The role of planarity in connectivity problems parameterized by treewidth

Julien Baste

The role of planarity in connectivity problems parameterized by treewidth

Julien Baste
Supervised by Ignasi Sau

AlGCo project-team, CNRS, LIRMM, Montpellier, France.
jbaste@ens-cachan.fr and ignasi.sau@lirmm.fr

Abstract. For some years it was believed that for "connectivity" problems such as Hamiltonian Cycle, algorithms running in time $2^{O(\mathbf{tw})} \cdot n^{O(1)}$ –called *single-exponential*– existed only on planar and other sparse graph classes, where **tw** stands for the treewidth of the n-vertex input graph. This was recently disproved by Cygan *et al.* [FOCS 2011] and Bodlaender *et al.* [ICALP 2013], who provided single-exponential algorithms on general graphs for essentially all connectivity problems that were known to be solvable in single-exponential time on sparse graphs. During my internship we further investigate the role of planarity in connectivity problems parameterized by treewidth, and convey that several problems can indeed be distinguished according to their behavior on planar graphs. In particular, we show that there exist problems that *cannot* be solved in time $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{O(1)}$ on general graphs but that can be solved in time $2^{O(\mathbf{tw})} \cdot n^{O(1)}$ when restricted to planar graphs, and problems that can be solved in time $2^{O(\mathbf{tw} \log \mathbf{tw})} \cdot n^{O(1)}$ on general graphs but that *cannot* be solved in time $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{O(1)}$ even when restricted to planar graphs, the negative results holding unless the ETH fails. We feel that our results constitute a first step in a subject that can be much exploited.

Keywords: parameterized complexity, treewidth, dynamic programming, connectivity problems, single-exponential algorithms, planar graphs.

1 Introduction

Motivation and previous work. Graph is a fundamental data structure that represent elements and relations between them. Studying algorithms performing on these graphs is crucial in many fields, networks for example. Many problems on graphs are well known and can be solved in polynomial time. Shortest Path, which is the problem of finding the shortest path between two given vertices, is an example of problem on graph that can be solved in linear time in the number of vertices of the input graph. On an other side, many problems in graph theory are known to be NP-hard. In this report we focus only on NP-hard problems. To improve known algorithms on NP-hard problems, a solution is to find a parameter, smaller than the size of the graph, and solve problems in function of this parameter. For graph classes where the parameter is small enough, solving problem in function of the given parameter improve the time complexity.

The starting point is to see that NP-hard problems are easy to solve in trees. Because of this observation, Robertson and Seymour introduce in [19]

a graph parameter call *treewidth* that is a way to measure the distance between a given graph and a tree. This parameter becames very interesting with the apparition of the Courcelle's theorem in [3]. This theorem states that each graph problem that can be expressed in monadic second order logic can be solved in time $f(\mathbf{tw})\cdot n$ on a graph with n vertices and treewidth $\mathbf{tw}$. Even if the f function is unavoidably huge [11], this theorem is really important as it gives a first set of problems that can be solved by using treewidth. It is especially important because we know many graph classes which have a bounded treewidth. For these graph classes, problem that can be expressed in monadic second order can be solved in linear time. The natural and crucial continuation is to identify problems where the function f does not grow too fast.

If the input is a n-vertex graph G given with a tree-decomposition of width $\mathbf{tw}$, then we know many problems that can be solved in time $2^{O(\mathbf{tw}\log\mathbf{tw})}\cdot n^{O(1)}$. Intuitively, it is the case for problems that can be solved via dynamic programming on a tree-decomposition by enumerating all partitions or packings of vertices in the bags of the tree-decomposition, which are $\mathbf{tw}^{\mathbf{tw}} = 2^{O(\mathbf{tw}\log\mathbf{tw})}$ many. In this report, all the problems considered can be solved in time $2^{O(\mathbf{tw}\log\mathbf{tw})}\cdot n^{O(1)}$. In particular, we are interesting in which of them can be solved in single-exponential time i.e. in time $2^{O(\mathbf{tw})}\cdot n^{O(1)}$. Let us discuss on existing works.

It is well known that many problems can be solved in single-exponential. Intuitively it is the case for problems with locally checkable certificates i.e. a certificate that uses a constant size for each vertex and that can be check by a cardinality check and by iteratively looking at the neighborhood of the input graph. Vertex Cover, where we should find a set of vertices of minimum size such that each vertex contains at least an element of this set, is an example of these problems. For it, we enumerate subsets of the bags of an optimal tree-decomposition, which are $2^{O(\mathbf{tw})}$, and keep only the subsets that give a local certificate. In this report, we are looking on problems that have not got locally checkable certificates. More precisely, we are interesting to the class of *connectivity problems* which contains problems such as Hamiltonian cycle, Steiner Tree, or Connected Vertex Cover. These problems are called like that because the solution should satisfy a connectivity requirement (see [1,4,21] for more details). The long time used algorithms to solve these problems used classic dynamic programming techniques that enumerate all partitions or packings of the bags of the tree-decomposition.

A succession of paper shows that, when restricted to sparse graphs, many of these connectivity problems can be solved in single-exponential time. It is the case for planar graphs [10], graphs with bounded genus [7,21], and graphs excluding a fixed graph as a minor [9,22]. The idea below these

improvements is to use a special type of branch-decompositions (which are objects similar to tree-decompositions) that has nice combinatorial properties. These properties lie on the fact that graphs are sparse.

Improvement in this field was blocked at this stage during a long time. The common belief is that, on general graphs, only problems with locally checkable certificates can be solved in single-exponential time. This belief was amplified with the proof, by Lokshtanov *et al.* [17], that DISJOINT PATHS, a connectivity problem, cannot be solved in $2^{o(\mathbf{tw}\log\mathbf{tw})}\cdot n^{O(1)}$ on general graph under the exponential time hypothesis. Cygan *et al.* [4] broke this belief with a single-exponential randomize algorithm, called Cut&Count, that solved connectivity problems like LONGEST PATH, FEEDBACK VERTEX SET, or CONNECTED VERTEX COVER on general graph. Bodlaender *et al.* [1] still improved it by exhibiting another single-exponential deterministic algorithm that basicaly solves the same problems So we arrive at a point where all connectivity problems that are single-exponential on sparse graph classes [7, 9, 10, 21, 22] are also solvable in single-exponential time on general graphs [1, 4], and we have not got any example of case where sparsity improves the algorithm time.

Our results. At this stage, we could suppose that sparsity is not an interesting restriction for obtaining single-exponential algorithms. In this article, we highlight that sparsity, especially planarity, *does* play a role in connectivity problems parameterized by treewidth. For connectivity problems that can be solved in time $2^{O(\mathbf{tw}\log\mathbf{tw})}\cdot n^{O(1)}$ in general graph, we distinct three disjoint types:

- **Type 1**: Problems that can be solved in time $2^{O(\mathbf{tw})}\cdot n^{O(1)}$ on general graphs.
- **Type 2:** Problems that *cannot* be solved in time $2^{o(\mathbf{tw}\log\mathbf{tw})}\cdot n^{O(1)}$ on general graphs (up to some reasonable complexity assumption), but that can be solved in time $2^{O(\mathbf{tw})}\cdot n^{O(1)}$ when restricted to planar graphs.
- **Type 3:** Problems that *cannot* be solved in time $2^{o(\mathbf{tw}\log\mathbf{tw})}\cdot n^{O(1)}$ even when restricted to planar graphs.

