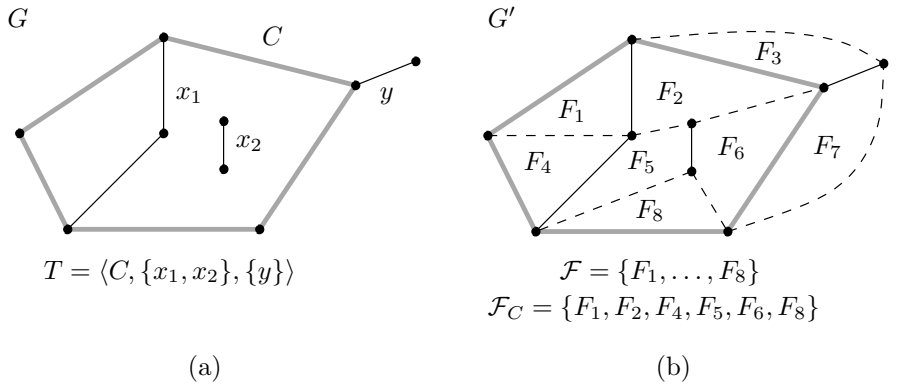

Figure 2: Illustration of Theorem 1: (a) Graph G with constraint T. (b) Supergraph G' of G with basis $\mathcal{F}$ of $\mathcal{C}(G)$ and set $\mathcal{F}_C$ of those cycles in $\mathcal{F}$ whose sum is C.

In case of 3-connected graphs, this criterion leads directly to an algorithm for finding a $\mathcal{T}$-embedding. To ensure linear time complexity, we use dual graphs to represent the relationships between facial cycles (see [7]). Suppose G is a 3-connected planar graph and $\mathcal{F}$ the unique set of facial cycles of some embedding of G. Any two facial cycles have at most one edge in common and each edge belongs to exactly two facial cycles. We define the dual graph G^* on $\mathcal{F}$ such that two facial cycles are adjacent iff they share an edge. Since edges of G are in a one-to-one correspondence to edges of G^*, let C^* denote the set of edges in G^* that correspond to the edges of $C \subseteq E(G)$. G^* has the interesting property that $C \subseteq E(G)$ is a cycle in G iff C^* is a minimal cut in G^*, i.e. removing C^* from G^* separates G^* into two components $U^*, W^* \subset G^*$. These components satisfy $C = \bigoplus_{F \in U^*} F = \bigoplus_{F \in W^*} F$. Thus, in any embedding of G U^* is the

set of facial cycles inside C and W^* is the set of facial cycles outside C, or vice versa. Figure 3 illustrates the relationship between G and G^*.

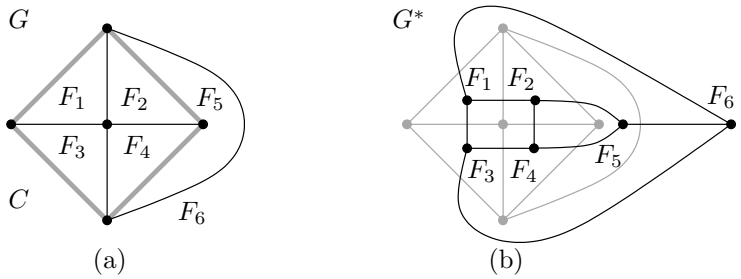

Figure 3: (a) A 3-connected planar graph G with a cycle C and unique faces $F_1, \ldots, F_6$. (b) The dual Graph G^* of G with minimal cut $C^* = \{F_1F_6, F_2F_5, F_4F_5, F_3F_6\}$ corresponding to C and the sets $U^* = \{F_1, F_2, F_3, F_4\}$, $W^* = \{F_5, F_6\}$.

To find a $\mathcal{T}$-embedding, we only have to decide for each C whether U^* or W^* is the set of facial cycles outside C and to search for a facial cycle F_0 belonging to each of these sets. Then F_0 can be chosen as the outer facial cycle. We describe the algorithm in Fig. 4 and prove its correctness in Theorem 2.

Algorithm *embed_triconnected_graph*
Input: G, $\mathcal{T}$
Output: outer facial cycle F_0 of some $\mathcal{T}$-embedding

1. Find the set $\mathcal{F}$ of facial cycles of some embedding of G.
2. Construct the dual graph G^*.
3. For each $\langle C, \mathrm{In}_C, \mathrm{Out}_C\rangle \in \mathcal{T}$ find the components U_C^*, W_C^* of $G^* - C^*$.
   Let be $U_C = \bigcup_{F\in U_C^*} F$ and $W_C = \bigcup_{F\in W_C^*} F$.
   If $\mathrm{In}_C \subset U_C$ and $\mathrm{Out}_C \subset W_C$, define $Z_C^* = W_C^*$;
   else if $\mathrm{In}_C \subset W_C$ and $\mathrm{Out}_C \subset U_C$, define $Z_C^* = U_C^*$; else stop.
4. If there is some cycle $F_0 \in \mathcal{F}$ belonging to each Z_C^*, return F_0.

Figure 4: Embedding algorithm for 3-connected graphs

Theorem 2 *Given a 3-connected graph G and a set $\mathcal{T}$ of topological constraints for G, algorithm* embed_triconnected_graph *generates a $\mathcal{T}$-embedding represented by its outer facial cycle iff G is $\mathcal{T}$-planar. Moreover, it has time complexity $O(nt)$.*

Proof: ($\Rightarrow$) Suppose the algorithm returns some F_0. There is some embedding $\widetilde{G}$ of G with inner facial cycles $\mathcal{F}' = \mathcal{F} \setminus \{F_0\}$ and outer facial cycle F_0. By definition of the Z_C^*, we have for any $\langle C, \mathrm{In}_C, \mathrm{Out}_C\rangle \in \mathcal{T}$ and $\mathcal{F}_C := \mathcal{F}\setminus Z_C^* \subseteq \mathcal{F}'$

that $\bigcup_{F\in\mathcal{F}_C} F$ contains all edges of In_C but no edge of Out_C. Thus, by Theorem 1, $\widetilde{G}$ is a $\mathcal{T}$-embedding.

($\Leftarrow$) Conversely, assume G to have a $\mathcal{T}$-embedding $\widetilde{G}$ with the set $\mathcal{F}'$ of inner facial cycles and outer facial cycle F'_0. We have to show that the algorithm always generates a cycle F_0 as required. Let for any $\langle C, \mathrm{In}_C, \mathrm{Out}_C\rangle \in \mathcal{T}$ $\mathcal{F}_C$ denote the sum of those cycles in $\mathcal{F}'$ that sum up to C. $\mathcal{F}_C$ and $(\mathcal{F}' \cup F'_0) \setminus \mathcal{F}_C$ are exactly the components U^*_C, W^*_C of $G^* - C^*$ and, by Theorem 1, $\bigcup_{F\in\mathcal{F}_C} F$ contains all edges of In_C but no edge of Out_C. Thus, the algorithm defines Z^*_C to be $(\mathcal{F}' \cup F'_0) \setminus \mathcal{F}_C$. At least F'_0 belongs to each Z^*_C such that the algorithm can return some cycle $F_0 \in \bigcap_C Z^*_C$.

Finally, this algorithm can be implemented as an $O(nt)$ procedure: generating the facial cycles of G by embedding G takes $O(n)$ time as well as the construction of the dual graph G^*. Since finding the components U^*_C and W^*_C of $G^* - C^*$ and testing the required conditions can both be done in $O(n)$ time, step 3 takes all in all $O(nt)$ time. The last step also needs only $O(nt)$ time. □

Basically, this polynomial time $\mathcal{T}$-planarity algorithm depends on the simple topological properties of 3-connected planar graphs. The next theorem shows that in the general case we cannot hope for such a polynomial time solution. This also holds for some natural restrictions of the problem.

Theorem 3 *The problem of $\mathcal{T}$-planarity is NP-complete.*

Proof: Membership in NP follows immediately from Theorem 2: we only have to guess a 3-connected planar supergraph G' of G and verify in polynomial time G' to be $\mathcal{T}$-planar.

Next we show the problem to be NP-hard by transformation from the NP-complete problem BETWEENNESS (see [13]) defined as: given a finite set A and a set S of ordered triples of distinct elements from A, is there a one-to-one function $f : A \longrightarrow \{1, \ldots, |A|\}$ such that for each $(a, b, c) \in S$ either $f(a) < f(b) < f(c)$ or $f(c) < f(b) < f(a)$ holds? Let $A = \{a_1, \ldots, a_n\}$ and S be an arbitrary instance of BETWEENNESS. We associate with A the graph G_A defined by $V(G_A) = \{x, y, a_1, \ldots, a_n\}$ and $E(G_A) = \{xa_i, a_iy \mid i \in \{1, \ldots, n\}\}$. For any i, j with $1 \le i < j \le n$ let $C_{i,j}$ denote the cycle (x, a_i, y, a_j, x) illustrated in Fig. 5. Then, S is encoded by a set $\mathcal{T}_S$ of topological constraints for G_A: let $\mathrm{In}_{C_{i,j}}$ be the set $\{xa_k \mid (a_i, a_k, a_j) \in S$ or $(a_j, a_k, a_i) \in S\}$ and $\mathcal{T}_S$ be the set of triples $\langle C_{i,j}, \mathrm{In}_{C_{i,j}}, \emptyset\rangle$ with nonempty $\mathrm{In}_{C_{i,j}}$.

We show that G_A is $\mathcal{T}_S$-planar iff there is some function f satisfying the requirements given above. Suppose $\widetilde{G_A}$ is a $\mathcal{T}_S$-embedding of G_A where a' and a'' are the two vertices of G_A belonging to the outer facial cycle of $\widetilde{G_A}$. We define a linear order f on A by $f(a') = 1$, $f(a'') = n$ and inductively $f(a) = i$ if $a \neq a''$ and a belongs to the outer facial cycle of $\widetilde{G_A}$ where the vertices $f^{-1}(1), \ldots, f^{-1}(i-1)$ are removed. It follows immediately that for each $(a, b, c) \in S$ $f(b)$ is between $f(a)$ and $f(c)$.

Conversely, suppose f has the required property. Then G_A can be embedded while preserving the order of f, i.e. $f^{-1}(i)$ and $f^{-1}(i+1)$ belong to the same

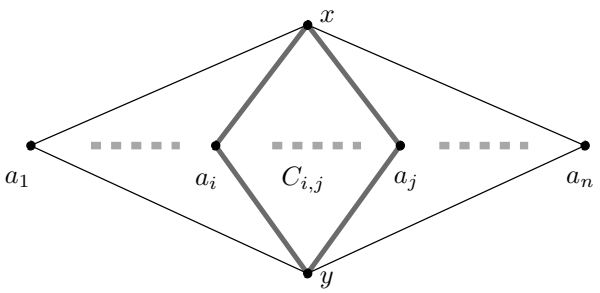

Figure 5: Graph G_A defined in the proof of Theorem 3

inner facial cycle and $f^{-1}(1)$ and $f^{-1}(n)$ belong to the outer facial cycle. Thus, all edges of $\mathrm{In}_{C_{i,j}}$ are inside $C_{i,j}$. □

According to the special properties of G_A and $\mathcal{T}_S$ defined in the proof, we can draw some further conclusions. Since G_A is 2-connected and has treewidth 2, $\mathcal{T}$-planarity remains NP-complete for the class of graphs having these properties. Thus, unlike many other graph problems, a bounded treewidth does not result in a polynomial time solution in case of $\mathcal{T}$-planarity. With respect to the connectivity number κ of graphs, we get the exact boundary between P and NP: the problem is NP-complete for $\kappa \leq 2$ while allowing a polynomial time solution for $\kappa \geq 3$. $\mathcal{T}$-planarity remains also NP-complete for Eulerian graphs since G_A is Eulerian (if $|A|$ is odd, we have to add the edge xy to $E(G_A)$). Interestingly, NP-completeness holds for topological constraints of the form $\langle C, \mathrm{In}_C, \emptyset \rangle$. Since in the proof any $xa_k \in \mathrm{In}_{C_{i,j}}$ can be expressed equivalently by $xa_i \in \mathrm{Out}_{C_{k,j}}$ and $xa_j \in \mathrm{Out}_{C_{i,k}}$, this is also valid for constraints of the form $\langle C, \emptyset, \mathrm{Out}_C \rangle$.

4 Reduction to Biconnected Graphs

As far as classical planarity is concerned, the planarity problem of a graph can be simply reduced to its blocks. Correspondingly, many graph embedding algorithms, e.g. [3] and [18], require the input graph to be 2-connected. The embedded blocks can be easily merged to an embedding of the whole graph. A corresponding reduction for $\mathcal{T}$-planarity, however, must involve all the topological constraints between the components and blocks. In this section we develop a polynomial time algorithm providing such a reduction in two steps: first, $\mathcal{T}$-planarity for a graph is reduced to its components and second, connected graphs with constraints are reduced to their blocks. Thus, it is sufficient to focus on 2-connected graphs when searching for polynomial time subproblems or heuristics to cope with the general NP-complete problem.

4.1 Reducing a Graph to Its Components

Topological constraints for an arbitrary planar graph can considerably differ from those for a 2-connected graph: this is due to the fact that topological constraints between its components or blocks imply yet another sort of constraints not subsumed by Definition 1. Suppose we have two components H and H' of a graph G and three constraints meaning that $x \in E(H')$ must be embedded inside cycles C_1 and C_2, but outside cycle C_3 of H. To find an embedding of G satisfying these constraints, we have to find an embedding $\widetilde{H}$ of H with an inner face inside $\widetilde{C_1}$ and $\widetilde{C_2}$, but outside $\widetilde{C_3}$ in which we can embed H'. Thus, we get a new constraint between cycles of the same component that cannot be equivalently expressed by topological constraints. Although again topological by nature, we call constraints of this sort *overlap constraints*. More formally:

Definition 2 *Let G be a graph and $\mathcal{L}$ be a set of pairs $L_i = \langle C\text{-}In_i, C\text{-}Out_i \rangle$ such that $C\text{-}In_i$ and $C\text{-}Out_i$ are sets of cycles of G. These pairs are called* overlap constraints *for G.*
An embedding $\widetilde{G}$ of G satisfies $\mathcal{L}$ if for each constraint $L_i \in \mathcal{L}$ there is a face of $\widetilde{G}$ in the interior of all cycles of $C\text{-}In_i$ and in the exterior of all cycles of $C\text{-}Out_i$; such an embedding is called an $\mathcal{L}$-embedding. Moreover, G is $\mathcal{L}$-planar if G has an $\mathcal{L}$-embedding.

We only consider overlap constraints that are induced by topological constraints though they might be of general interest. If a graph has an embedding satisfying both topological constraints $\mathcal{T}$ and overlap constraints $\mathcal{L}$, we call it $\mathcal{TL}$-planar and its embedding $\mathcal{TL}$-embedding for short. If H is a subgraph of G, we denote with $\mathcal{T}|_H$ the set of topological constraints which $\mathcal{T}$ induces on H, i.e. $\langle C_i, \mathrm{In}_i \cap E(H), \mathrm{Out}_i \cap E(H) \rangle$ if C_i is a cycle in H and $\langle C_i, \mathrm{In}_i, \mathrm{Out}_i \rangle \in \mathcal{T}$.