Our main contribution is to show that Type 2 and Type 3 are non-empty. It permits to distinct connectivity problems not only on their behaviours on general graphs but also on their behaviours on planar graphs. More precisely, we prove the following results:

– For many problems with locally checkable certificates, it is known that single-exponential time algorithms are the best we can do unless the ETH fails [14]. These lower bounds are proved for general graphs and each problem need an ad-hoc proof. We prove, in Section 4, that PLANAR 3-COLORABILITY, which is a problem of Type 1, cannot be solved

in time $2^{o(\mathbf{tw})} \cdot n^{O(1)}$ unless the ETH fails, even when the planar input graph has maximum degree at most 5.
- We reuse Cycle Packing, Max Cycle Cover, and Maximally Disconnected Dominating Set that are prove in [4] to be unsolvable in time $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{O(1)}$ unless ETH fails. We prove that these problems are of type 2. In Section 5 we focus on Planar Cycle Packing and prove that it can be solved in time $2^{O(\mathbf{tw})} \cdot n^{O(1)}$ but cannot be solved in $2^{o(\mathbf{tw})} \cdot n^{O(1)}$ unless ETH fails.
- In Section 6, we introduce Monochromatic Disjoint Paths, a variant of the Disjoint Paths problem on a vertex-colored graph with additional color restrictions for the paths, and prove that this problem is of Type 3. We exhibit an algorithm solving Monochromatic Disjoint Paths in time $2^{O(\mathbf{tw} \log \mathbf{tw})} \cdot n^{O(1)}$ and show that Planar Monochromatic Disjoint Paths cannot be solved in time $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{O(1)}$.

Section 2 is dedicated to preliminaries. We introduce global ideas of previous works in Section 3. Section 4, Section 5, and Section 6 are dedicated respectively to problems of Type 1, Type 2, and Type 3. We add in Section 7 a lower bound for Planar Subgraph Isomorphism that uses the same ideas than the reduction of Planar Monochromatic Disjoint Paths and in Section 8 a lower bound for Planar Disjoint Paths that uses the same ideas than the reduction of Planar Cycle Packing. We finish in Section 9 by highlighting some further investigations that can be done.

2 Preliminaries

Exponential time hypothesis. The *Exponential time hypothesis.* is the hypothesis that 3-SAT cannot be solved in subexponential time. We use *ETH* as a shortcut for Exponential time hypothesis.

Sets. We use the notation $[n]$ for the set $\{1, \ldots, n\}$. In the set $[k] \times [k]$, a *row* is a set $\{i\} \times [k]$ and a *column* is a set $[k] \times \{i\}$ for some $i \in [k]$.

Graphs. For any graph-theoretic notation not defined here, the reader is referred to [5]. A *graph* G is a pair (V, E) where V is the set of vertices of G and $E \subseteq V \times V$ is the set of edges. All the considered graphs are undirected and contain neither loops nor multiple edges. We denote by $V(G)$ the set of vertices of the graph G and by $E(G)$ its set of edges. When the context is clear, n stands for the number of vertices of a graph G, which will typically be the input graph of the problem under consideration. A *subgraph* $H = (V_H, E_H)$ of a graph $G = (V, E)$ is a graph such that $V_H \subseteq V$ and $E_H \subseteq E \cap (V_H \times V_H)$. The *degree* of a vertex v in a graph G denoted by $\deg_G(v)$, is the number of edges of G containing v.

A *grid* $m * k$ is a graph $Gr_{m,k} = (\{a_{i,j} | i \in [m], j \in [k]\}, \{\{a_{i,j}, a_{i+1,j}\} | i \in [m-1], j \in [k]\} \cup \{\{a_{i,j}, a_{i,j+1}\} | i \in [m], j \in [k-1]\})$. When $m = k$ we just speak about a *grid of size* k.

We say that there is a *path* $s \dots t$ in a graph G if there exist $m \in \mathbb{N}$ and $x_0, \dots, x_m$ in $V(G)$ such that $x_0 = s$, $x_m = t$, and for all $i \in [m]$, $\{x_{i-1}, x_i\} \in E(G)$.

A *tree* T is a graph with no cycles. In a tree, a *leaf* is a vertex with degree 1.

Treewidth. A *tree-decomposition* of width w of a graph $G = (V, E)$ is a pair (T, σ), where T is a tree and $\sigma = \{B_t | B_t \subseteq V, t \in V(T)\}$ such that:

- $\bigcup_{t \in V(T)} B_t = V$,
- For every edge $\{u, v\} \in E$ there is a $t \in V(T)$ such that $\{u, v\} \subseteq B_t$,
- $B_i \cap B_k \subseteq B_j$ for all $\{i, j, k\} \subseteq V(T)$ such that j lies on the path $i \dots k$ in T,
- $\max_{i \in V(T)} |B_t| = w + 1$.

The set B_t are called *bag*. The *treewidth* of G, denoted $\mathbf{tw}(G)$, is the smallest integer w such that there is a tree-decomposition of G of width w. An *optimal tree-decomposition* is a tree-decomposition of width $\mathbf{tw}$.

Pathwidth. A *path-decomposition* of a graph $G = (V, E)$ is a tree-decomposition, (T, σ) such that T is a path. The *pathwidth* of G, denoted $\mathbf{pw}(G)$, is the smallest integer w such that there is a path-decomposition of G of width w. Clearly, for any graph G, we have $\mathbf{tw}(G) \leqslant \mathbf{pw}(G)$.

Branchwidth. A *branch-decomposition* (T, σ) of a graph $G = (V, E)$ consists of an unrooted ternary tree T and a bijection $\sigma : L \to E$ from the set L of leaves of T to the edge set of G. We define for every edge e of T the *middle set* $\mathbf{mid}(e) \subseteq V(G)$ as follows: Let T_1 and T_2 be the two connected components of $T \backslash \{e\}$. Then let G_i be the graph induced by the edge set $\{\sigma(f) : f \in L \cap V(T_i)\}$ for $i \in \{1, 2\}$. The *middle set* is the intersection of the vertex set of G_1 and G_2, i.e., $\mathbf{mid}(e) := V(G_1) \cap V(G_2)$. When we consider T as rooted, we let G_e be the graph G_i such that T_i does not contain the root of T. The *width* of (T, σ) is the maximum order of the middle sets over all edges of T, i.e., $w(T, \sigma) := \max\{|\mathbf{mid}(e)| | e \in T\}$. The *branchwidth* of G, denoted $\mathbf{bw}(G)$, is the minimum width over all branch decompositions of G. An *optimal branch decomposition* of G is a branch decomposition (T, σ) of width $\mathbf{bw}(G)$.

The branch-width of a graph G with at least 3 edges is linked to treewidth by the relation from [20]: $\mathbf{bw}(G) - 1 \leqslant \mathbf{tw}(G) \leqslant \lfloor \frac{3}{2} \mathbf{bw}(G) \rfloor - 1$.

Planar graphs. Let Σ be the sphere $\{(x, y, z) \in \mathbb{R}^3 : x^2 + y^2 + z^2 = 1\}$. By a *$\Sigma$-plane* graph G we mean a planar graph G with the vertex set $V(G)$, the edge set $E(G)$, and the face set $F(G)$ drawn without edge crossings in Σ.

An O-arc is a subset of Σ homeomorphic to a circle. An O-arc in Σ is called a *noose* of a Σ-plane graph G if it meets G only in vertices and intersects with every faces at most once. Each noose O bounds two open discs Δ_1, Δ_2 in Σ, i.e., $\Delta_1 \cap \Delta_2 = \emptyset$ and $\Delta_1 \cup \Delta_2 \cup O = \Sigma$.

For a Σ-plane graph G, we define a *sphere cut decomposition* or *sc-decomposition* (T, σ, π) of G as a branch decomposition such that for every edge e of T there exists a noose O_e bounding the two open discs Δ_1 and Δ_2 such that $G_i \subseteq \Delta_i \cup O_e$, $1 \leqslant i \leqslant 2$. Thus O_e meets G only in $\mathbf{mid}(e)$ and its length is $|\mathbf{mid}(e)|$.

Partition. A *partition* P of a set S is a set of subset of S such that $\bigcup_{s \in P} s = S$ and for all $s_1, s_2 \in P$, $s_1 \cap s_2 = \emptyset$. A partition P is called *non-crossing matching* if for each $s_1, s_2 \in P$, for each $a, b \in s_1$ and $c, d \in s_2$ with $a < b$ and $c < d$ then one of the following situations occurred: $a < b < c < d$, $a < c < d < b$, $c < d < a < b$, or $c < a < b < d$. Kreweras showed in [15] that the number of non-crossing partition on $[k]$ for $k \in \mathbb{N}$ is at most 4^k.

Matchings. A *matching* M is a set of pair, also call edges, such that for each $e, e' \in M$, $e \neq e'$, $e \cap e' = \emptyset$. For a matching M in a graph G, we denote by $V[M]$ the set of all vertices that belong to an edge of M. We say that two matchings M and M' are *disjoint* if $V[M] \cap V[M'] = \emptyset$.

A matching M on $V = \{v_1, \ldots, v_n\}$, for some $n \in \mathbb{N}^*$, is called *non-crossing matching* if for each $\{v_a, v_b\}, \{v_c, v_d\} \in M$, with $a < b$ and $c < d$, then one of the following situations occurred: $a < b < c < d$, $a < c < d < b$, $c < d < a < b$, or $c < a < b < d$.

Kreweras showed in [15] that the number of non-crossing matchings on $[k]$ for $k \in \mathbb{N}$ is at most 2^k.

Planar problem. Let P be a problem defined on graphs. We denote by Planar P the restriction of the problem P to the case where the input graphs are restricted to be planar.

3 Algorithms in $2^{O(\mathbf{tw})} \cdot n^{O(1)}$ for connectivity problems

3.1 On planar graphs

Let us present the work of Dorn *et al.* In [10], they present how to solve the majority of the planar connectivity problems in time $2^{O(\mathbf{tw})} \cdot n^{O(1)}$. For

these algorithms, we use a special branch-decomposition, the sphere cut decomposition, that has nice combinatorial properties. Let $G = (V, E)$ a graph given with its planar embedding and let (T, σ, π) be the sphere cut decomposition of the graph G. Let e be an edge of T then we can draw a circle, called noose, that meets G only in vertices and intersects with every faces at most once. Each noose O_e bounds two open discs Δ_1, Δ_2 in Σ, i.e., $\Delta_1 \cap \Delta_2 = \emptyset$ and $\Delta_1 \cup \Delta_2 \cup O = \Sigma$ such that $G_i \subseteq \Delta_i \cup O_e$, $1 \leqslant i \leqslant 2$. Thus O_e meets G only in $\mathbf{mid}(e)$ and its length is $|\mathbf{mid}(e)|$. That means that for each cut, we isolate two planar subgraphs of G that we link together. In particular, when performing dynamic programming, when we merge the two tables of the two children e_1 and e_2 of e, these tables correspond to two planar graphs where we keep information only to the vertices that are in the $\mathbf{mid}(e_i)$ and we construct an other planar graph (see Fig. 1 for an example).

This planar property mainly reduce the number of partitions in the cut set we have to consider. Kreweras showed in [15] that the number of non-crossing partitions on $[k]$ for $k \in \mathbb{N}$ is at most 4^k. So when we enumerate all partitions or packing of vertices in the $\mathbf{mid}(e)$ for a planar connectivity problems, we just need to keep $4^{\mathbf{tw}}$ partitions for each $\mathbf{mid}(e)$, $e \in T$. With this, the complexity analysis of algorithms that we use in the general case gives in the planar case a solution in time $2^{O(\mathbf{tw} \log \mathbf{tw})} \cdot n^{O(1)}$.

This technique can be adaped for input graph with bounded genus or that exclude a graph H as minor. As it uses the structure of the input graph, it cannot be generalized to general graphs.

3.2 Cut & Count

The Cut&Count technique [4] is based on Monte-Carlo algorithms and can give false negative with probability at most $\frac{1}{2}$. On the other hand, it cannot give false positive and solves many connectivity problems in time $2^{O(\mathbf{tw})} \cdot n^{O(1)}$. The Cut&Count technique is based on the isolation lemma [18] that permits to count objects modulo 2 since if we have many solutions to a problem, it reduces them to a unique one with a high probability.

Definition 1 *A function $\omega : U \to Z$ isolates a set family $F \subseteq 2^U$ if there is a unique $S \in F$ with $\omega(S) = min_{s \in S}(S)$, with the notation $\omega(X) = \Sigma_{u \in X} \omega(u)$.*

Definition 2 (Isolation Lemma) *Let $\mathcal{F} \subseteq 2^U$ be a set family over a universe U with $|\mathcal{F}| > 0$. For each $u \in U$, choose a weight $\omega(u) \in \{1...N\}$ uniformly and independently at random. Then*

$$Prob[\omega \text{ isolate } \mathcal{F}] \geqslant 1 - \frac{U}{N}$$

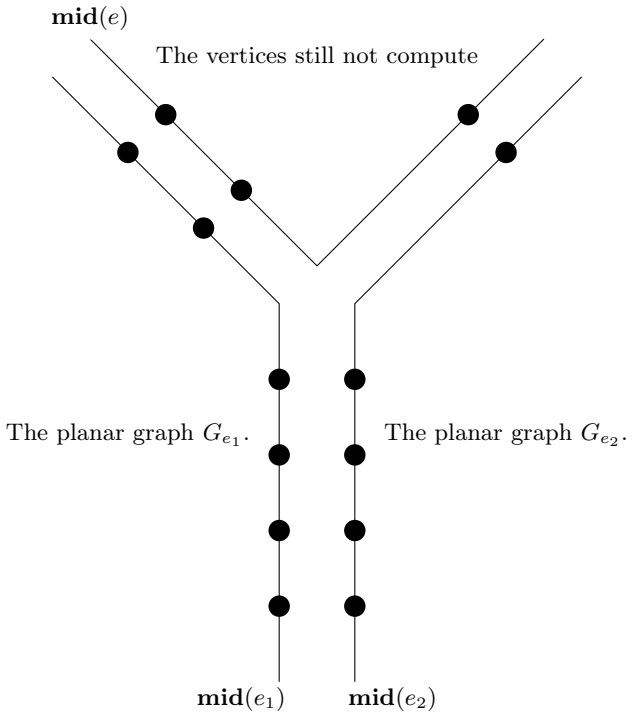


Fig. 1. An illustration of a use of sphere cut decomposition.