Suppose G is a planar graph, $\mathcal{T}$ is a set of topological constraints for G and $\mathcal{H}$ is the set of components of G to which we want to reduce $\mathcal{T}$-planarity. To represent the induced constraints between distinct components $H, H' \in \mathcal{H}$, we define the following sets:

$$\begin{aligned} \text{C-In}_{H,H'} &= \{C \subset H \mid \langle C, \mathrm{In}_C, \mathrm{Out}_C \rangle \in \mathcal{T} \text{ with } \mathrm{In}_C \cap E(H') \neq \emptyset\}, \\ \text{C-Out}_{H,H'} &= \{C \subset H \mid \langle C, \mathrm{In}_C, \mathrm{Out}_C \rangle \in \mathcal{T} \text{ with } \mathrm{Out}_C \cap E(H') \neq \emptyset\}. \end{aligned}$$

The sets $\text{C-In}_{H,H'}$ define moreover a relation $\prec$ on $\mathcal{H}$ by $H' \prec H$ iff $\text{C-In}_{H,H'} \neq \emptyset$. This relation means that we have to find an embedding of H that has a face inside all cycles of $\text{C-In}_{H,H'}$ and outside all cycles of $\text{C-Out}_{H,H'}$, thus satisfying the overlap constraint $\langle \text{C-In}_{H,H'}, \text{C-Out}_{H,H'} \rangle$. However, there are much more overlap constraints for H. Consider the set $\mathcal{H}'$ of all successors of H, i.e. $\{H' \mid H' \prec H\}$, and let K be a component of (the undirected graph of) $\prec$ restricted to $\mathcal{H}'$. Then all $H' \in K$ must lie in the same face of an embedding of H. The stronger overlap constraints for H are thus $\langle \bigcup_{H' \in K} \text{C-In}_{H,H'}, \bigcup_{H' \in K} \text{C-Out}_{H,H'} \rangle$ for all such K.

Now consider an embedding $\widetilde{G}$ of G and the induced ordering $<$ on $\mathcal{H}$ by $H' < H$ iff $\widetilde{G}|_{H'}$ is in some inner face of $\widetilde{G}|_H$. It can be easily seen, that $<$ is

a strict partial order with the additional property that $H' < H$ and $H' < H''$ imply either $H < H''$ or $H = H''$ or $H'' < H$ due to planarity. In other words, $<$ is the transitive closure of a disjoint union of rooted trees. If moreover $\widetilde{G}$ is a $\mathcal{T}$-embedding, the relation $\prec$ defined above is a subrelation of $<$. An example of both relations is given in Fig. 6.[2]

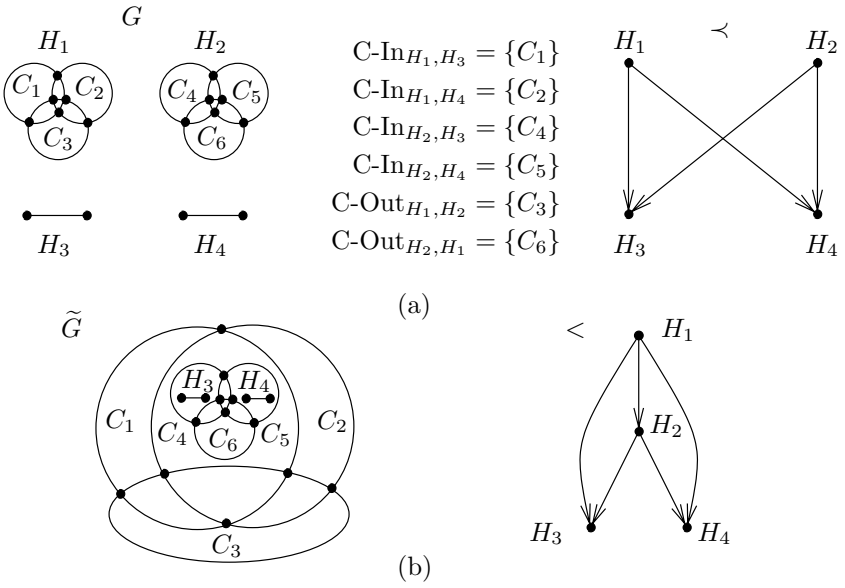

Figure 6: (a) A graph G consisting of components $H_1, \ldots, H_4$ and topological constraints between them represented by the sets C-In$_{H_i,H_j}$ and C-Out$_{H_i,H_j}$; these sets define the relation $\prec$. (b) An embedding $\widetilde{G}$ satisfying these constraints and the relation $<$ induced by $\widetilde{G}$.

The method of reducing $\mathcal{T}$-planarity of G to its components consists just in finding such an extention $<$ of $\prec$ and appropriate embeddings of the components of G. Instead of the strict partial ordering $<$, it is more convenient to construct its transitive reduction which is a disjoint union of rooted trees. When first considering subsets $\mathcal{H}' \subset \mathcal{H}$ such that $\prec|_{\mathcal{H}'}$ is a connected relation, we can easily formulate a recursive procedure *order_components* shown in Fig. 7 for constructing the transitive reduction of $<$ w.r.t. to $\mathcal{H}'$. It makes use of an algorithm A for generating an embedding of a connected graph satisfying given topological and overlap constraints; we also assume that the faces corresponding to the overlap constraints are returned. Such an algorithm A is provided by the reduction to blocks developed in the next section. The procedure *order_components* returns a tree of embedded components in $\mathcal{H}'$ where an edge from $\widetilde{H}$ to $\widetilde{H'}$ is labelled with a face of $\widetilde{H}$ in which $\widetilde{H'}$ must be embedded. Figure 8 shows the only two possible labelled trees that can result from the graph and constraints given in

[2]Note that we visualize relations like $\prec$ by drawing an arrow from H to H' if $H' \prec H$.

Fig. 6 (a). Here we have labelled an edge from $\widetilde{H}$ to $\widetilde{H'}$ by the overlap constraint for $\widetilde{H}$ instead of its corresponding face. The first labelled tree corresponds to the embedding shown in Fig. 6 (b).

Algorithm *order_components*
Input: $\mathcal{H}' \subset \mathcal{H}$ such that $\prec|_{\mathcal{H}'}$ is connected
Output: labelled tree of embedded components of $\mathcal{H}'$

1. Determine the set M of components maximal in $\prec|_{\mathcal{H}'}$.
2. Use algorithm A to find an $H \in M$ with an embedding $\widetilde{H}$ satisfying $\mathcal{T}_H$ and set $\mathcal{L}_H$ of constraints $L_i = \langle \bigcup_{H' \in K_i} \text{C-In}_{H,H'}, \bigcup_{H' \in K_i} \text{C-Out}_{H,H'} \rangle$ for components K_i of $\prec|_{\mathcal{H}' \setminus \{H\}}$.
   If this succeeds, let F_i be the faces of $\widetilde{H}$ corresponding to L_i, else stop.
3. For all K_i compute *order_components*(K_i) resulting in labelled trees R_i.
4. Merge the R_i to a new tree with root $\widetilde{H}$ by adding edges from $\widetilde{H}$ to the roots of R_i labelled with F_i. Finally, return this new tree.

Figure 7: Algorithm for ordering components

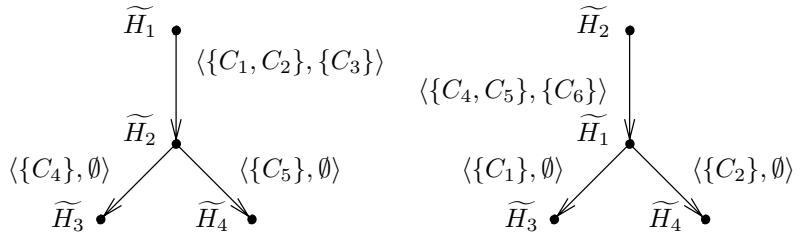

Figure 8: Two labelled trees that can result from procedure *order_components* for the graph and constraints given in Fig. 6. Instead of faces, the edges are labelled with the corresponding overlap constraints.

The complete reduction algorithm *reduce_to_components* is given in Fig. 9. To obtain a $\mathcal{T}$-embedding of G, we only have to embed the components of G according to the structure of the generated trees. Finally, we prove correctness and the time complexity for this algorithm in Theorem 4. We therefore assume that an algorithm A is available for embedding a component H with topological constraints $\mathcal{T}_H$ and overlap constraints $\mathcal{L}_H$. Its time complexity is assumed to be $O(p(n_H, k_H, t_H, l_H))$ for some polynomial p with parameters $n_H = |H|$, $t_H = |\mathcal{T}_H|$, $l_H = |\mathcal{L}_H|$ and the number k_H of all cycles occurring in the constraints of $\mathcal{T}_H$ and $\mathcal{L}_H$. Note that each overlap constraint consists then of at most k_H cycles.

Theorem 4 *Let G be a planar graph and $\mathcal{T}$ be a set of topological constraints for G. Then, algorithm* reduce_to_components *generates a $\mathcal{T}$-embedding of G iff*

Algorithm *reduce_to_components*
Input: G, $\mathcal{T}$
Output: $\mathcal{T}$-embedding of G

1. Determine the set $\mathcal{H}$ of components of G.
2. Determine the sets C-In$_{H,H'}$ and C-Out$_{H,H'}$, the induced constraints $\mathcal{T}_H := \mathcal{T}|_H$ and the relation $\prec$. If $\prec$ is cyclic, stop.
3. For each component K_i of $\prec$ compute *order_components*(K_i) resulting in labelled trees R_i.
4. Construct the embedding of G as described by the R_i, i.e. embed $\widetilde{H'}$ into face F of $\widetilde{H}$ iff there is an edge from $\widetilde{H}$ to $\widetilde{H'}$ labelled with F.

Figure 9: Reduction of $\mathcal{T}$-planarity of a graph to its components

G is $\mathcal{T}$-planar. Moreover, it takes $O\left(n^4 + n^2t + n \cdot p(n,t,t,n)\right)$ time provided that the time complexity of A is given by p.

Proof: ($\Rightarrow$) Suppose $\widetilde{G}$ is an embedding of G generated by the algorithm and $<$ is the induced relation on $\mathcal{H}$. We have to show that $\widetilde{G}$ is indeed a $\mathcal{T}$-embedding. Let be $\langle C, \text{In}_C, \text{Out}_C\rangle \in \mathcal{T}$ and $\widetilde{H}$ be the generated embedding of the component $H \in \mathcal{H}$ with $C \subset H$.

First, consider an $x \in \text{In}_C$ belonging to component $H' \in \mathcal{H}$. If $H = H'$, x is inside $\widetilde{C}$ in $\widetilde{H}$ since $\widetilde{H}$ is a $\mathcal{T}_H$-embedding. Otherwise we have $H' \prec H$. When choosing H for embedding in step 2 of algorithm *order_components*, H' belongs to some component K_i of $\prec|_{\mathcal{H}'\setminus\{H\}}$. Thus, the overlap constraint L_i ensures that all $H' \in K_i$ are embedded in a face inside $\widetilde{C}$ of $\widetilde{H}$.

Second, consider an $x \in \text{Out}_C$ belonging to component $H' \in \mathcal{H}$. Again, if $H = H'$, x is outside $\widetilde{C}$ in $\widetilde{H}$ since $\widetilde{H}$ is a $\mathcal{T}_H$-embedding. Otherwise we have $C \in$ C-Out$_{H,H'}$. If $H' < H$ holds in $\widetilde{G}$, H' must be in some component K_i of $\prec|_{\mathcal{H}'\setminus\{H\}}$ determined in step 2 of algorithm *order_components*. Thus, the overlap constraint L_i ensures that all $H' \in K_i$ are embedded in a face outside $\widetilde{C}$ of $\widetilde{H}$. If $H' < H$ does not hold in $\widetilde{G}$, $\widetilde{H'}$ is obviously outside $\widetilde{C}$ of $\widetilde{G}$.

($\Leftarrow$) Conversely, we show that the algorithm indeed finds an embedding of G if G is $\mathcal{T}$-planar. By the first part of the proof, such an embedding always satisfies $\mathcal{T}$. So let $\widetilde{G}$ be a $\mathcal{T}$-embedding of G and $<$ be its induced relation on $\mathcal{H}$. Since $<$ is an extention of $\prec$, $\prec$ is in particular acyclic. For a connected subrelation $\prec|_{\mathcal{H}'}$ of $\prec$ we denote with $\widetilde{G'}|_{\mathcal{H}'}$ the restriction of some $\mathcal{T}$-embedding $\widetilde{G'}$ of G to the components in $\mathcal{H}'$. Note that each $\widetilde{G'}|_{\mathcal{H}'}$ has a unique maximal component in $<|_{\mathcal{H}'}$.

We must show that for each call of procedure *order_components* the search for an appropriate maximal component H always succeeds. For this purpose we define inductively a $\mathcal{T}$-embedding $\widetilde{G}_{\mathcal{H}'}$ for each call *order_components*$(\mathcal{H}')$ such that the maximal component of $\widetilde{G}_{\mathcal{H}'}|_{\mathcal{H}'}$ is a possible choice in step 2. Let K_i be the components of $\prec$ and let $\widetilde{G'}$ be the embedding of G built by placing

all $\widetilde{G}|_{K_i}$ into different regions of the plane. Obviously, $\widetilde{G}'$ satisfies $\mathcal{T}$ again. We define each $\widetilde{G}_{K_i}$ to be $\widetilde{G}'$. Thus, for each call *order_components*(K_i) the maximal component of $\widetilde{G}_{K_i}|_{K_i}$ is a possible choice for H in step 2.

Consider now some call *order_components*$(\mathcal{H}')$ and let $\widetilde{G}_{\mathcal{H}'}$ be already defined. By assumption, the search in step 2 for an $H \in \mathcal{H}'$ succeeds; however, H is not necessarily identical with the maximal component of $\widetilde{G}_{\mathcal{H}'}|_{\mathcal{H}'}$. Let K_i be the components of $\prec|_{\mathcal{H}'\setminus\{H\}}$ and F_i be the corresponding faces of the generated embedding $\widetilde{H}$ in which the components in K_i must be embedded. Transform $\widetilde{G}_{\mathcal{H}'}$ into a new embedding $\widetilde{G}'$ of G by replacing $\widetilde{G}_{\mathcal{H}'}|_H$ with $\widetilde{H}$ and placing each $\widetilde{G}_{\mathcal{H}'}|_{K_i}$ into the face F_i of $\widetilde{H}$. Again, $\widetilde{G}'$ satisfies $\mathcal{T}$. We define each $\widetilde{G}_{K_i}$ to be $\widetilde{G}'$. Consequently, for each call *order_components*(K_i) the maximal component of $\widetilde{G}_{K_i}|_{K_i}$ is a possible choice for H in step 2. This completes the proof that the algorithm never stops irregularly if G is $\mathcal{T}$-planar.

Finally, we derive the time complexity of *reduce_to_components* from algorithm A. As can be easily seen, step 1, 2 and 4 of *reduce_to_components* takes $O(n)$, $O(nt)+O(n^2)$ and $O(n)$ time, respectively. To establish the time needed by step 3, we consider the call *order_components*$(\mathcal{H}')$ for some $\mathcal{H}' \subset \mathcal{H}$. First, the set M of maximal components of $\prec|_{\mathcal{H}'}$ can be determined in $O(n)$ time. For some $H_j \in M$ let be $n_j = |H_j|$, $t_j = |\mathcal{T}_{H_j}|$, $l_j = |\mathcal{L}_{H_j}|$ and k_j be the number of cycles occurring in $\mathcal{T}_{H_j}$ and $\mathcal{L}_{H_j}$, i.e. the number of cycles in H_j occurring in $\mathcal{T}$. Then, computing the l_j components of $\prec|_{\mathcal{H}'\setminus\{H_j\}}$ can be done in $O(n^2)$ time and determining the constraints $\mathcal{L}_{H_j}$ needs $O(nk_j)$ time. By assumption, testing if H_j is $\mathcal{T}_{H_j}\mathcal{L}_{H_j}$-planar can be done in $O(p(n_j,k_j,t_j,l_j))$ time. In the worst case, however, all the $H_j \in M$ must be checked. It is clear that $\sum n_j = n$, $\sum k_j = t$ and $\sum t_j \le t$. Additionally, we have $\sum l_j \le 2n$ since $\sum l_j$ is largest if $\prec|_{\mathcal{H}'}$ is a tree and for a tree the sum is twice the number of edges. Thus, step 2 of *order_components* can be done in

$$\begin{aligned}
&\sum \big(O(n^2) + O(nk_j) + O(p(n_j,k_j,t_j,l_j))\big) \\
&\quad = \quad O\Big(\sum n^2 + \sum nk_j + p\Big(\sum n_j, \sum k_j, \sum t_j, \sum l_j\Big)\Big) \\
&\quad = \quad O\left(n^3 + nt + p(n,t,t,n)\right).
\end{aligned}$$

Since the number of calls of *order_components* is bounded by n, we get the overall complexity $O\left(n^4 + n^2t + n \cdot p(n,t,t,n)\right)$ for algorithm *reduce_to_components*. □

4.2 Reducing a Component to Its Blocks

The second part of reducing $\mathcal{T}$-planarity of a graph G consists of reducing its components H with induced topological constraints $\mathcal{T}_H$ and overlap constraints $\mathcal{L}_H$ to its blocks. Again, this reduction step leads to a new sort of constraints between blocks which are similar, but not entirely equivalent with overlap constraints. Consider two blocks B and B' of H having a cutvertex v in common and suppose we have constraints in $\mathcal{T}_H$ meaning that B' must be embedded inside cycles C_1 and C_2, but outside cycle C_3 of B. A corresponding embedding

$\widetilde{B}$ of B must clearly satisfy the overlap constraint $\langle\{C_1, C_2\}, C_3\rangle$. However, due to the way these blocks are connected to each other, the topological constraints imply additional requirements: first, the boundary of the face in $\widetilde{B}$ inside C_1 and C_2, but outside C_3 must contain the cutvertex v, and second, v must belong to the boundary of the outer face of the embedding $\widetilde{B'}$ of B' in order to merge $\widetilde{B}$ and $\widetilde{B'}$ to an embedding of $B \cup B'$ satisfying the constraints given above. Since these constraints are an extention of overlap constraints, we call them *extended overlap constraints*.

Definition 3 *Let G be a graph and $\mathcal{X}$ be a set of triples $X_i = \langle C\text{-}In_i, C\text{-}Out_i, v\rangle$ such that $C\text{-}In_i$ and $C\text{-}Out_i$ are sets of cycles of G and v is a vertex of G. These triples are called* extended overlap constraints *for G.*
An embedding $\widetilde{G}$ of G satisfies $\mathcal{X}$ if for each constraint $X_i \in \mathcal{X}$ there is a face of $\widetilde{G}$ in the interior of all cycles of $C\text{-}In_i$ and in the exterior of all cycles of $C\text{-}Out_i$ and whose boundary contains v; such an embedding is called an $\mathcal{X}$-embedding. Moreover, G is $\mathcal{X}$-planar if G has an $\mathcal{X}$-embedding.

We are again only interested in those extended overlap constraints that are derived in the reduction step. It can easily be seen that the requirement for $\widetilde{B'}$ to have v at the boundary of the outer face can be expressed by the extended overlap constraint $\langle\emptyset, \text{C-Out}, v\rangle$ where C-Out denotes the set of all cycles occurring in the constraints for B'. This is because taking the face corresponding to this extended overlap constraint as the outer face results in the required embedding where v is on the outer face boundary and all other constraints are preserved.

The reduction of $\mathcal{T}_H\mathcal{L}_H$-planarity of H to its blocks works much like the first reduction step of G to its components. This is due to the fact that any embedding of H induces again an ordering $<$ on the set $\mathcal{B}$ of blocks of H with the same properties as defined in the previous section. Thus we can adopt the key idea of constructing an appropriate ordering of the blocks. However, some minor modifications are necessary to cope with both the overlap constraints $\mathcal{L}_H$ and the extended overlap constraints derived during the reduction process. We use the block graph H_b of H, as defined in Sect. 1, to represent the complete decomposition of H into its blocks and cutvertices. Note that H_b is a tree because H is connected.

Analogously to the previous section, we represent the constraints between distinct blocks $B, B' \in \mathcal{B}$ by the sets $\text{C-In}_{B,B'}$ and $\text{C-Out}_{B,B'}$. By defining them in terms of C-components, many constraints between B and further blocks are inferred. For the sake of a uniform representation, we transform the overlap constraints $L_i \in \mathcal{L}_H$ into the sets $\text{C-In}_{B,L_i}$ and $\text{C-Out}_{B,L_i}$.

$$
\begin{aligned}
\text{C-In}_{B,B'} &= \{C \subset B \mid \langle C, \text{In}_C, \text{Out}_C\rangle \in \mathcal{T}_H \text{ with } \text{In}_C \cap E(Z) \neq \emptyset \\
&\qquad \text{for some } C\text{-component } Z \text{ containing } B'\}, \\
\text{C-Out}_{B,B'} &= \{C \subset B \mid \langle C, \text{In}_C, \text{Out}_C\rangle \in \mathcal{T}_H \text{ with } \text{Out}_C \cap E(Z) \neq \emptyset \\
&\qquad \text{for some } C\text{-component } Z \text{ containing } B'\}, \\
\text{C-In}_{B,L_i} &= \{C \subset B \mid L_i = \langle \text{C-In}_i, \text{C-Out}_i\rangle \in \mathcal{L}_H \text{ with } C \in \text{C-In}_i\}, \\
\text{C-Out}_{B,L_i} &= \{C \subset B \mid L_i = \langle \text{C-In}_i, \text{C-Out}_i\rangle \in \mathcal{L}_H \text{ with } C \in \text{C-Out}_i\}.
\end{aligned}
$$

The sets C-In$_{B,B'}$ and C-In$_{B,L_i}$ define again a relation $\prec$ on $\mathcal{B} \cup \mathcal{L}_H$ by $B' \prec B$ iff C-In$_{B,B'} \neq \emptyset$ and $L_i \prec B$ iff C-In$_{B,L_i} \neq \emptyset$. The reason for including the constraints of $\mathcal{L}_H$ is that this ensures all blocks B with $L_i \prec B$ to be embedded in a linear order of $<$ as must be the case in any embedding of H satisfying $\mathcal{L}_H$ (see for example Fig. 12).

The algorithm *reduce_to_blocks* providing the second reduction step is presented in Fig. 10. Again, the resulting embedding of H is obtained from a labelled tree that is generated by the procedure *order_blocks* given in Fig. 11. This procedure assumes that some block B_0 is selected from those blocks that are maximal in $\prec$ to serve as a maximal block in the relation $<$ induced by the $\mathcal{T}_H\mathcal{L}_H$-embedding of H to be generated. Thus, contrary to the reduction algorithm in the previous section, finding a $\mathcal{T}_H\mathcal{L}_H$-embedding of H possibly requires several complete reduction processes if necessary until a proper B_0 leads to the required embedding. This is due to the fact that the generated extended overlap constraints for the blocks are partially depending on the choice of B_0; for instance, all other blocks B with cutvertex q in the path $B_0, \ldots, q, B$ of H_b are to be embedded with q at the outer face boundary. Instead of relation $\prec$, procedure *order_blocks* uses for technical reasons an extention $\prec'$ of $\prec$ by adding $B' \prec' B$ iff $B_0, \ldots, B, q, B'$ is a path in H_b. This enforces the search of which block to consider next in step 2 of *order_blocks* to take those blocks that are nearest to B_0 in the tree H_b. Note that in general, however, $\prec'$ is no longer a subrelation of $<$ induced by some $\mathcal{T}_H\mathcal{L}_H$-embedding of H. Algorithm *order_blocks* works similarly to its corresponding procedure *order_components* except for those parts where the block graph must be taken into account.

Algorithm *reduce_to_blocks*
Input: component H with constraints $\mathcal{T}_H$ and $\mathcal{L}_H$
Output: $\mathcal{T}_H\mathcal{L}_H$-embedding of H

1. Determine the set $\mathcal{B}$ of blocks of H and the blockgraph H_b of H.
2. Determine the sets C-In$_{B,B'}$, C-Out$_{B,B'}$, C-In$_{B,L_i}$ and C-Out$_{B,L_i}$, the constraints $\mathcal{T}_B := \mathcal{T}_H|_B$ and the relation $\prec$. If $\prec$ is cyclic, stop.
3. Find a block B_0 maximal in $\prec$ such that *order_blocks*$(\mathcal{B} \cup \mathcal{L}_H)$ results in a labelled tree R using the extention $\prec'$ of $\prec$ by $B' \prec' B$ iff $B_0, \ldots, B, q, B'$ is a path in H_b. If this fails, stop.
4. Construct the embedding of H as described by R, i.e. for any edge from $\widetilde{B}$ to $\widetilde{B'}$ labelled with F and for the cutvertex p with path B, p, B' in H_b embed $\widetilde{B'}$ into face F at p of $\widetilde{B}$. Additionally, return the faces corresponding to the overlap constraints $L_i \in \mathcal{L}_H$, i.e. the labels of the edges to L_i in R.

Figure 10: Reduction of $\mathcal{T}_H\mathcal{L}_H$-planarity of a connected graph to its blocks

Figure 12 illustrates this reduction step with a simple connected graph H whose blocks $B_1, \ldots, B_4$ are cycles $C_1, \ldots, C_4$, respectively, and B_5 is a bridge.

Algorithm *order_blocks*
Input: $\mathcal{B}' \subset \mathcal{B} \cup \mathcal{L}_H$ such that $\prec' |_{\mathcal{B}'}$ is connected
Output: labelled tree of embedded components of $\mathcal{B}'$

1. Determine the set M of components maximal in $\prec' |_{\mathcal{B}'}$.
2. Use algorithm A' to find some $B \in M$ such that:
   (i) for any component K_i of $\prec' |_{\mathcal{B}' \setminus \{B\}}$ there is at most one cutvertex $p_i \in B$ in H_b separating B from K_i,
   (ii) B has a $\mathcal{T}_B\mathcal{L}_B\mathcal{X}_B$-embedding $\widetilde{B}$ where
   $\mathcal{L}_B$ is the set of overlap constraints $L'_i = \langle \text{C-In}_{B,K_i}, \text{C-Out}_{B,K_i} \rangle$ for all $K_i \in \mathcal{L}_H$,
   $\mathcal{X}_B$ is the set of extended overlap constraints $X_i = \langle \bigcup_{B' \in K_i} \text{C-In}_{B,B'}, \bigcup_{B' \in K_i} \text{C-Out}_{B,B'}, p_i \rangle$ for all $K_i \notin \mathcal{L}_H$.
   If $B \neq B_0$, $\mathcal{X}_B$ contains the additional constraint that q satisfying the path $B_0, \ldots, q, B$ in H_b must be at the outer face boundary.
   If this succeeds, let F_i be the faces of $\widetilde{B}$ corresponding to L'_i and X_i, respectively, else stop.
3. For all $K_i \notin \mathcal{L}_H$ compute *order_blocks*(K_i) resulting in labelled trees R_i. For any $K_i \in \mathcal{L}_H$ define R_i to be K_i.
4. Merge the R_i to a new tree with root $\widetilde{B}$ by adding edges from $\widetilde{B}$ to the roots of R_i labelled with F_i. Finally, return this new tree.

Figure 11: Algorithm for ordering blocks

The topological constraints and the overlap constraint L for H are directly encoded in the relation $\prec$. As can easily be seen, the only correct choice for B_0 among the candidates B_1 and B_3 is $B_0 = B_1$. This is because $B_0 = B_3$ is separated in H_b by two cutvertices from $B_2 = C_2$ and $B_4 = C_4$ which inhibits embedding one of these cycles into the other and thus contradicts the overlap constraint L.

To establish in Theorem 5 correctness and time complexity of algorithm *reduce_to_blocks*, we assume an algorithm A' to be available for embedding a block B with topological constraints $\mathcal{T}_B$, overlap constraints $\mathcal{L}_B$ and extended overlap constraints $\mathcal{X}_B$. The time complexity of A' is assumed to be $O(p'(n_B, k_B, t_B, l_B, x_B))$ for some polynomial p' with parameters $n_B = |B|$, $t_B = |\mathcal{T}_B|$, $l_B = |\mathcal{L}_B|$, $x_B = |\mathcal{X}_B|$ and the number k_B of all cycles occurring in the constraints of $\mathcal{T}_B$, $\mathcal{L}_B$ and $\mathcal{X}_B$. Consequently, each overlap constraint and each extended overlap constraint consists of at most k_B cycles.

Theorem 5 *Let H be a connected planar graph and $\mathcal{T}_H$ be a set of topological constraints and $\mathcal{L}_H$ be a set of overlap constraints for H. Then, algorithm* reduce_to_blocks *generates a $\mathcal{T}_H\mathcal{L}_H$-embedding of H iff H is $\mathcal{T}_H\mathcal{L}_H$-planar. Moreover, it takes $O\left(n^5 + n^4 l + n^3 k + n^2 kl + nt + n^2 \cdot p'(n, k, t, l, n + l)\right)$ time provided that the time complexity of A' is given by p' and $n = |H|$, $t = |\mathcal{T}_H|$,*

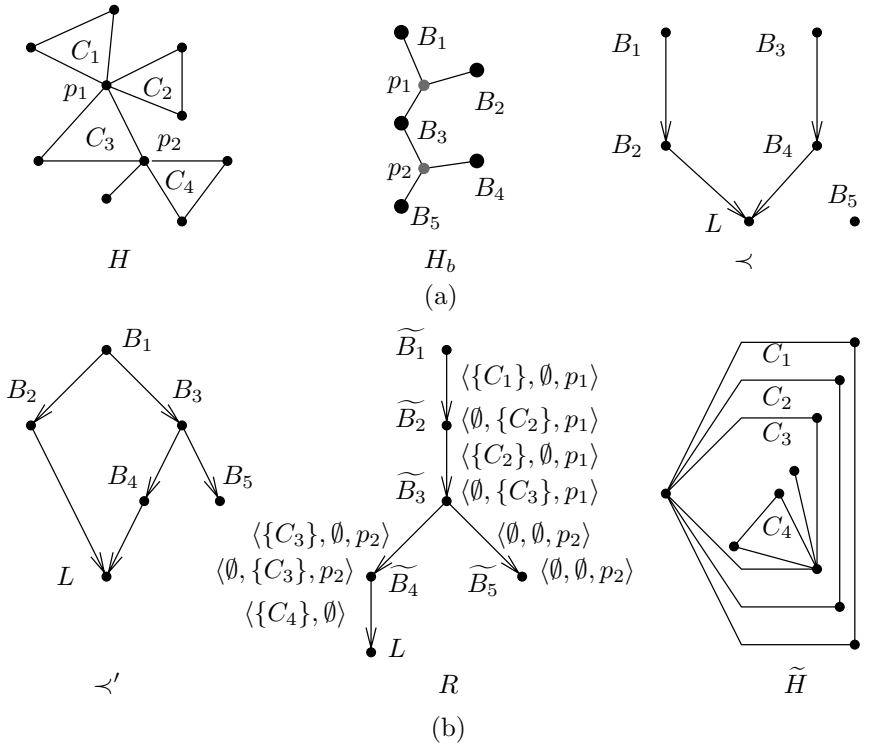

Figure 12: (a) A connected graph H with its block graph H_b and the relation $\prec$ determined by C-In$_{B_1,B_2} = \{C_1\}$, C-In$_{B_3,B_4} = \{C_3\}$, C-In$_{B_2,L} = \{C_2\}$ and C-In$_{B_4,L} = \{C_4\}$. (b) The relation $\prec'$, the labelled tree R and its corresponding embedding $\widetilde{H}$ when B_1 is chosen as B_0.