Let $\mathcal{S} \subseteq 2^U$ be a set of solution, we want to know if this set is empty. As its name says, the Cut&Count technique is based on two parts :

The Cut part: We relax the connectivity requirement so we take a set $\mathcal{R}$ of all possible connected candidate solutions so $\mathcal{S} \subseteq \mathcal{R}$. We also consider the set $\mathcal{C}$ of pairs (X, C) where $X \in \mathcal{R}$ and C is a consistent cut of X.

The Count part: We compute $|\mathcal{C}|$ modulo 2 using a sub-procedure. Non-connected candidate solutions $X \subseteq \mathcal{R} \backslash \mathcal{S}$ are canceled since they are consistent with an even number of cuts. Connected candidates $X \in \mathcal{S}$ remain.

Note that we use the isolation lemma to obtain an odd number of solutions in order to make the counting part work.

3.3 rank based and Square determinant algorithms

The Cut&Count technique has several disadvantages. First, it is a probabilistic solution, we cannot be confident in the result. Second, they do not extend to counting the number of witnesses. Third, they do not give intuition for the optimal substructure or equivalence classes. Cut&Count technique is an hope to deterministic algorithms working faster than

$2^{O(\mathbf{tw}\log \mathbf{tw})} \cdot n^{O(1)}$. The first idea is to derandomize Cut&Count but it requieres to derandomize the isolation lemma that remains a big open question.

In [1], authors get rid of these disadvantages. They use the "rank based" and the "determinant" approaches that permit to remove weight and counting versions problems that appear in Cut&Count. Removing disadvantages has a price and the runtime is slightly worse.

The main idea to the rank based approach is, given a graph G and tree-decomposition (T, σ), to define for each bags of the tree-decomposition a *representative* set such that it is enough to enumerate only these representative elements and such that they are $2^{O(\mathbf{tw})}$ many. For it, authors define few operations and prove that if we can solve a problem with a dynamic programming algorithm that uses only these operations then, we can find for each bag of the tree-decomposition a representative set of size $2^{O(\mathbf{tw})}$ and so, we can solve the considered problem in time $2^{O(\mathbf{tw})} \cdot n^{O(1)}$.

In an other hand, if the goal is to compute the number of solution to a given connectivity problem, the determinant approach is more adapted. The key idea is to define for a graph G a matrix A that has the main property that if G is a tree then $|\det(A)| = 1$ else $\det(A) = 0$. With this observation, to count the number of solutions of our connectivity problem, we bring the count of the number of solutions to the count of the number of some trees.

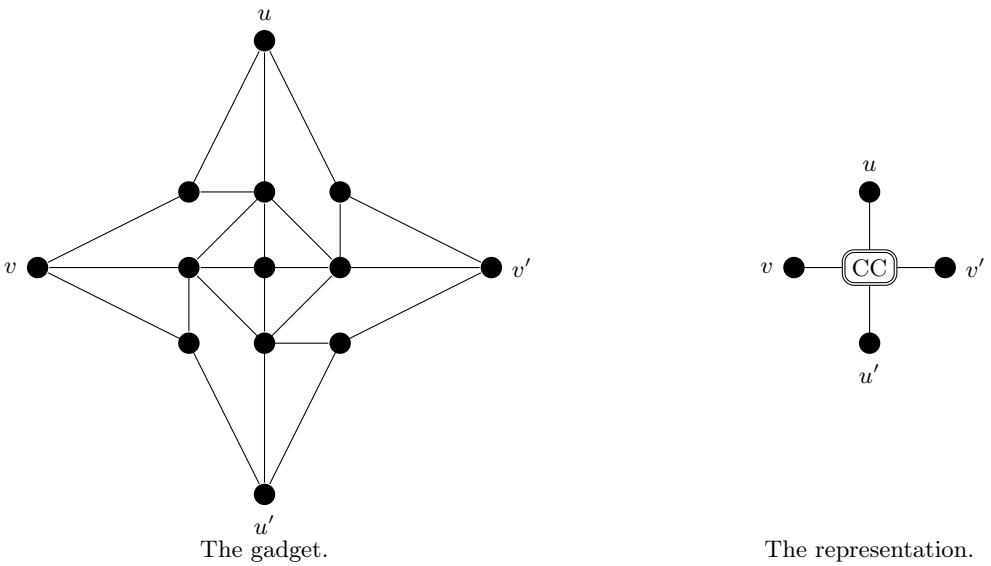
4 Tight Problems of Type 1

In this section we show that Planar 3-colorability cannot be solved in time $2^{o(\mathbf{tw})} \cdot n^{O(1)}$ even when the input graph has maximum degree at most 5. It is a commun belief that NP-hard problems parametrized by **tw** cannot be solved in time $2^{o(\mathbf{tw})} \cdot n^{O(1)}$, assuming some standard complexity assumtion. By making reduction that preserved subexponential complexity, the lower bound $2^{o(\mathbf{tw})} \cdot n^{O(1)}$ was proved in [14] for problems such that k-Colorability, k-Set Cover, Independent Set, or Vertex Cover assuming ETH. For these reductions the main point is that the relevant parameters should not increase more than linearly. To our best knowledge, there is no such reduction for Planar 3-colorability or Planar Cycle Packing. We show that there is a reduction where the parameter increases quadratically which turn out to be enough for proving an exponential lower bound in the treewidth.

3-colorability
Input: A graph $G = (V, E)$.
Question: Is there a color function $c : V \to \{1, 2, 3\}$ such that for all $\{x, y\} \in E$, $c(x) \neq c(y)$.

The gadget. The representation.

Fig. 2. Color gadget.



The gadget. The representation.

Fig. 3. Cross-color gadget.

Theorem 1. Planar 3-colorability *cannot be solved in time* $2^{o(\sqrt{n})} \cdot n^{O(1)}$ *unless the ETH fails even when the input graph is restricted to have maximum degree at most 5.*

Proof: For the reduction, we need some planar gadgets. The first one is depicted in Fig. 2 and named C-gadget. This gadget ensures that two vertices u and u' are in the same color class. Note that we can extend the C-gadget for three vertices u, u', and u'' and ensure the three vertices to be in the same color class by fixing a C-gadget between u and u' and an other C-gadget between u' and u''. The second one is depicted in Fig. 3 and named CC-gadget. In this gadget, introduced in [13], we can check that if u, v, u' and v' are in the same face and oriented in this order around the face, that u and u' are in the same color class and v and v' are in the same color class.