$l = |\mathcal{L}_H|$ and k is the number of all cycles occurring in the constraints of $\mathcal{T}_H$ and $\mathcal{L}_H$.

Proof: ($\Rightarrow$) Suppose $\widetilde{H}$ is an embedding of H generated by the algorithm and $<$ is the induced relation on $\mathcal{B}$. We have to show that $\widetilde{H}$ is indeed a $\mathcal{T}_H\mathcal{L}_H$-embedding. Analogously to the proof of Theorem 4, it can easily be seen that the induced topological constraints $\mathcal{T}_B$ and the extended overlap constraints X_i produced in step 2 of algorithm *order_blocks* guarantee that $\widetilde{H}$ satisfies $\mathcal{T}_H$.

To show that $\widetilde{H}$ also satisfies $\mathcal{L}_H$, we consider an overlap constraint $L = \langle\text{C-In}, \text{C-Out}\rangle \in \mathcal{L}_H$. Suppose F is the face in $\widetilde{H}$ corresponding to L and B_F is the block with an edge to L labelled with F in the tree R resulting from step 3 of *reduce_to_blocks*. That is, B_F appears as the minimal block in $<$ containing cycles from C-In. When choosing B_F for embedding in step 2 of *order_blocks*, the generated overlap constraint $L'_i = \langle\text{C-In}_{B_F,K_i}, \text{C-Out}_{B_F,K_i}\rangle$ for $K_i = L$ clearly ensures that F is a face inside all cycles of C-In in B_F and outside all cycles of C-Out in B_F. Let $C \in$ C-In belong to some block $B \neq B_F$. Since L and

B_F belong to the same component K_i determined in step 2 of *order_blocks*, the face of $\widetilde{B}$ corresponding to the extended overlap constraint X_i is guaranteed to be inside C. Thus, B_F is embedded inside C. Next, consider some $C \in$ C-Out belonging to a block $B \neq B_F$. If $B_F < B$ holds in $\widetilde{H}$, B_F must be in a component K_i. Then, the constraint X_i ensures again its corresponding face in $\widetilde{B}$ to be outside C. Otherwise, B_F is obviously outside C in $\widetilde{H}$.

($\Leftarrow$) Conversely, we show that the algorithm never stops irregularly if H is $\mathcal{T}_H\mathcal{L}_H$-planar. Suppose $\widetilde{H}$ is an embedding of H satisfying the constraints $\mathcal{T}_H$ and $\mathcal{L}_H$ and $<$ is the induced relation on $\mathcal{B}$. Clearly, the restriction of $\prec$ to $\mathcal{B}$ is a subrelation of $<$; thus $\prec$ must be acyclic. Since in step 3 of *reduce_to_blocks* possibly all maximal blocks in $\prec$ are considered, it is sufficient to show that the choice of a block B_0 which is maximal in $<$ succeeds. So let $\prec'$ be the extention of $\prec$ defined in step 3 which is also acyclic. In analogy to the proof of Theorem 4 we define a $\mathcal{T}_H\mathcal{L}_H$-embedding $\widetilde{H}_{\mathcal{B}'}$ for each call *order_blocks*($\mathcal{B}'$) such that the maximal component of $\widetilde{H}_{\mathcal{B}'}|_{\mathcal{B}'}$ is a possible choice in step 2.

First we define $\widetilde{H}_{\mathcal{B}\cup\mathcal{L}_\mathcal{H}} := \widetilde{H}$ such that for the call *order_blocks*($\mathcal{B} \cup \mathcal{L}_\mathcal{H}$) B_0 is an appropriate choice in step 2; moreover, B_0 is the only candidate because it is the unique maximal block in $\prec'$. Consider now some call *order_blocks*($\mathcal{B}'$) and let $\widetilde{H}_{\mathcal{B}'}$ be already defined. By assumption, the search in step 2 for some $B \in \mathcal{B}'$ succeeds; however, B is not necessarily identical with the maximal block of $\widetilde{H}_{\mathcal{B}'}|_{\mathcal{B}'}$. Let K_i be the components of $\prec' |_{\mathcal{B}'\setminus\{B\}}$ and F_i be the corresponding faces of the generated embedding $\widetilde{B}$ in which the blocks in K_i must be embedded. Moreover, let $p_i \in B$ be the cutvertex in H_b separating B from K_i. Then transform $\widetilde{H}_{\mathcal{B}'}$ into a new embedding $\widetilde{H}'$ of H by replacing $\widetilde{H}_{\mathcal{B}'}|_B$ with $\widetilde{B}$ and placing each $\widetilde{H}_{\mathcal{B}'}|_{K_i}$ into face F_i of $\widetilde{B}$ at cutvertex p_i. Again, $\widetilde{H}'$ satisfies $\mathcal{T}_H$ and $\mathcal{L}_H$. When each $\widetilde{H}_{K_i}$ is defined to be $\widetilde{H}'$, we assure that for each call *order_blocks*(K_i) the maximal block of $\widetilde{H}_{K_i}|_{K_i}$ is a possible choice in step 2. This completes the proof that this algorithm never stops irregularly if H is $\mathcal{T}_H\mathcal{L}_H$-planar.

Finally, we derive the time complexity of *reduce_to_blocks* from algorithm A'. As can be easily seen, step 1 and 4 of *reduce_to_blocks* takes $O(n)$ and $O(n+l)$ time, respectively. The construction of the sets C-In$_{B,B'}$ and C-Out$_{B,B'}$ in step 2 depends on the C-components which can be determined in $O(nt)$ time. Then the four sets can be determined in $O(nt) + O(kl)$ time. Moreover, defining the sets $\mathcal{T}_B$ can be done in $O(nt)$ time and the relation $\prec$ can be computed and checked in $O(n^2 + nl)$ time.

We consider next the call *order_blocks*($\mathcal{B}'$) for some $\mathcal{B}' \subset \mathcal{B} \cup \mathcal{L}_H$. The set M of maximal components in $\prec' |_{\mathcal{B}'}$ can be determined in $O(n)$ time. For some $B_j \in M$ let be $n_j = |B_j|$, $t_j = |\mathcal{T}_{B_j}|$, $l_j = |\mathcal{L}_{B_j}|$, $x_j = |\mathcal{X}_{B_j}|$ and k_j be the number of all cycles occurring in $\mathcal{T}_{B_j}$, $\mathcal{L}_{B_j}$ and $\mathcal{X}_{B_j}$, i.e. the number of all cycles in B_j occurring in $\mathcal{T}_H$ and $\mathcal{L}_H$. Then, determining the $l_j + x_j$ components K_i of $\prec' |_{\mathcal{B}'\setminus\{B_j\}}$ needs $O(n^2 + nl)$ time and determining the cutvertices p_i needs $O(n)$ time. Moreover, the constraints $\mathcal{X}_{B_j}$ can be found in $O((n+l)k_j)$ time whereas the constraints $\mathcal{L}_{B_j}$ consists of sets already generated. By assumption, a $\mathcal{T}_{B_j}\mathcal{L}_{B_j}\mathcal{X}_{B_j}$-embedding, if any, can be found in $O(p'(n_j, k_j, t_j, l_j, x_j))$ time.

In the worst case, however, all the $B_j \in M$ must be checked. It is clear that $\sum n_j \leq 2n$, $\sum k_j = k$, $\sum t_j \leq t$ and $\sum l_j = l$. By the argument given in the proof of Theorem 4, we additionally get $\sum x_j \leq 2(n+l)$. Thus step 2 of *order_blocks* can be done in

$$\begin{aligned}
&\sum (O(n^2+nl) + O((n+l)k_j) + O(p'(n_j,k_j,t_j,l_j,x_j))) \\
&= O\left(\sum (n^2+nl) + \sum (n+l)k_j + p'\left(\sum n_j, \sum k_j, \sum t_j, \sum l_j, \sum x_j\right)\right) \\
&= O\left(n^3 + n^2 l + nk + kl + p'(n,k,t,l,n+l)\right).
\end{aligned}$$

Since for a fixed B_0 the number of calls is bounded by n, the time needed by the call *order_blocks*$(\mathcal{B} \cup \mathcal{L}_H)$ in step 3 of algorithm *reduce_to_blocks* can be estimated by $n\left(O(n) + O\left(n^3 + n^2 l + nk + kl + p'(n,k,t,l,n+l)\right)\right)$, that is $O\left(n^4 + n^3 l + n^2 k + nkl + n \cdot p'(n,k,t,l,n+l)\right)$. In the worst case, however, all maximal blocks B_0 in $\prec$ must be checked. Thus we finally obtain the overall complexity

$$\begin{aligned}
&O\left(nt + kl + n^2 + nl\right) + n\left(O\left(n^4 + n^3 l + n^2 k + nkl + n \cdot p'(n,k,t,l,n+l)\right)\right) \\
&= O\left(n^5 + n^4 l + n^3 k + n^2 kl + nt + n^2 \cdot p'(n,k,t,l,n+l)\right).
\end{aligned}$$

□

As an immediate consequence, we get the complexity of this algorithm when only topological constraints $\mathcal{T}_H$ are given. In this case we have $l = 0$ and $k = t = |\mathcal{T}_H|$. Moreover, the reduction to blocks leads only to extended overlap constraints, but not to overlap constraints.

Corollary 1 *For a connected planar graph H with $n = |H|$ and a set $\mathcal{T}_H$ of topological constraints for H with $t = |\mathcal{T}_H|$, the algorithm* reduce_to_blocks *takes $O\left(n^5 + n^3 t + n^2 \cdot p'(n,t,t,0,n)\right)$ time.*

The combination of the algorithms developed in this and the previous section leads to the complete reduction method of $\mathcal{T}$-planarity of a graph G to its blocks. When replacing algorithm A with algorithm *reduce_to_blocks* in Theorem 4 (and matching the corresponding parameters), the following time complexity of the complete reduction can easily be verified.

Corollary 2 *Suppose G is a planar graph with $n = |G|$ and $\mathcal{T}$ is a set of topological constraints for G with $t = |\mathcal{T}|$. Then, algorithm* reduce_to_components *combined with algorithm* reduce_to_blocks *takes $O\left(n^6 + n^4 t + n^3 \cdot p'(n,t,t,n,2n)\right)$ time provided that the time complexity of A' is given by p'.*

We finally note that extended overlap constraints can often be simplified in the following way. Consider some constraint $X_i = \langle \text{C-In}_i, \text{C-Out}_i, v\rangle$ and a cycle $C \in \text{C-In}_i$. In case of $v \notin C$ we can remove C from C-In$_i$ and create the topological constraint meaning that v is to be embedded inside C instead. If this constraint is satisfied, any face with v at its boundary is certainly in the interior of C. The cycles of C-Out$_i$ containing v can be treated analogously. Thus, this simplification results in extended overlap constraints whose vertex is contained in all cycles involved.

5 Restricted Topological Constraints

The NP-completeness result of planarity with topological constraints opens the search for interesting tractable subproblems. Such subproblems can be defined in terms of various restrictions on graphs or constraints. For instance, one can study the effect of restricting the degree of graphs to the time complexity of $\mathcal{T}$-planarity. Instead of graph parameters, we are concerned in this section with a particular parameter inherent to topological constraints which we call *cycle intersection order*.

Definition 4 *Given a graph G and a set $\mathcal{C}$ of cycles in G, the maximum number of vertices two distinct cycles of $\mathcal{C}$ have in common is called the* cycle intersection order *of $\mathcal{C}$.*
For a set $\mathcal{T}$ of topological constraints for G we denote the set of cycles occurring in $\mathcal{T}$ by $\mathcal{C}_\mathcal{T}$. The cycle intersection order of $\mathcal{T}$ is the cycle intersection order of $\mathcal{C}_\mathcal{T}$. If d is the cycle intersection order of $\mathcal{T}$, we also write $\mathcal{T}_d$ for $\mathcal{T}$.

In other words, the cycle intersection order of $\mathcal{T}$ is the maximum order of all graphs $C_i \cap C_j$ for distinct $C_i, C_j \in \mathcal{C}_\mathcal{T}$. We consider it a natural parameter of topological constraints since it captures an important aspect of the relationships between cycles of $\mathcal{C}_\mathcal{T}$ which is worth studying. In particular, we are interested in finding the exact boundary between P and NP w.r.t. this parameter.

Recalling the proof of Theorem 3, we can immediately conclude that $\mathcal{T}_d$-planarity remains NP-hard for all $d \geq 3$. This is because any two distinct cycles involved in the defined constraints share either 2 or 3 vertices. By a minor modification of encoding a BETWEENNESS instance, we obtain NP-hardness even for $d \geq 2$.

Theorem 6 *The problem of $\mathcal{T}_d$-planarity is NP-complete for all $d \geq 2$.*

Proof: It is sufficient to show NP-hardness of $\mathcal{T}_d$-planarity for $d = 2$. Consider an instance of BETWEENNESS, i.e. a set $A = \{a_1, \ldots, a_n\}$ and a set $S = \{s_1, \ldots, s_r\}$ of ordered triples $s_j = (a_{j_1}, a_{j_2}, a_{j_3})$ of distinct elements of A. We transform the graph G_A defined in the proof of Theorem 3 into a graph G'_A by replacing each vertex a_i by the path $a_{i,1}, \ldots, a_{i,r}$ (with an edge from each of these vertices to x and y) and adding some cycle C which meets all these vertices involved exactly in x and y. This construction is illustrated in Fig. 13.

For any $s_j = (a_{j_1}, a_{j_2}, a_{j_3})$ let $C^j_{j_1,j_3}$ denote the cycle $(x, a_{j_1,j}, y, a_{j_3,j}, x)$. We define $\mathcal{T}'_S$ to contain the constraints $\langle C^j_{j_1,j_3}, \{xa_{j_2,j}\}, \emptyset\rangle$ for all triples s_j and, additionally, the constraint $\langle C, \{xa_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq r\}, \emptyset\rangle$. This last constraint ensures any embedding of G'_A to induce some linear order of the paths $a_{i,1}, \ldots, a_{i,r}$. Obviously, all these cycles have exactly x and y in common such that the cycle intersection order of $\mathcal{T}'_S$ is 2. Then it can be proved analogously to Theorem 3 that G'_A is $\mathcal{T}'_S$-planar iff $\langle A, S\rangle$ is a positive instance; however, we omit the details. □

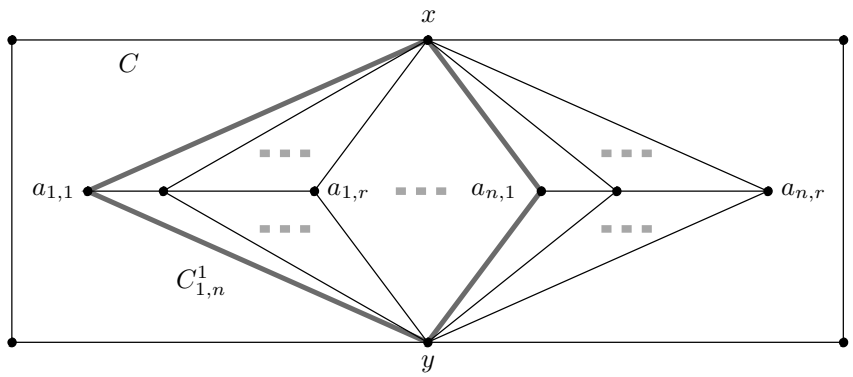

Figure 13: Graph G'_A defined in the proof of Theorem 6

Interestingly, the remaining problem of $\mathcal{T}_1$-planarity turns out to have a positive solution. The next sections are devoted to the development of a polynomial time algorithm for generating a $\mathcal{T}_1$-embedding. Thus, the boundary of $\mathcal{T}_d$-planarity between P and NP will be proved to be $d = 1$. Due to the polynomial time reduction method presented in the previous section, it is sufficient to find a polynomial time embedding algorithm for 2-connected graphs and constraints that are derived from topological constraints with cycle intersection order 1. Expectedly, such an algorithm depends primarily on certain structural properties of sets of cycles in a 2-connected graph with cycle intersection order 1. The next section explores these properties as far as required for developing the $\mathcal{T}_1$-embedding algorithm.

5.1 Decomposition of Biconnected Graphs by Cycles of Intersection Order 1

Suppose we are given a nonempty set $\mathcal{C}$ of cycles in a planar 2-connected graph G such that $\mathcal{C}$ has cycle intersection order 1. Moreover, let n denote $|G|$ and k denote $|\mathcal{C}|$. Since any edge of G belongs to at most one cycle of $\mathcal{C}$ and each cycle consists of at least three edges, we immediately get that $|\mathcal{C}| \leq \frac{3n-6}{3} = n - 2$. This also holds obviously for planar graphs with a smaller connectivity number than 2. Nevertheless, we continue to use parameter k when analyzing time complexity.

The main purpose of this section is to present a method of decomposing G into subgraphs with respect to $\mathcal{C}$ having certain desirable properties which support the construction of an embedding of G satisfying given constraints. This decomposition is described in terms of C-components of G for cycles $C \in \mathcal{C}$. The fact that G admits such a decomposition is mainly due to the cycle intersection order 1 of $\mathcal{C}$; in particular, nonplanar graphs admit the same decomposition.

We denote the set of C-components in G for all $C \in \mathcal{C}$ with $\mathcal{H}_\mathcal{C}$ and often write H_C for C-components H of G. As a simple consequence of cycle

intersection order 1 of $\mathcal{C}$, we have that C' must be entirely contained in some C-component of G for any distinct $C, C' \in \mathcal{C}$; this C-component is denoted by $H_{C,C'}$. The following lemma describes a frequently used relationship between C-components for different cycles of $\mathcal{C}$.

Lemma 1 *Let G be a 2-connected graph, $\mathcal{C}$ be a set of cycles in G with cycle intersection order 1 and $C_1, C_2, C_3 \in \mathcal{C}$. Then, if C_2 and C_3 belong to different C_1-components, C_1 and C_3 belong to the same C_2-component.*

Proof: We show that there is a path p in G from some vertex $a \in V(C_1) \setminus V(C_2)$ to some vertex $b \in V(C_3) \setminus V(C_2)$ such that p does not meet C_2. Then, since p is entirely contained in some C_2-component, both C_1 and C_3 necessarily belong to that C_2-component, too.

The C_1-component H of G containing C_3 has at least 2 vertices of attachment, but by assumption $|C_1 \cap C_2| \leq 1$, at most one of them belongs to C_2. So let $a \in V(C_1) \setminus V(C_2)$ be the other attachment vertex. Since C_2 is not contained in H, C_2 can only meet H at its attachment points. Then let $H' \subset H$ be the component of $G - C_1$ which contains vertices of C_3 and let b be such a vertex. Thus we have $H' \cap C_2 = \emptyset$ and $b \in V(C_3) \setminus V(C_2)$. Finally, if a already belongs to C_3, we define p to be the trivial path a. Otherwise, if e is an edge joining a to some $z \in H'$ and p' is a path in H' from z to b, then $p = e \cup p'$ is the required path from a to b without meeting C_2. □

Any cycle $C \in \mathcal{C}$ with more than one C-component divides G into just these C-components. It is more appropriate for our purpose, however, to take the union of these C-components with C as the subgraphs into which C divides G. Having in mind the way how cutvertices divide a connected graph into blocks, one can ask whether there is a similar decomposition of G into maximal subgraphs that cannot be further divided by cycles of $\mathcal{C}$. This motivates the following definition of *parts*.

Definition 5 *A subset $\mathcal{C}'$ of $\mathcal{C}$ with $|\mathcal{C}'| \geq 2$ is called* separable *if there is some $C \in \mathcal{C}$ with at least two C-components in G each of which containing cycles from $\mathcal{C}'$. Otherwise, $\mathcal{C}'$ is called* inseparable.
An inseparable $\mathcal{C}'$ defines the subgraph $P_{\mathcal{C}'} = \bigcap_{C \in \mathcal{C}'} C \cup H_C$ of G where H_C is the C-component containing all cycles from $\mathcal{C}' \setminus \{C\}$.
We call the subgraphs $P_{\mathcal{C}'}$ defined by some maximal inseparable $\mathcal{C}'$ and the subgraphs $C \cup H_C$ for a C-component H_C not containing cycles from $\mathcal{C}$ parts *of G w.r.t. $\mathcal{C}$. The set of all parts is denoted by $\mathcal{P}$.*
The decomposition graph *D is defined on $\mathcal{C} \cup \mathcal{P}$ by the edges CP iff $C \subset P$ for $C \in \mathcal{C}$ and $P \in \mathcal{P}$.*

To illustrate these definitions, consider the graph G and its decomposition graph D shown in Fig. 14. For instance, $\{C_5, C_8\}$ is separable since C_5 and C_8 belong to different C_1-components. $\{C_1, C_8\}$ is inseparable, but not maximal inseparable since $\{C_1, C_8, C_7\}$ is also inseparable. This set, however, is maximal inseparable, thus defining a part. As we will see, the decomposition graph is always a tree.

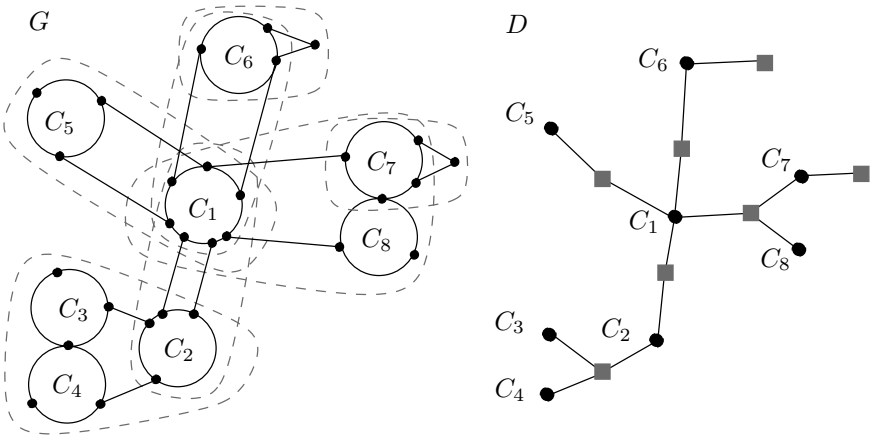

Figure 14: A 2-connected graph G (drawn nonplanar) and its decomposition graph. The cycles of $\mathcal{C}$ with intersection order 1 are drawn as circles and the parts are marked by dashed outlines and square nodes, respectively.

The definition of maximal inseparable subsets and parts, however, is nonconstructive. Thus we develop next an effective method for finding all maximal inseparable subsets of $\mathcal{C}$ which then can be transformed into a procedure for computing the parts of G. For this purpose we define in Proposition 1 the relation $\ll$ on $\mathcal{C}$ depending on some fixed root cycle C_r. It is called *weakly linear* if $C \ll C'$ and $C \ll C''$ imply $C' \ll C''$ or $C' = C''$ or $C'' \ll C'$.

Proposition 1 *Let be $C_r \in \mathcal{C}$ and the relation $\ll$ be defined on $\mathcal{C}$ by $C \ll C_r$ and $C' \ll C$ iff $C' \not\subset H_{C,C_r}$ for all distinct $C, C' \in \mathcal{C} \setminus \{C_r\}$. Then, $\ll$ is a weakly linear, strict partial order.*

Proof: It is sufficient to consider only cycles distinct from C_r. Obviously, $\ll$ is asymmetric since $C' \ll C$ implies by Lemma 1 that C_r and C are in the same C'-component, i.e. $C \not\ll C'$.

To show transitivity, assume that $C'' \ll C'$ and $C' \ll C$ holds and suppose that $C'' \not\ll C$. Since C'' and C_r are in the same C-component, but not C' and C_r, we conclude that C and C'' are in the same C'-component. In connection with $C'' \ll C'$ we then get $C \ll C'$ in contradiction to asymmetry. Thus $C'' \ll C$ must hold.

To prove that $\ll$ is weakly linear, assume that $C \ll C'$ and $C \ll C''$ and further suppose that $C' \neq C''$ and $C' \not\ll C''$. The latter means that C' and C_r are in the same C''-component, thus from $C \ll C''$ follows that C and C' are in different C''-components. Then, by Lemma 1, C'' and C are in the same C'-component. In connection with $C \ll C'$ this means that C'' and C_r are in different C'-components, thus $C'' \ll C'$. □

In other words, the transitive reduction of $\ll$ is a rooted tree with root C_r. For the example graph of Fig. 14 with C_1 chosen for C_r, $\ll$ is shown in Fig. 15.

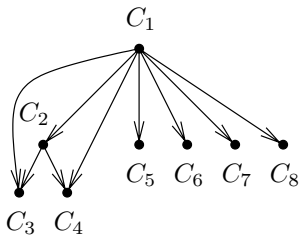

Figure 15: Relation $\ll$ for the graph given in Fig. 14 and root cycle $C_r = C_1$.

Using the relation $\ll$, we are able to determine the maximal inseparable subsets $\mathcal{C}'$ of $\mathcal{C}$, no matter which cycle is the root cycle. We show in Theorem 7 that we get all maximal inseparable sets $\mathcal{C}'$ by choosing some cycle C and all direct successors C' of C in $\ll$ belonging to the same C-component of G. The cycle C' is a successor of C if $C' \ll C$ holds, and a direct successor if there is no C'' with $C' \ll C'' \ll C$. The proof of this theorem makes use of the following lemma on maximal inseparable sets.

Lemma 2 *Let $\mathcal{C}' \subset \mathcal{C}$ be maximal inseparable and $B \in \mathcal{C}$ be a cycle not belonging to $\mathcal{C}'$. Then, there is some $C' \in \mathcal{C}'$ dividing B from $\mathcal{C}' \setminus \{C'\}$.*

Proof: Since $\mathcal{C}' \cup \{B\}$ is separable, there is a $B_1 \in \mathcal{C}$, $B_1 \neq B$, dividing B from all cycles of $\mathcal{C}' \setminus \{B_1\}$. If $B_1 \in \mathcal{C}'$, we are finished. Otherwise, since $\mathcal{C}' \cup \{B_1\}$ is also separable, there is some $B_2 \in \mathcal{C}$, $B_2 \neq B_1$, dividing B_1 from $\mathcal{C}' \setminus \{B_2\}$. We show that B_2 also divides B from $\mathcal{C}' \setminus \{B_2\}$. By Lemma 1, B_2 and all cycles of $\mathcal{C}'$ are in the same B_1-component, thus B and B_2 are in different B_1-components. Again by Lemma 1, B and B_1 are in the same B_2-component. Then we get that B_2 also divides B from $\mathcal{C}' \setminus \{B_2\}$.

If $B_2 \in \mathcal{C}'$, we are finished. Otherwise, one can find again a $B_3 \in \mathcal{C}$ dividing B_2, and thus B, from $\mathcal{C}' \setminus \{B_3\}$, and so forth. This process results in some B_i belonging to $\mathcal{C}'$ that divides B from $\mathcal{C}' \setminus \{B_i\}$. □

Theorem 7 *Let $\mathcal{C}'$ be a subset of $\mathcal{C}$ with $|\mathcal{C}'| \geq 2$. Then, $\mathcal{C}'$ is maximal inseparable iff there is a $C \in \mathcal{C}'$ such that $\mathcal{C}' \setminus \{C\}$ is a set of all direct successors $C' \ll C$ belonging to the same C-component in G.*

Proof: ($\Rightarrow$) Suppose $\mathcal{C}'$ is maximal inseparable. By Proposition 1, there is a cycle $C \in \mathcal{C}'$ that is maximal in $\ll|_{\mathcal{C}'}$. Let H be the C-component containing all the other cycles of $\mathcal{C}'$. We first prove that all $C' \in \mathcal{C}' \setminus \{C\}$ are successors of C. If $C_r \in \mathcal{C}'$, we have $C = C_r$ and, by definition, $C' \ll C$ for all $C' \in \mathcal{C}' \setminus \{C\}$. Otherwise we have $C_r \notin \mathcal{C}'$. We show that C_r is not contained in H; then all $C' \in \mathcal{C}' \setminus \{C\}$ are in a different C-component than C_r, thus it follows that $C' \ll C$. So suppose that $C_r \subset H$. Since $\mathcal{C}'$ is maximal inseparable, from Lemma 2 follows that there is some $C^* \in \mathcal{C}'$, $C^* \neq C_r$, dividing C_r from $\mathcal{C}' \setminus \{C^*\}$. C^* must be identical with C: otherwise the maximality of C implies $C \not\ll C^*$, i.e. C^* does not divide C_r from $\mathcal{C}' \setminus \{C^*\}$. However, $C = C^*$ means

that C_r is in a different C-component than all $C' \in \mathcal{C}' \setminus \{C\}$, contrary to the assumption that $C_r \subset H$.

Next we show that all $C' \in \mathcal{C}' \setminus \{C\}$ are direct successors of C in $\ll$. Suppose that there is some C^* with $C' \ll C^* \ll C$. From $C^* \ll C$ follows by Lemma 1 that C and C_r are in the same C^*-component. In connection with $C' \ll C^*$ we get that C' and C are in different C^*-components. Then, however, $\mathcal{C}'$ would not be inseparable.

Finally, we have to show that any direct successor C' of C belonging to H is already contained in $\mathcal{C}'$. Suppose that $C' \notin \mathcal{C}'$. By Lemma 2, there is some $C^* \in \mathcal{C}'$, $C^* \neq C'$, dividing C' from $\mathcal{C}' \setminus \{C^*\}$. From $C^* \ll C$ follows that C_r and C are in the same C^*-component, and $C' \not\ll C^*$ means that C_r and C' are in the same C^*-component. Then, C^* does not divide C' from C and, thus, from $\mathcal{C}' \setminus \{C^*\}$. This shows that C' must belong to $\mathcal{C}'$.