We reduce from 3-colorability. Let $G = (V, E)$ be a general graph with $V = \{v_1, \ldots, v_n\}$ and let $n = |V|$. We proceed to construct a planar graph H.

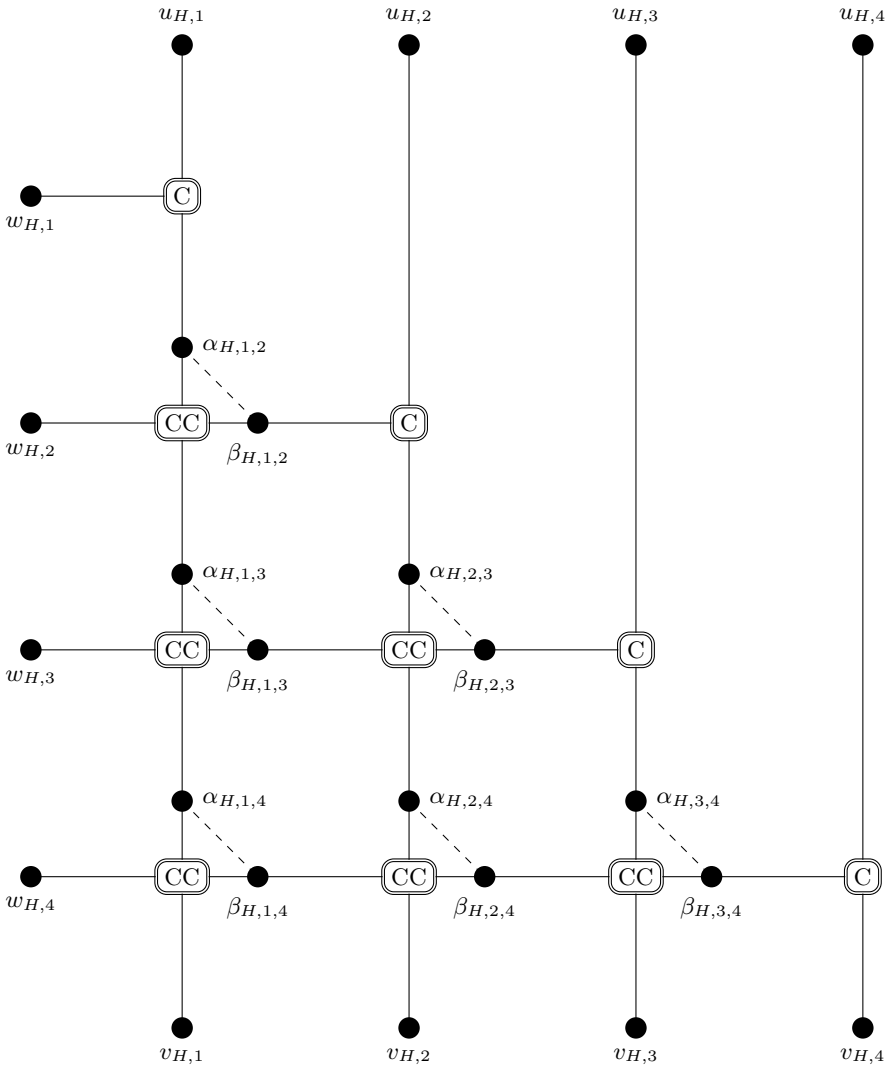
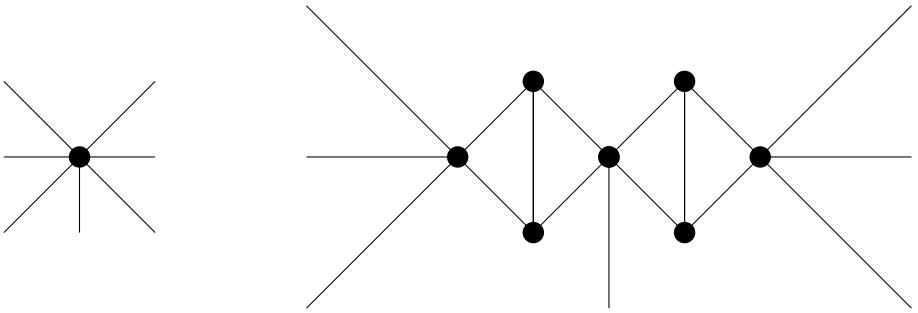


Fig. 4. An example of H_x graph with $n_x = 4$.

We construct the graph H, illustrated in figure 4, as follows :

- For each $i \in [n_x]$, $u_{H,i}, v_{H,i}, w_{H,i} \in V(H)$.
- For each $i, j \in [n_x]$, $i < j$, $\alpha_{H,i,j} \in V(H)$ and $\beta_{H,i,j} \in V(H)$.
- For each $i \in \{1, \ldots, n_x - 1\}$, there is a C-gadget between $u_{H,i}$ and $\alpha_{H,i-1,i}$.
- For each $i \in \{2, \ldots, n_x\}$, there is a C-gadget between $u_{H,i}$ and $\beta_{H,i-1,i}$.
- There is a C-gadget between u_{H,n_x} and w_{H,n_x}.
- There is a C-gadget between $u_{H,1}$ and $v_{H,1}$.
- For each $i, j \in \{2, \ldots, n_x - 1\}$, $i < j$ there is a CC-gadget between $\alpha_{H,i,j}$, $\beta_{H,i,j}$, $\alpha_{H,i,j+1}$, and $\beta_{H,i-1,j}$.
- For each $i \in \{2, \ldots, n_x - 1\}$, $i < j$ there is a CC-gadget between α_{H,i,n_x}, β_{H,i,n_x}, $w_{H,i}$, and $\beta_{H,i-1,n_x}$.



The original node. The simulation of the same node but with locally maximal degree 5.

Fig. 5. Maximal degree diminution to 5.

- For each $j \in \{2, \ldots, n_x - 1\}$, $i < j$ there is a CC-gadget between $\alpha_{H,1,j}$, $\beta_{H,1,j}$, $\alpha_{H,1,j+1}$, and $v_{H,j}$.
- There is a CC-gadget between $\alpha_{H,1,n_x}$, $\beta_{H,1,n_x}$, $w_{H,1}$, and v_{H,n_x}.
- For each $i, j \in [n_x]$, $i < j$, if $(v_i, v_j) \in E$, then $(\alpha_{H,i,j}, \beta_{H,i,j}) \in E(H)$.

Note that H is planar as the C-gadget and the CC-gadget are planar and we can keep the planarity when building H as Fig. 4 shows.