($\Leftarrow$) Suppose $\mathcal{C}'$ satisfies the requirements where C is the maximal cycle. Let H be the C-component containing all other cycles of $\mathcal{C}'$. We first show that $\mathcal{C}'$ is inseparable. So let C^* be some cycle of $\mathcal{C}$. If $C^* \not\subset H$, C^* is in a different C-component than all $C' \in \mathcal{C}' \setminus \{C\}$. By Lemma 1, all $C' \in \mathcal{C}'$ are in the same C^*-component. Suppose $C^* \subset H$, thus in particular $C^* \neq C$ and $C^* \neq C_r$. If $C^* \in \mathcal{C}'$, we have $C^* \ll C$, i.e. C_r and C are in the same C^*-component. Since for all $C' \in \mathcal{C}' \setminus \{C\}$ we have $C' \ll C$, and in particular $C' \not\ll C^*$, C_r and C' are in the same C^*-component. Thus, all $C' \in \mathcal{C}'$ belong to the same C^*-component. Otherwise, if $C^* \notin \mathcal{C}'$, C^* must be a successor of some $C' \in \mathcal{C}' \setminus \{C\}$ according to the assumption. From $C^* \ll C' \ll C$ follows that C_r, C' and C are in the same C^*-component. Since for all other $C'' \in \mathcal{C}' \setminus \{C, C'\}$ we have $C'' \not\ll C^*$, it follows that C_r and C'' are in the same C^*-component. Thus, all $C' \in \mathcal{C}'$ again belong to the same C^*-component.

Next we prove that $\mathcal{C}'$ is maximal inseparable. Suppose that for some $C^* \notin \mathcal{C}'$ $\mathcal{C}' \cup \{C^*\}$ is also inseparable. Then C^* must be contained in H. By assumption, C^* must be a successor of some $C' \in \mathcal{C}' \setminus \{C\}$, i.e. C^* and C_r are in different C'-components. From $C' \ll C$ follows that C and C_r are in the same C'-component. Thus, C^* and C are in different C'-components such that $\mathcal{C}' \cup \{C^*\}$ is separable. □

From the results shown above we get some immediate consequences characterizing parts and decomposition graphs in more detail. We summarize them in Corollary 3 and 4.

Corollary 3 *Let $\mathcal{P}$ be the set of parts of G w.r.t. $\mathcal{C}$. Then the following propositions hold:*

(i) Any part $P_{\mathcal{C}'} \in \mathcal{P}$ contains exactly the cycles $\mathcal{C}'$ from $\mathcal{C}$.

(ii) Any cycle $C \in \mathcal{C}$ in some part $P \in \mathcal{P}$ has exactly one C-component in P.

(iii) For any part $P \in \mathcal{P}$ and cycle $C \in \mathcal{C}$ we have $P \subset C \cup H_C$ for some C-component H_C of G.

(iv) Two distinct parts $P, P' \in \mathcal{P}$ have at most one cycle of $\mathcal{C}$ in common.

(v) Any part $P \in \mathcal{P}$ is 2-connected.

(vi) $\bigcup_{P \in \mathcal{P}} P = G$.

Corollary 4 *Let $\mathcal{P}$ be the set of parts and D be the decomposition graph of G w.r.t. $\mathcal{C}$. Then the following propositions hold:*

(i) D is a tree with $|D| < n$.

(ii) If H_C is a C-component of G for $C \in \mathcal{C}$ and P is the part of G with $C \subset P \subset C \cup H_C$, then $C \cup H_C$ is the union of all parts belonging to the component of $D - C$ which contains P.

(iii) G has less than n C-components, i.e. $|\mathcal{H}_\mathcal{C}| < n$.

The usefulness of decomposing G into its parts for the embedding algorithm to be developed comes in particular from proposition (ii) of Corollary 3: it guarantees that any cycle of $\mathcal{C}$ in some part P is a facial cycle of any embedding of P. As we will see in Sect. 5.2, this allows controlling the satisfaction of constraints during the embedding process. By proposition (ii) of Corollary 4, we have a one-to-one correspondence ε between C-components of G and edges of D: each edge $CP \in E(D)$ represents the C-component H_C with $C \subset P \subset C \cup H_C$, thus we define $\varepsilon(H_C) = CP$. This correspondence ensures the decomposition graph D to be a complete representation of the structure of G w.r.t. $\mathcal{C}$.

The method for finding the maximal inseparable subsets of $\mathcal{C}$ provided by Theorem 7 leads directly to an efficient procedure for computing the parts of G. The order in which the cycles are processed plays a crucial role in this procedure. Suppose $\ll$ is defined by some fixed cycle C_r and the cycles of $\mathcal{C}$ are in the reverse order $C_1, \ldots, C_k$ of some topological sorting, i.e. $C_j \ll C_i$ implies $j < i$. Thus, we always have $C_k = C_r$. Define $G_0 = G$ and $\mathcal{P}_i$, G_i and E_i for $1 \leq i \leq k$ as follows:

$$\begin{aligned} \mathcal{P}_i &= \{C_i \cup H \mid H \text{ is a } C_i\text{-component of } G_{i-1} \text{ with } C_r \not\subset H\}, \\ G_i &= C_i \cup H_r, \text{ for the } C_i\text{-component } H_r \text{ of } G_{i-1} \text{ with } C_r \subset H_r, \\ E_i &= \{C'P \mid C' \in \mathcal{C} \text{ and } C' \subset P \in \mathcal{P}_i\}. \end{aligned}$$

Defining $\mathcal{P}_{C_r} = \bigcup \mathcal{P}_i$ and the graph D_{C_r} on $\mathcal{C} \cup \mathcal{P}_{C_r}$ by the edge set $\bigcup E_i$, it can easily be seen in Proposition 2 that $\mathcal{P}_{C_r}$ is in fact the set of all parts of G and D_{C_r} is its decomposition graph.

Proposition 2 *For any $C_r \in \mathcal{C}$ $\mathcal{P}_{C_r}$ is the set $\mathcal{P}$ of parts and D_{C_r} is the decomposition graph D of G w.r.t. $\mathcal{C}$. Moreover, $\mathcal{P}$ and D can be computed by an $O(nk)$ procedure.*

Proof: By the definition of the E_i, $D_{C_r} = D$ follows from $\mathcal{P}_{C_r} = \mathcal{P}$. Due to the particular order of the cycles C_i, the successors C_j of C_i contained in G_{i-1} are exactly the direct successors of C_i. Moreover, each C_j has only one C_j-component in G_{i-1}, namely that one which contains C_i and C_r. By Theorem 7, $\mathcal{P}_i$ contains exactly the parts built by C_i and its direct successors C_j belonging

to the same C_i-component and also the parts built by C_i and a C_i-component not containing any other cycle of $\mathcal{C}$.

Finally, it can easily be seen that a procedure can be implemented for generating $\mathcal{P}$ and D which takes $O(nk)$ time. The relation $\ll$ can be determined for some fixed C_r by computing all C-components in $\mathcal{H}_{\mathcal{C}}$ which needs $O(nk)$ time. The topological sorting can be done in $O(k^2) = O(nk)$ time. And in each step i the determination of all C_i-components of G_{i-1} takes $O(n)$ time. Thus the overall complexity is $O(nk)$. □

Figure 16 illustrates the stepwise construction of the parts of graph G given in Fig. 14. The sequence of cycles is $C_3, C_4, C_2, C_5, C_6, C_7, C_8, C_1$ which is the reverse order of some topological sorting of the relation $\ll$ shown in Fig. 15.

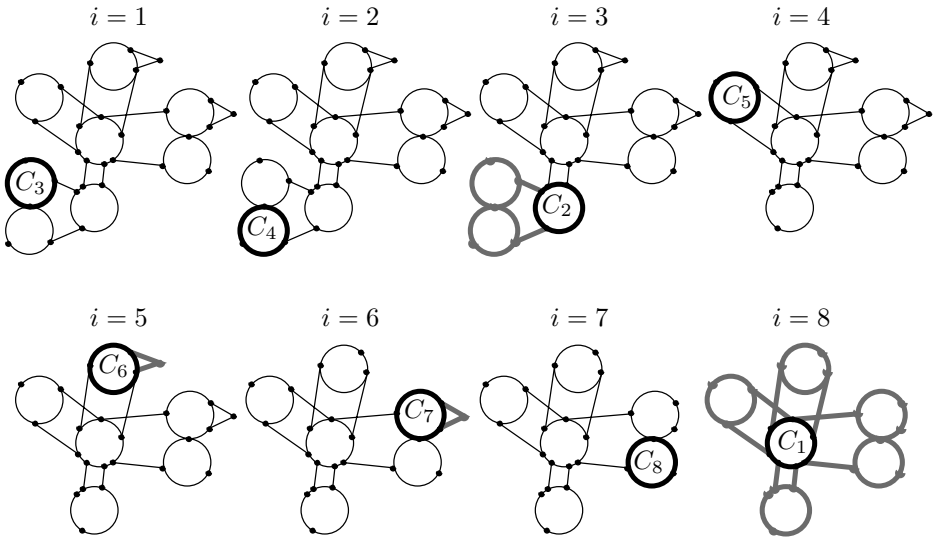

Figure 16: Construction of parts of the graph given in Fig. 14 using the ordering $C_3, C_4, C_2, C_5, C_6, C_7, C_8, C_1$. For each step i G_i is colored black and the parts in $\mathcal{P}_i$ are the grey-colored subgraphs in connection with the cycle of step i.

5.1.1 Representing Embeddings by Labelled Decomposition Graphs

Decomposition graphs are particularly interesting for the planarity problem with topological constraints. We show in this section that labelled decomposition graphs can represent, in a sense, embeddings of 2-connected planar graphs up to topological constraints, i.e. all embeddings corresponding to the same labelled decomposition graph satisfy the same topological constraints involving cycles from $\mathcal{C}$.

Consider an embedding $\widetilde{G}$ of G. Due to the correspondence between edges of D and C-components of G, we can label each edge CP with the binary information on which side of C the C-component H_C corresponding to CP is

embedded in $\widetilde{G}$. If $\widetilde{P} \subset \text{Int } \widetilde{C}$ holds in $\widetilde{G}$, we label CP with 1 and otherwise, if $\widetilde{P} \subset \text{Ext } \widetilde{C}$, we label CP with 0. Obviously, edges representing overlapping C-components have complementary labels. Since $\widetilde{G}$ defines a weakly linear, strict partial order $<$ on $\mathcal{C}$ by $C' < C$ iff $\widetilde{C'} \subset \text{Int } \widetilde{C}$, there is at least one maximal $C_r \in \mathcal{C}$ w.r.t. $<$ contained in the exterior of any other cycle of $\mathcal{C}$. As a consequence, for any path $C_r, \ldots, P, C$ in D the edge PC is labelled with 0. Thus $\widetilde{G}$ induces a labelling of D according to the following definition.

Definition 6 *Let D be the decomposition graph of G w.r.t. $\mathcal{C}$. A* labelling λ *of D is a function $\lambda : E(D) \longrightarrow \{0, 1\}$ satisfying the following conditions:*

(1) $\lambda(\varepsilon(H_C)) + \lambda(\varepsilon(H'_C)) = 1$ for overlapping C-components H_C and H'_C.

(2) There is a $C_r \in \mathcal{C}$ such that for all paths $C_r, \ldots P, C$ in D: $\lambda(PC) = 0$.

A labelling of D together with D is called a labelled decomposition graph.

Suppose $\widetilde{G}$ and $\widetilde{G}'$ are embeddings of G inducing the same labelling of D. Then any C-component H_C is inside C in $\widetilde{G}$ iff it is inside C in $\widetilde{G}'$, i.e. $\widetilde{G}$ and $\widetilde{G}'$ satisfy the same topological constraints involving cycles from $\mathcal{C}$. However, $\widetilde{G}$ and $\widetilde{G}'$ need not necessarily be topologically equivalent, they can differ in the particular embeddings of the C-components. Conversely, if two embeddings of G induce different labellings of D, they must be distinguished by at least one topological constraint. For this reason labelled decomposition graphs represent embeddings up to topological constraints. For the example graph G of Fig. 14 an embedding of G together with its labelled decomposition graph is shown in Fig. 17.

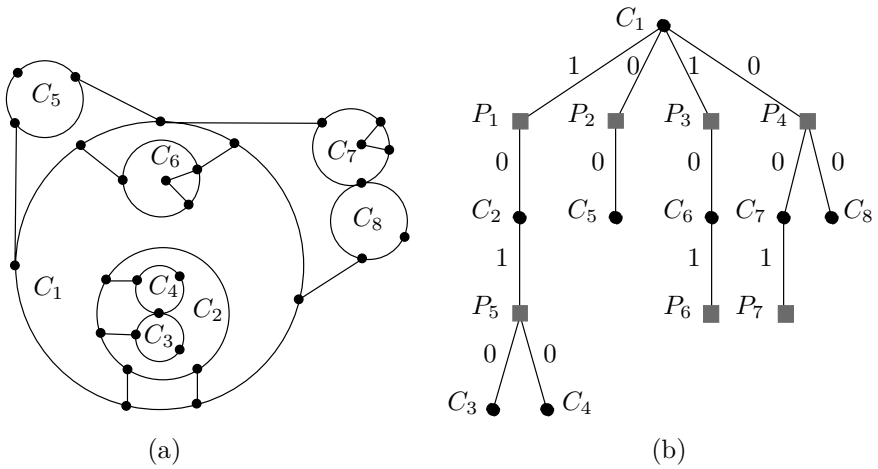

Figure 17: (a) An embedding of the graph G given in Fig. 14. (b) The decomposition graph D of G with a labelling induced by this embedding.

Given a labelling λ of D, one can easily construct an embedding of G which induces exactly λ: one has only to embed C_r and then attach in a depth-first

manner all parts P to the already embedded cycle C for edges $CP \in E(D)$ according to the label of CP. Algorithm *embedding* shown in Fig. 18 describes this embedding procedure more precisely. Depending on the labelling λ, it generates a set $\mathcal{F}$ of pairs $\langle F, S_F \rangle$ where F is a cycle in G and $S_F \subseteq \mathcal{C}$; the cycles F are called cycles of $\mathcal{F}$ and S_F are called their labels. If we choose some $\langle F_0, \emptyset \rangle$ from $\mathcal{F}$, which is guaranteed to exist, F_0 is the outer facial cycle and the cycles of $\mathcal{F}$ are the facial cycles of some embedding of G that induces λ. In this case the inner facial cycles of any $C \in \mathcal{C}$ are exactly those cycles F of $\mathcal{F}$ with $C \in S_F$. The procedure must be called with $C = C_r$ and $\mathcal{F} = \{\langle C_r, \emptyset \rangle, \langle C_r, \{C_r\} \rangle\}$; the cycles of $\mathcal{F}$ represent the two facial cycles of the embedded C_r.

Algorithm *embedding*
Input: labelling λ, $C \in \mathcal{C}$, $\mathcal{F} \subset \mathcal{C}(G) \times 2^{\mathcal{C}}$
Output: $\mathcal{F}$

For each edge $CP \in E(D)$ do:

1. Let H be the (unique) C-component in P. Find some pair $\langle F, S_F \rangle \in \mathcal{F}$ such that F contains all vertices of attachment of H and $C \in S_F$ iff $\lambda(CP) = 1$.
2. Determine the set $\mathcal{F}'$ of facial cycles of some embedding of $H \cup F$.
3. Remove $\langle F, S_F \rangle$ from $\mathcal{F}$ and add $\langle F', S_F \rangle$ to $\mathcal{F}$ for all $F' \in \mathcal{F}' \setminus \{F\}$.
4. For any edge $PC' \in E(D)$ set $S_{C'} := S_{C'} \cup \{C'\}$ for cycle C' of $\mathcal{F}$ and compute *embedding*(λ, C', $\mathcal{F}$).

Return $\mathcal{F}$.

Figure 18: Algorithm for generating an embedding corresponding to a labelled decomposition graph

To illustrate the construction of an embedding, Fig. 19 shows the sequence of partial embeddings of the example graph according to the labelling given in Fig. 17 (b). This sequence corresponds to the sequence of the sets $\mathcal{F}$ of facial cycles generated by some depth-first search of D. For each visited edge CP H is the grey-colored subgraph and F is the face cycle it is attached to. The correctness of this algorithm is proved in Proposition 3.

Proposition 3 *Let λ be a labelling of the decomposition graph of G w.r.t. $\mathcal{C}$. Then algorithm* embedding *generates a set $\mathcal{F}$ of pairs $\langle F, S_F \rangle$ with cycle $F \subseteq G$ and $S_F \subseteq \mathcal{C}$. $\mathcal{F}$ satisfies the following conditions (where $\mathcal{F}(C)$ denotes $\{F \mid \langle F, S_F \rangle \in \mathcal{F}$ with $C \in S_F\}$):*

(i) $\{F \mid \langle F, S_F \rangle \in \mathcal{F}\}$ *is the set of facial cycles of some embedding of G.*

(ii) *There is some* $\langle F_0, \emptyset \rangle \in \mathcal{F}$.

(iii) $C = \bigoplus_{F \in \mathcal{F}(C)} F$ *for all* $C \in \mathcal{C}$.

(iv) $P \subset \bigcup_{F \in \mathcal{F}(C)} F$ *iff* $\lambda(CP) = 1$ *for* $CP \in E(D)$.

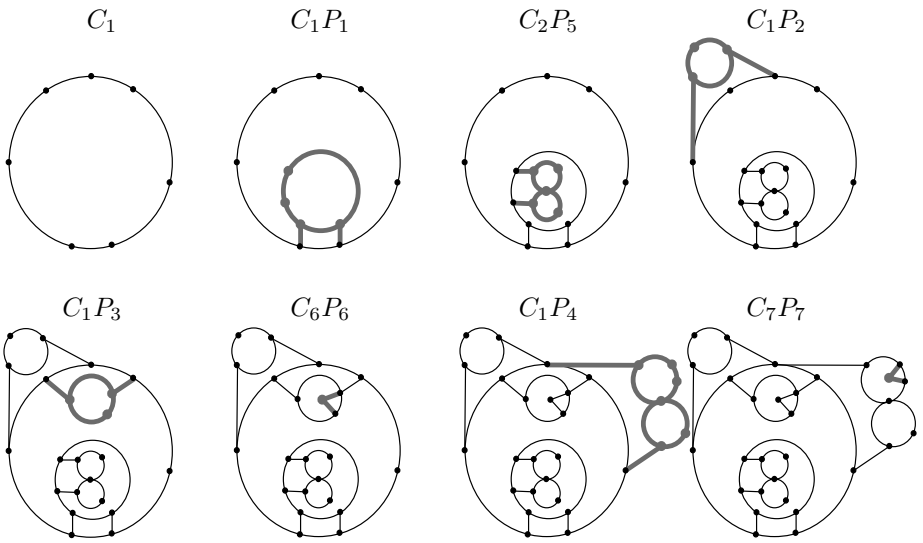

Figure 19: Illustration of algorithm *embedding* for the labelled decomposition graph D of Fig. 17 (b). For each edge CP H is the grey-colored subgraph and F the face cycle it is attached to.

Moreover, the algorithm takes $O(n^2)$ time.

Proof: Define $\mathcal{F}_0 = \{\langle C_r, \emptyset\rangle, \langle C_r, \{C_r\}\rangle\}$ and for $i = 1, \ldots, |\mathcal{P}|$ $\mathcal{F}_i$ to be the sets $\mathcal{F}$ generated by the depth-first search of the algorithm. Note that the number of visited edges $CP \in E(D)$ is identical with the number of parts of G. Let $\mathcal{F}_i(C)$ denote $\{F \mid \langle F, S_F\rangle \in \mathcal{F}_i \text{ with } C \in S_F\}$. We show by induction that the following propositions hold for all $i \geq 0$; then, since G is the union of all of its parts, $\mathcal{F}_{|\mathcal{P}|}$ satisfies the claim.

(i) $\{F \mid \langle F, S_F\rangle \in \mathcal{F}_i\}$ is the set of facial cycles of some embedding of $\bigcup_{F' \in \mathcal{F}_i} F'$.

(ii) There is some $\langle F_0, \emptyset\rangle \in \mathcal{F}_i$.

(iii) $C = \bigoplus_{F' \in \mathcal{F}_i(C)} F'$ if $\mathcal{F}_i(C) \neq \emptyset$.

(iv) If for $CP \in E(D)$ $P \subset \bigcup_{F' \in \mathcal{F}_i} F'$, then $P \subset \bigcup_{F' \in \mathcal{F}_i(C)} F'$ iff $\lambda(CP) = 1$.

Clearly, $\mathcal{F}_0$ satisfies all these conditions. We assume them to hold for all $j \leq i$ for some i. Let CP be the currently visited edge of D and H be the C-component of P. Due to condition (1) of Definition 6 and (i), there is some pair $\langle F, S_F\rangle \in \mathcal{F}_i$ such that F contains the vertices of attachment of H and $C \in S_F$ iff $\lambda(CP) = 1$. Let $\mathcal{F}'$ be the generated set of facial cycles of some embedding of $H \cup F$. By Corollary 3 (ii), F and all $C' \in \mathcal{C}$ with $PC' \in E(D)$ are facial cycles of this embedding and, thus, are contained in $\mathcal{F}'$. Moreover, we have $F = \bigoplus_{F' \in \mathcal{F}' \setminus \{F\}} F'$.

By the definition of $\mathcal{F}_{i+1} := \mathcal{F}_i \setminus \{\langle F, S_F\rangle\} \cup \{\langle F', S_F\rangle \mid F' \in \mathcal{F}', F' \neq F\}$ and $S_{C'} := S_{C'} \cup \{C'\}$, the conditions (i)–(iv) can be easily verified. Condition (i) holds since $\mathcal{F}' \setminus \{F\}$ is a refinement of F and (ii) holds because of cycle intersection order 1 of $\mathcal{C}$. Condition (iii) is shown for C' with $PC' \notin E(D)$ by

$$\bigoplus_{F' \in \mathcal{F}_{i+1}(C')} F' = \bigoplus_{F' \in \mathcal{F}_i(C') \setminus \{F\}} F' \oplus \bigoplus_{F' \in \mathcal{F}' \setminus \{F\}} F' = \bigoplus_{F' \in \mathcal{F}_i(C')} F' = C'$$

and for C' with $PC' \in E(D)$ by $\mathcal{F}_{i+1}(C') = \{C'\}$. The last condition (iv) follows immediately from the choice of F depending on $\lambda(CP)$ and condition (2) of Definition 6.

Finally, we note that this algorithm can be implemented as an $O(n^2)$ procedure since G has less than n parts and steps 1 to 4 can be done in $O(n)$ time. □

5.2 A Polynomial Time $\mathcal{T}_1$-Embedding Algorithm

Provided with the analysis of the previous sections, we are now able to present a polynomial time algorithm for generating $\mathcal{T}_1$-embeddings of planar graphs. This algorithm is a combination of the reduction algorithms given in Sect. 4 and the procedure for generating $\mathcal{TLX}$-embeddings of 2-connected graphs which we develop next. So let G denote a 2-connected planar graph and $\mathcal{T}$, $\mathcal{L}$ and $\mathcal{X}$ denote a set of topological constraints, overlap constraints and extended overlap constraints for G, respectively, such that the set $\mathcal{C}$ of all cycles involved in these constraints is of cycle intersection order 1. Moreover, we denote $n = |G|$, $t = |\mathcal{T}|$, $l = |\mathcal{L}|$, $x = |\mathcal{X}|$ and $k = |\mathcal{C}|$.

The main idea for generating a $\mathcal{TLX}$-embedding of G is finding an appropriate labelling of the decomposition graph D of G w.r.t. $\mathcal{C}$ depending on $\mathcal{T}$, $\mathcal{L}$ and $\mathcal{X}$ that can be transformed by algorithm *embedding* into a $\mathcal{TLX}$-embedding. For this purpose, we determine edges of D whose labels can be deduced from the constraints and, thus, necessarily belong to this labelling. To find labels according to condition (1) of Definition 6, we make use of overlap graphs G_C of cycles $C \in \mathcal{C}$. Consider a component Z of G_C with at least two C-components. Since G_C is bipartite, Z has a unique partition $Z = Z_1 \dot\cup Z_2$ such that both Z_1 and Z_2 contain no overlapping C-components. Then, a label for $CP \in E(D)$ with $CP = \varepsilon(H_C)$ for some C-component $H_C \in Z$ clearly determines the labels of edges representing all the other C-components in Z. We denote with $[CP]$ the set of edges representing all the C-components belonging to the same part of the partition as CP and denote with $[CP]^-$ the set of edges representing all the C-components belonging to the other part. Thus, two edges of $[CP]$ and $[CP]^-$, respectively, must have complementary labels. We call $[CP]$ the equivalence class of CP and $[CP]^-$ its complement.

With the following definitions we determine equivalence classes of edges of D whose labels can be easily deduced from the constraints. $E_1(\mathcal{T})$ and $E_0(\mathcal{T})$ are the sets of equivalence classes whose edges must have label 1 and 0, respectively, due to the topological constraints in $\mathcal{T}$. The sets $E_1(\mathcal{L}), E_0(\mathcal{L})$ and

$E_1(\mathcal{X}), E_0(\mathcal{X})$ are to be interpreted analogously. Note, however, that these sets are not meant to cover all labels that can be deduced.

$$\begin{array}{rcl} E_1'(\mathcal{T}) & = & \{[\varepsilon(H_C)] \mid H_C \cap \mathrm{In}_C \neq \emptyset \text{ for } H_C \in \mathcal{H}_\mathcal{C}, \langle C, \mathrm{In}_C, \mathrm{Out}_C\rangle \in \mathcal{T}\} \\ E_0'(\mathcal{T}) & = & \{[\varepsilon(H_C)] \mid H_C \cap \mathrm{Out}_C \neq \emptyset \text{ for } H_C \in \mathcal{H}_\mathcal{C}, \langle C, \mathrm{In}_C, \mathrm{Out}_C\rangle \in \mathcal{T}\} \\ E_1(\mathcal{T}) & = & E_1'(\mathcal{T}) \cup \{[CP]^- \mid [CP] \in E_0'(\mathcal{T})\} \\ E_0(\mathcal{T}) & = & E_0'(\mathcal{T}) \cup \{[CP]^- \mid [CP] \in E_1'(\mathcal{T})\} \end{array}$$

$$\begin{array}{rcl} E_0(\mathcal{L}) & = & \{[\varepsilon(H_{C,C'})] \mid C \in \text{C-Out}_i, C' \in \text{C-In}_i \text{ for } \langle \text{C-In}_i, \text{C-Out}_i\rangle \in \mathcal{L}\} \\ E_1(\mathcal{L}) & = & \{[CP]^- \mid [CP] \in E_0(\mathcal{L})\} \end{array}$$

$$\begin{array}{rcl} E_0'(\mathcal{X}) & = & \{[\varepsilon(H_{C,C'})] \mid C \in \text{C-Out}_i, C' \in \text{C-In}_i, \langle \text{C-In}_i, \text{C-Out}_i, v\rangle \in \mathcal{X}\} \\ E_0''(\mathcal{X}) & = & \{[\varepsilon(H_C)] \mid v \notin C \in \text{C-Out}_i, v \in H_C \in \mathcal{H}_\mathcal{C}, \langle \text{C-In}_i, \text{C-Out}_i, v\rangle \in \mathcal{X}\} \\ E_1'(\mathcal{X}) & = & \{[\varepsilon(H_C)] \mid v \notin C \in \text{C-In}_i, v \in H_C \in \mathcal{H}_\mathcal{C}, \langle \text{C-In}_i, \text{C-Out}_i, v\rangle \in \mathcal{X}\} \\ E_1(\mathcal{X}) & = & E_1'(\mathcal{X}) \cup \{[CP]^- \mid [CP] \in E_0'(\mathcal{X}) \cup E_0''(\mathcal{X})\} \\ E_0(\mathcal{X}) & = & E_0'(\mathcal{X}) \cup E_0''(\mathcal{X}) \cup \{[CP]^- \mid [CP] \in E_1'(\mathcal{X})\} \end{array}$$

As a result, we define $E_1 = E_1(\mathcal{T}) \cup E_1(\mathcal{L}) \cup E_1(\mathcal{X})$ and $E_0 = E_0(\mathcal{T}) \cup E_0(\mathcal{L}) \cup E_0(\mathcal{X})$. Next we consider overlap constraints and extended overlap constraints in more detail. By their semantics and cycle intersection order 1 of $\mathcal{C}$, the cycles of C-In$_i$ belonging to some $L_i \in \mathcal{L}$ or $X_i \in \mathcal{X}$ are embedded in any $\mathcal{TLX}$-embedding of G in a linear order w.r.t. the induced relation $<$ on $\mathcal{C}$. Thus, they must necessarily belong to a path of D if G is $\mathcal{TLX}$-planar. If this holds for $L_i = \langle \text{C-In}_i, \text{C-Out}_i\rangle$, we denote $w(L_i)$ the smallest path containing all cycles of C-In$_i$ and say that L_i induces $w(L_i)$; the constraints of $\mathcal{X}$ are treated analogously. Suppose such a path is of the form $C_1, P_1, \ldots P_j, C_j, P_j', \ldots, P_s, C_s$ with C_1, C_j and C_s belonging to a set C-In$_i$. Then in a $\mathcal{TLX}$-embedding either $C_1 < \ldots < C_j < \ldots < C_s$ or $C_s < \ldots < C_j < \ldots < C_1$ holds. We define the set E_* of equivalence classes whose edges potentially get label 1, due to these two cases, during the search for an appropriate labelling:

$$\begin{array}{rcl} E_* & = & \{[CP] \mid CP \in E(w(L_i)), C \in \text{C-In}_i \text{ if } L_i \in \mathcal{L} \text{ induces } w(L_i)\} \cup \\ & & \{[CP] \mid CP \in E(w(X_i)), C \in \text{C-In}_i \text{ if } X_i \in \mathcal{X} \text{ induces } w(X_i)\}. \end{array}$$

To construct an appropriate labelling, we have to find in particular a $C_r \in \mathcal{C}$ satisfying condition (2) of Definition 6. For each edge $CP \in E(D)$ let R_{CP} be the set $\{C' \in \mathcal{C} \mid C' \not\subset \varepsilon^{-1}(CP)\}$, i.e. the set of cycles not contained in the C-component represented by CP. We can restrict the set of candidates for C_r with the help of R_{CP}: suppose CP has label 1, then no cycle in $\mathcal{C} \setminus R_{CP}$ can satisfy condition (2). Moreover, consider a path $w(L_i)$ whose first and last edges are CP and $C'P'$, respectively. Then we can restrict the set of candidates to $R_{L_i} := R_{CP} \cup R_{C'P'}$; R_{X_i} is defined analogously. From what is considered so far, we have the following set of candidates:

$$R = \bigcap_{[CP] \in E_1} R_{CP} \cap \bigcap_{L_i \in \mathcal{L}} R_{L_i} \cap \bigcap_{X_i \in \mathcal{L}} R_{X_i}.$$

In Definition 7 we establish the criterion of being an appropriate labelling that respects the constraints $\mathcal{T}$, $\mathcal{L}$ and $\mathcal{X}$, thus we call it $\mathcal{TLX}$-labelling.

Definition 7 *A $\mathcal{TLX}$-*labelling λ *of D is a labelling satisfying the following additional conditions:*

(1) $\lambda(CP) = 1$ *if* $[CP] \in E_1$.