Because of the properties on the C-gadget and the CC-gadget, for each i in $[n]$, $u_{H,i}$, $v_{H,i}$, and $w_{H,i}$ are in the same color class. Because of the edges $(\alpha_{H,i,j}, \beta_{H,i,j})$, if there is an edge between v_i and v_j in G, then $u_{H,i}$ and $u_{H,j}$ should receive different colors. If we have a 3-coloring of G then by coloring $u_{H,i}$ with the color of v_i for each $i \in [n]$ then we find a 3-coloring of H. If we have a coloring of H we color each v_i of V with the color of $u_{H,i}$ for each $i \in [n]$.

Let us argue about the maximal degree of the graph H. With the previous construction, the maximal degree in 7. To restrict it to 5, we need to insert two color gadgets as show in figure 5 for each node with degree 6 or 7.

Let us argue about the number of vertices of H. H is a grid of size n where each node is replace by a C-gadget, a CC-gadget, or is removed. As these two gadgets have at most 13 vertices, and in the worst case, all these new vertices have degree 7 and we need to replace them by two color gadgets, that have 7 vertices, to reduce to a degree 5, we have that $|V(H)| \leqslant 65 \cdot n^2$. As [14] prove that 3-COLORABILITY cannot be solved in time $2^{o(n)} \cdot n^{O(1)}$, the theorem follow. □

Corollary 1. PLANAR 3-COLORABILITY *cannot be solved in time* $2^{o(\mathbf{tw})} \cdot n^{O(1)}$ *unless the ETH fails, where* **tw** *stands for the treewidth of the input graph even when the input graph is restricted to have maximum degree at most 5.*

Proof: As a planar graph G on n vertices satisfies $\mathbf{tw}(G) = O(\sqrt{n})$ [12], an algorithm in time $2^{o(\mathbf{tw})} \cdot n^{O(1)}$ for Planar 3-colorability implies that there is an algorithm in time $2^{o(\sqrt{n})} \cdot n^{O(1)}$, which is impossible by Theorem 4 unless the ETH fails. □

5 Problems of Type 2

In this section we prove that there exist problems of Type 2. More precisely, we prove that Cycle Packing, Max Cycle Cover, Maximally Disconnected Dominating Set, and Maximally Disconnected Vertex Cover are of Type 2.

5.1 The upper bound for Planar Cycle Packing

Cycle Packing
Input: A graph $G = (V, E)$ and an integer ℓ_0.
Parameter: The treewidth **tw** of G.
Question: Does G contain ℓ_0 pairwise vertex-disjoint cycles?

Theorem 2. *The* Planar Cycle Packing *problem can be solved in time* $2^{O(\mathbf{tw})} \cdot n^{O(1)}$.

Proof: We strongly follow the techniques introduced in [8,10,21,22], which are based on *Catalan structures*.

Let G be a graph, $X \subseteq V(G)$, and M a matching on $V(G) \backslash X$. Intuitively, M represents the endpoints of the paths we are building and X is the set of vertices that are already inside a path but they are not an endpoint of any path. We define $G[(X, M, \ell)] = (V[M], M)$. We say that $G[(X_1, M_1, \ell_1), (X_2, M_2, \ell_2)]$ is *defined* if $X_1 \cap (X_2 \cup V[M_2]) = X_2 \cap (X_1 \cup V[M_1]) = \emptyset$ and we define $G[(X_1, M_1, \ell_1), (X_2, M_2, \ell_2)] = G[(X_1, M_1, \ell_1)] \cup G[(X_2, M_2, \ell_2)]$. Otherwise, we say that $G[(X_1, M_1, \ell_1), (X_2, M_2, \ell_2)]$ is *undefined*. We say that $cp(G, X, M) \geqslant \ell$ if G contains paths joining each pair of vertices given by M and ℓ cycles, all pairwise vertex-disjoint.

We now consider $G = (V, E)$ to be our Σ-plane input graph and ℓ_0 our integer. Let (T, μ, π) be a sc-decomposition of G of width **bw**. As in [10], we root T by arbitrarily choosing an edge e and we subdivide it by inserting a new node s. Let e' and e'' be the new edges and set $\mathbf{mid}(e') = \mathbf{mid}(e'') = \mathbf{mid}(e)$. We create a new node root r, we connect it to s, by an edge e_r, and set $\mathbf{mid}(e_r) = \emptyset$. The root e_r is not consider as a leaf.

Let $e \in E(T)$ and $\mathcal{R}_e = \{(X, M, \ell) | X \subseteq \mathbf{mid}(e)$, M is a matching of a subset of $\mathbf{mid}(e) \backslash X$ and $cp(G_e, X, M) \geqslant \ell\}$. We observe that there

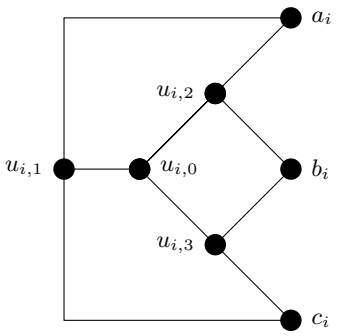


Fig. 6. The SCi-gadget.

exist ℓ_0 pairwise vertex-disjoint cycles in G if and only if $(\emptyset, \emptyset, \ell_0) \in \mathcal{R}_{e_r}$. We should now compute $\mathcal{R}_{e_r}$. If e is a leaf then $G_e = (\{x, y\}, \{(x, y)\})$ and $\mathcal{R}_e = \{(\emptyset, \emptyset, 0), (\emptyset, \{(x, y)\}, 0)\}$. Otherwise, let e_1 and e_2 be the two children of e in $E(T)$. $\mathcal{R}_e$ is the set of all triples (X, M, ℓ) such that there exist $(S_1, S_2) = ((X_1, M_1, \ell_1), (X_2, M_2, \ell_2)) \in \mathcal{R}_{e_1} \times \mathcal{R}_{e_2}$ such that $M \subseteq ((V[M_1] \cup V[M_2]) \cap (\mathbf{mid}(e) \setminus X))^2$, $G[S_1, S_2]$ is defined, all vertices in $\mathbf{mid}(e)$ of degree at least two in $G[S_1, S_2]$ are in X, and we can find, in $G[S_1, S_2]$, ℓ_3 cycles and a path $x \dots y$ for each $(x, y) \in M$ such that $\min(\ell_1 + \ell_2 + \ell_3, \ell_0) \geqslant \ell$.

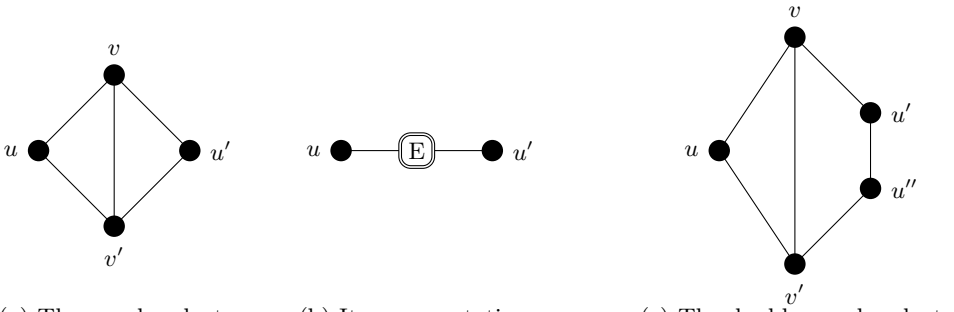
Note that $G[S_1, S_2]$ is a minor of G so $G[S_1, S_2]$ is also planar. As we have considered a sc-decomposition and all the paths we consider in $G[S_1, S_2]$ are pairwise vertex-disjoint, because each vertex has degree at most two, the maximal number of distinct matchings M is bounded by the number of non-crossing matchings, that is at most $2^{\mathbf{mid}(e)}$. As we have at most $3^{\mathbf{mid}(e)}$ choices for X and $V[M]$, it follows that for each $e \in E(T)$, $|\mathcal{R}_e| \leqslant 6^{\mathbf{mid}(e)} \cdot \ell_0$. As for each $e \in E(T)$ such that e is not a leaf, we have to merge the tables of the two children e_1 and e_2 of e then this algorithm can check in $O(36^{\mathbf{bw}} \cdot \ell_0^2 \cdot |V(G)|)$ steps if G contains at least ℓ_0 cycles. Note that the constant can probably be optimized with fast matrix multiplication for example but it is outside the scope of this paper. □

5.2 The lower bound for Planar Cycle Packing

Theorem 3. *The* Planar Cycle Packing *problem cannot be solved in time* $2^{o(\sqrt{n})} \cdot n^{O(1)}$ *unless the ETH fails.*

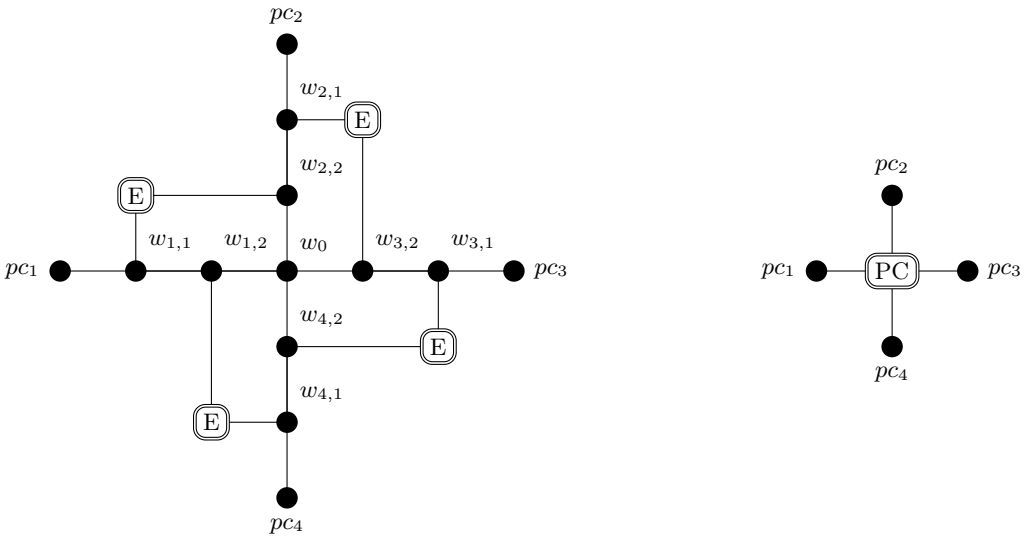
Proof: To prove this theorem, we reduce from Planar 3-colorability where the input graph has maximum degree at most 5. Let $G = (V, E)$ be a planar graph with maximum degree at most 5 with $V = \{v_1, \dots, v_n\}$. We proceed to construct a planar graph H together with a planar embedding.

In this proof, we abuse notation and say that we *ask* for x cycles in a gadget to say that the number of cycles we are looking for in the Planar



(a) The expel gadget. (b) Its representation. (c) The double-expel gadget.

Fig. 7. The expel gadget and the double-expel gadget.

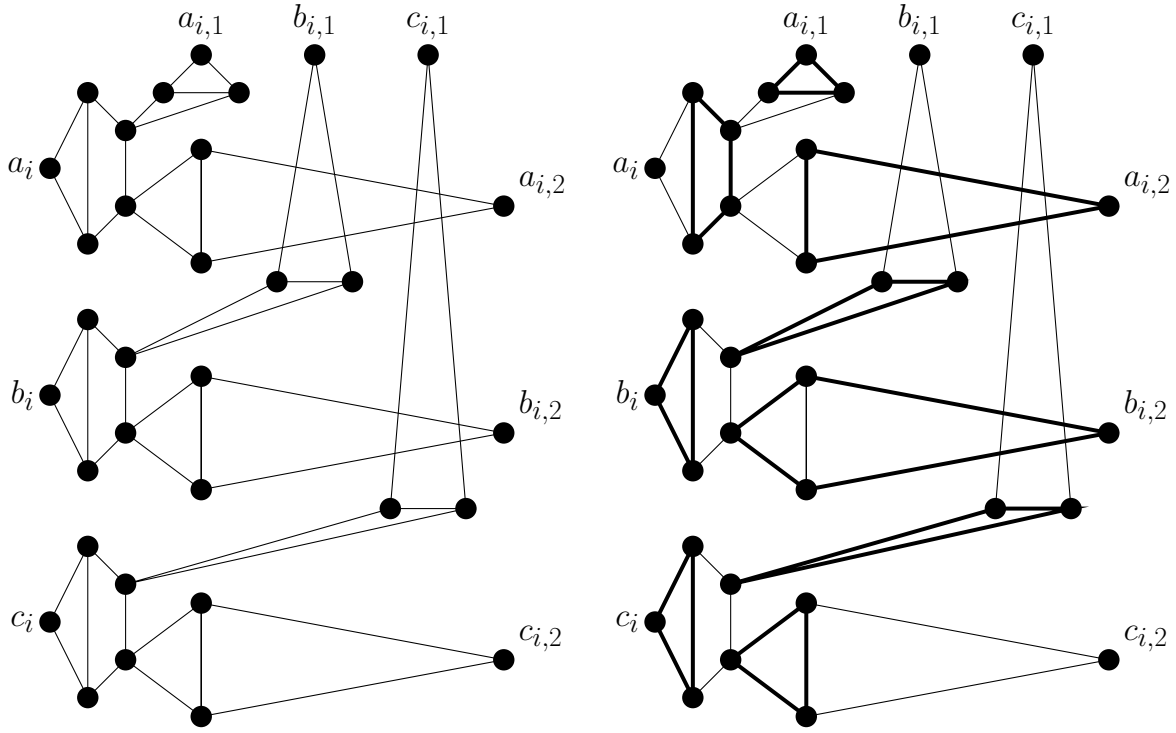


The gadget. The representation.

Fig. 8. Path-crossing gadget.

Cycle Packing problem is increased by x. We will ask for a certain number of cycles in each of the introduced gadgets, which by construction will lead to a set of cycles of maximum cardinality in H.