(2) $\lambda(CP) = 0$ *if* $[CP] \in E_0$.

(3) $C_r \in R$*; each* $L_i \in \mathcal{L}$ *and* $X_i \in \mathcal{X}$ *induces in particular a path in* D.

(4) For all paths $C_r, \ldots, C, P$ *with* $[CP] \in E_*$*:* $\lambda(CP) = 1$.

Clearly, any $\mathcal{TLX}$-embedding of G induces a $\mathcal{TLX}$-labelling on D. It can also easily be verified by Proposition 3 that the converse holds as well: given a $\mathcal{TLX}$-labelling of D, algorithm *embedding* generates some $\langle \mathcal{F}, S_F \rangle$ such that each F_0 with $\langle F_0, \emptyset \rangle$ is the outer facial cycle and the cycles of $\mathcal{F}$ are the facial cycles of some $\mathcal{TLX}$-embedding of G.

The algorithm *find_labelling* in Fig. 20 generates a $\mathcal{TLX}$-labelling of D iff G is $\mathcal{TLX}$-planar. Its main part is a depth-first search of D beginning at a candidate $C_r \in R$ that labels the visited edges. We consider D a directed tree with root C_r such that each edge is uniquely denoted by CP or PC depending on whether $C_r, \ldots, C, P$ or $C_r, \ldots, P, C$ is a path in D. When attaching labels to edges, labelling an edge with 0 or 1 fails if this edge already has the complementary label 1 or 0, respectively. Proposition 4 establishes the correctness of the algorithm.

Algorithm *find_labelling*
Input: decomposition graph D of G, constraints $\mathcal{T}$, $\mathcal{L}$, $\mathcal{X}$
Output: $\mathcal{TLX}$-labelling λ with the particular $C_r \in \mathcal{C}$

1. Determine E_1 and E_0. If $E_1 \cap E_0 \neq \emptyset$, stop.
   Check if each $L_i \in \mathcal{L}$ and $X_i \in \mathcal{X}$ induces a path in D; if not so, stop.
   Determine E_* and R. If $R = \emptyset$, stop.
2. Label any edge CP of D with 1 if $[CP] \in E_1$, and with 0 if $[CP] \in E_0$.
   Then label any edge CP not labelled so far with $*$ if $[CP] \in E_*$.
3. Find a $C_r \in R$ such that the following operations executed during a depth-first search of D beginning at C_r are successful:
   3.1 If edge PC is visited, label all edges in $[PC]$ with 0 and all edges in $[PC]^-$ with 1.
   3.2 If edge CP with label $*$ is visited, label all edges in $[CP]$ with 1 and all edges in $[CP]^-$ with 0.
   3.3 If edge CP without any label is visited, label all edges in $[CP]$ with 0 and all edges in $[CP]^-$ with 1, or vice versa.
4. If this succeeds, return C_r and labelling λ defined by the labels of D.

Figure 20: Algorithm for generating a $\mathcal{TLX}$-labelling of the decomposition graph

Proposition 4 *Given constraints $\mathcal{T}$, $\mathcal{L}$ and $\mathcal{X}$ for G, algorithm* find_labelling *generates a $\mathcal{TLX}$-labelling iff G is $\mathcal{TLX}$-planar. Moreover, the algorithm takes $O\left(n^2k + kl + kx\right)$ time.*

Proof: ($\Rightarrow$) It can easily be seen that the set of labels generated by the algorithm defines in fact a $\mathcal{TLX}$-labelling λ of D. Since for any edge $CP \in E(D)$ the edges in $[CP]$ and in $[CP]^-$ have complementary labels and, moreover, each edge has a unique label 0 or 1, condition (1) of Definition 6 is satisfied. Condition (2) is guaranteed by step 3.1, thus λ is a labelling. Conditions (1) and (2) of Definition 7 are ensured by step 2 and (3) is checked in step 1 and 3. The last condition (4) holds because of step 3.2. Thus λ is a $\mathcal{TLX}$-labelling and G is $\mathcal{TLX}$-planar.

($\Leftarrow$) Suppose G has a $\mathcal{TLX}$-embedding. Then there is a $\mathcal{TLX}$-labelling λ' induced by this embedding. In particular, we have that $E_1 \cap E_0 = \emptyset$, each $L_i \in \mathcal{L}$ and $X_i \in \mathcal{X}$ induces a path in D and $R \neq \emptyset$. Since for any edge CP of D labelled in step 2 with 0 or 1 it necessarily holds that $\lambda'(CP) = 0$ or $\lambda'(CP) = 1$, respectively, and the cycle C_r is searched exhaustively in R, it is guaranteed that the algorithm finds a complete set of labels for D which defines a $\mathcal{TLX}$-labelling as shown above.

Finally, we estimate the time complexity of this algorithm. To determine the sets in step 1, it is necessary to find the overlap graphs of cycles of $\mathcal{C}$ which can be done in $O(kn^2)$ time. Then, determining the sets $E_1(\mathcal{T})$, $E_0(\mathcal{T})$ and $E_1(\mathcal{L})$, $E_0(\mathcal{L})$ and $E_1(\mathcal{X})$, $E_0(\mathcal{X})$ needs $O(nt)$, $O(kl)$ and $O(kx)$ time, respectively. Finding the paths induced by the constraints in $\mathcal{L}$ and $\mathcal{X}$ and determining E_* can be done in $O(nl + nx)$ time. Moreover, the set R can be computed in $O(nk + lk + xk)$ time. Next, labelling edges in step 2 and during the depth-first search of D needs $O(n)$ time, thus step 3 takes $O(kn)$ time. The resulting time complexity is therefore $O(n^2k + kl + kx)$. □

Consider again the example graph of Fig. 14 and suppose we have a topological constraint meaning that cycle C_6 should be embedded inside cycle C_1 and an overlap constraint meaning that C_1 and C_3 should include a common face. The first constraint determines the set $E_1(\mathcal{T}) = \{[C_1P_3]\}$ and, thus, $E_0(\mathcal{T}) = \{[C_1P_2] = [C_1P_4]\}$. The other sets for $\mathcal{L}$ and $\mathcal{X}$ are empty. Because of the second constraint we get $E_* = \{[C_1P_1], [C_2P_1], [C_2P_5], [C_3P_5]\}$ indicating that C_1, C_2 and C_3 are to be embedded in some linear inclusion order. The set of candidates is $R = \{C_1, C_3, C_5, C_7, C_8\}$. Algorithm *find_labelling* encodes these sets by the partial labelling shown in Fig. 21 (a). In figure (b) we see some complete labelling that can result from this partial labelling when C_1 is chosen for C_r; it corresponds to the embedding shown in Fig. 17.

The complete $\mathcal{TLX}$-embedding algorithm is shown in Fig. 22. Its correctness and complexity can be easily obtained from what is developed so far.

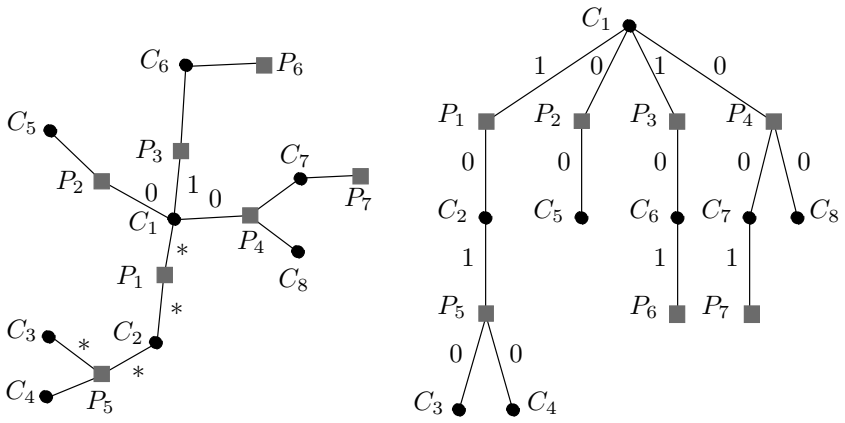

Figure 21: Illustration of algorithm *find_labelling*: (a) A partial labelling encoding a topological constraint and an overlap constraint. (b) A complete labelling produced by the algorithm.

Algorithm *embedding_with_constraints*
Input: G, constraints $\mathcal{T}$, $\mathcal{L}$, $\mathcal{X}$
Output: $\mathcal{TLX}$-embedding of G

1. Determine the decomposition graph D of G.
2. Compute *find_labelling*$(D, \mathcal{T}, \mathcal{L}, \mathcal{X})$ resulting in a labelling λ and cycle C_r if G is $\mathcal{TLX}$-planar; otherwise stop.
3. Compute *embedding*$(\lambda, C_r\{\langle C_r, \emptyset\rangle, \langle C_r, \{C_r\}\rangle\})$ resulting in $\langle \mathcal{F}, S_F\rangle$.
4. Embed G such that the cycles of $\mathcal{F}$ are the facial cycles and F_0 is the outer facial cycle for some $\langle F_0, \emptyset\rangle \in \mathcal{F}$.

Figure 22: Algorithm for generating a $\mathcal{TLX}$-embedding

Proposition 5 *Given $\mathcal{T}$, $\mathcal{L}$, $\mathcal{X}$ for G, algorithm* embedding_with_constraints *generates a $\mathcal{TLX}$-embedding iff G is $\mathcal{TLX}$-planar. Moreover, it takes $O(n^2k + kl + kx)$ time.*

The combination of algorithm *embedding_with_constraints* with algorithm *reduce_to_blocks* and algorithm *reduce_to_components* leads to the complete $\mathcal{T}_1$-embedding procedure for planar graphs. Using Corollary 1 and 2 of Theorem 5, we get the complexity of this procedure depending on whether G is disconnected, connected or 2-connected.

Theorem 8 *Suppose G is a planar graph with $n = |G|$ and $\mathcal{T}$ is a set of topological constraints for G of cycle intersection order 1 with $t = |\mathcal{T}|$. Then the following propositions hold:*

(i) If G is disconnected, a $\mathcal{T}$-embedding can be found in $O(n^6)$ time.

(ii) If G is connected, a $\mathcal{T}$-embedding can be found in $O(n^5)$ time.

(iii) If G is 2-connected, a $\mathcal{T}$-embedding can be found in $O(n^2t)$ time.

6 Conclusions

This paper contributes to the relatively new research direction on planar graph embedding with user-defined constraints. We introduced a special sort of constraints, called topological constraints, and studied the problem of finding an embedding satisfying a given set of topological constraints. As this problem turned out to be NP-complete, we developed a polynomial time procedure for reducing the problem for arbitrary planar graphs to a problem for biconnected graphs which allows to focus on biconnected graphs when searching for heuristics or tractable subproblems. This reduction, however, results in another two types of constraints one has to cope with in developing embedding algorithms. We then used this reduction for studying a particular subproblem defined by a parameter called cycle intersection order. With respect to this parameter, we proved the exact boundary between P and NP by showing that this subproblem remains NP-complete if the cycle intersection order exceeds 1 and by presenting a polynomial time embedding algorithm for cycle intersection order 0 or 1.

Acknowledgments

The author wish to thank the referees for several useful suggestions.

References

[1] J.A. Bondy, U.S.R. Murty. *Graph Theory with Applications*, MacMillan, 1976

[2] K. Booth, G. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci.*, 13: 335–379, 1976

[3] N. Chiba, T. Nishizeki, S. Abe, T. Ozawa. A linear algorithm for embedding planar graphs using PQ-trees. *J. Comput. Syst. Sci.*, 30: 54–76, 1985

[4] G. Demoucron, Y. Malgrange, R. Pertuiset. Graphes planaires: reconnaissance et construction de représentations planaires topologiques. *Revue Française de Recherche Opérationelle*, 8: 33–47, 1964

[5] G. Di Battista, P. Eades, R. Tamassia, I.G. Tollis. Algorithms for drawing graphs: An annotated bibliography. *Comp. Geom.*, 4(5): 235–282, 1994

[6] G. Di Battista, P. Eades, R. Tamassia, I.G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice Hall, 1999

[7] R. Diestel. *Graph Theory*, Springer, 1997

[8] C. Dornheim. Graph embedding with topological cycle-constraints. In J. Kratochvil (ed.), *Graph Drawing*, Proc. 7th Int. Symp. GD'99, LNCS 1731, 155–164, 1999

[9] P. Eades, Q.W. Feng. Multilevel visualization of clustered graphs. In S. North (ed.), *Graph Drawing*, Proc. 4th Int. Symp. GD'96, LNCS 1190, 101–112, 1997

[10] P. Eades, Q.W. Feng, H. Nagamochi. Drawing clustered graphs on an orthogonal grid. *J. Graph Algorithms Appl.*, 3(4): 3–29, 1999

[11] P. Eades, P. Mutzel. Graph drawing algorithms. In M.J. Atallah (ed.), *Algorithms and Theory of Computation Handbook*, CRC Press, 1999

[12] Q.W. Feng, R.F. Cohen, P. Eades. Planarity for clustered graphs. In P.G. Spirakis (ed.), *Algorithms – ESA'95*, Proc. 3rd European Symp., LNCS 979, 213–226, 1995

[13] M.R. Garey, D.S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness*, Freeman, 1979

[14] D. Harel. On visual formalisms. *Communications of the ACM*, 31(5): 514–530, 1988

[15] W. He, K. Marriott. Constrained graph layout. In S. North (ed.), *Graph Drawing*, Proc. 4th Int. Symp. GD'96, LNCS 1190, 217–232, 1997

[16] J. Hopcroft, R.E. Tarjan. Efficient planarity testing. *J. Assoc. Comput. Mach.*, 21:549–568, 1974

[17] S. MacLane. A combinatorial condition for planar graphs. *Fundam. Math.*, 28: 22–32, 1937

[18] K. Mehlhorn, P. Mutzel. On the embedding phase of the Hopcroft and Tarjan planarity testing algorithm. *Algorithmica*, 16: 233–242, 1996

[19] P. Mutzel, R. Weiskircher. Optimizing over all combinatorial embeddings of a planar graph. In G. Cornuejols, R.E. Burkard, G.J. Woeginger (eds.), *Integer Programming and Combinatorial Optimization*, Proc. 7th Int. IPCO Conf., LNCS 1610, 361–376, 1999

[20] T. Nishizeki, N. Chiba. *Planar Graphs: Theory and Algorithms*. Annals of Discrete Mathematics (32), North-Holland Mathematics Studies, 1988

[21] K. Sugiyama, K. Misue. Visualization of structured information: Automatic drawing of compound digraphs. *IEEE Trans. Syst., Man and Cybernetics*, 21(4): 876–892, 1991

[22] R. Tamassia. Constraints in graph drawing algorithms. *Constraints*, 3(1): 87–120, 1998

[23] R. Tamassia. Advances in the theory and practice of graph drawing. *Theor. Comp. Sci.*, 217(2): 235–254, 1999

[24] C. Thomassen. Planarity and duality of finite and infinite graphs. *J. Comb. Theory, Ser. B*, 29: 244–271, 1980

[25] W.T. Tutte. Matroids and graphs. *Trans. Am. Math. Soc.*, 90: 527–552, 1959

[26] W.T. Tutte. How to draw a graph. *Proc. Lond. Math. Soc.*, III. Ser. 13: 743–768, 1963

[27] C. Williams, J. Rasure, C. Hansen. The state of the art of visual languages for visualization. *Visualization '92*, Proc., IEEE Computer Society Press, 202–209, 1992