We start by introducing some gadgets. For each $i \in [n]$, corresponding to the vertices $v_1, \ldots, v_n$ of G, we add to H the SC_i-gadget depicted in Fig. 6. More precisely, $SC_i = (\{a_i, b_i, c_i, u_{i,0}, u_{i,1}, u_{i,2}, u_{i,3}\}, \{(u_{i,0}, u_{i,1}), (u_{i,0}, u_{i,2}), (u_{i,0}, u_{i,3}), (a_i, u_{i,1}), (a_i, u_{i,2}), (b_i, u_{i,2}), (b_i, u_{i,3}), (c_i, u_{i,1}), (c_i, u_{i,3})\})$. We ask for a cycle inside this gadget. This cycle imposes that at least one of the vertices $\{a_i, b_i, c_i\}$, named a *selected vertex* of the SC_i-gadget, is used by the inner cycle and leaves the possibility that the two others are free. The intended meaning of each SC_i-gadget is as follows. The three vertices a_i, b_i, and c_i correspond to the three colors in the 3-coloring of G, namely a, b, and c. If for instance a_i



(a) The gadget. (b) An example.

Fig. 9. Bifurcate gadget: To keep planarity, there is a path-crossing gadget in each edge intersection.

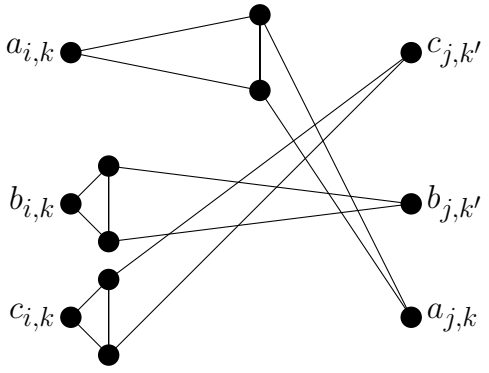


Fig. 10. Edge gadget: To keep planarity, there is a path-crossing gadget in each edge intersection.

is a selected vertex for i, it will imply that vertex v_i can be colored with color a.

Therefore, each SC_i-gadget defines the available colors for vertex v_i, which we call the *color output* of vertex v_i. In order to construct H that define a valid 3-coloring of G, we need to propagate the color output of v_i

as many times as the degree of v_i in G. For this, we introduce a gadget call *bifurcate* gadget. Before proceeding to the description of the gadget, let us describe its intended function. The objective is, starting with the vertices a_i, b_i, and c_i of the SC_i-gadget, to construct a set of $\deg_G(v_i)$ triples $\{a_{i,k}, b_{i,k}, c_{i,k}\}$ for $1 \leqslant k \leqslant \deg_G(v_i)$ such that in each triple there will be again at least one selected vertex, defined by the cycles that we will construct in the bifurcate gadgets. Note that in the SC_i-gadget the choice of a selected vertex in each triple $\{a_{i,k}, b_{i,k}, c_{i,k}\}$ naturally defines a color output for vertex v_i. The crucial property of the gadget is that the intersection of the color outputs given by all the triples is non-empty if and only if the graph H contains enough vertex-disjoint cycles. In other words, the existence of the appropriate number of vertex-disjoint cycles in H will define an available color for each vertex v_i in G.

We now proceed to the construction of the bifurcate gadget. First we need to introduce three other auxiliary gadgets. The first two ones, called *expel* and *double-expel* gadgets, are depicted in Fig. 7. Formally, for two vertices u and u', the expel gadget is defined as $EG_{u,u'} = (\{u, u', v, v'\}, \{(u, v), (u, v'), (u', v), (u', v'), (v, v')\})$, and we ask for a cycle inside each such expel gadget. This gadget ensures that if u is in another cycle, then u' is necessarily used by the internal cycle and vice-versa. Similarly, the double-expel gadget for three vertices u, u', and u'' is defined as $DEG_{u,u',u''} = (\{u, u', u'', v, v'\}, \{(u, v), (u, v'), (u', v), (u'', v'), (u', u''), (v, v')\})$, and we also ask for a cycle inside each such gadget. This gadget ensures that if u is in another cycle, then u' and u'' are necessarily used by the internal cycle and that if u' or u'' are in an external cycle, then u is necessarily used by the internal cycle.

As in our construction the edges of the expel gadgets will cross, we need a gadget that replaces each edge-crossing with a planar subgraph while preserving the existence of the original edges, in the sense that each of the crossing edges gets replaced by a path joining the endvertices of the original edge. This gadget is called *path-crossing* gadget and is depicted in Fig. 8. Formally the path-crossing gadget PCG is such that $\{pc_1, pc_2, pc_3, pc_4, w_0, w_{1,1}, w_{1,2}, w_{2,1}, w_{2,2}, w_{3,1}, w_{3,2}, w_{4,1}, w_{4,2}\} \subseteq V(PCG)$, $E(PCG)$ contains two paths $pc_1, w_{1,1}, w_{1,2}, w_0, w_{3,2}, w_{3,1}, pc_3$ and $pc_2, w_{2,1}, w_{2,2}, w_0, w_{4,2}, w_{4,1}, pc_4$, and we add four expel gadgets $EG_{w_{1,1},w_{2,2}}$, $EG_{w_{2,1},w_{3,2}}$, $EG_{w_{3,1},w_{4,2}}$, $EG_{w_{4,1},w_{1,2}}$ to PCG. We ask in this gadget only the four cycles asked by the expel gadgets. This gadget ensures that an external cycle that contain an edge from a path-crossing gadget should go *straight*, i.e., for all $\alpha \in [4]$, if the cycle arrives at a vertex pc_α it should exit by $pc_{(\alpha+1 \pmod 4)+1}$. If a cycle does not respect this property, we say that the cycle *turns* inside the path-crossing gadget. That is, the gadget preserves

the existence of the original crossing edges whenever there are no cycles that turn inside it. Note that, of course, both paths corresponding to the two original crossing edges cannot be used simultaneously by a set of cycles in the planar graph H. We can now define the bifurcate gadget, which is depicted in Fig. 9(a), and where each of the 12 edge-crossings should be replaced by a path-crossing gadget. Note that each bifurcate gadgets contains 6 expel and 3 double-expel gadgets. We ask in this gadget the 48 cycles of the path-crossing gadgets, the 3 cycles of the double expel gadgets and the 6 cycles of the expel gadgets. Note that, indeed, given a triple $\{a_i, b_i, c_i\}$ defining a color output for a vertex v_i, the cycles asked in the bifurcate gadget define two triples $\{a_{i,1}, b_{i,1}, c_{i,1}\}$ and $\{a_{i,2}, b_{i,2}, c_{i,2}\}$, which in turn define two color outputs compatible with the one defined by $\{a_i, b_i, c_i\}$, in the sense that there is a common available color for v_i. For example, in Fig. 9(b) vertex a_i is the only selected vertex of $\{a_i, b_i, c_i\}$ (given by the corresponding SC_i-gadget which is not shown in the figure for the sake of visibility), and the bold cycles define the selected vertices for the triples $\{a_{i,1}, b_{i,1}, c_{i,1}\}$ and $\{a_{i,2}, b_{i,2}, c_{i,2}\}$. Note that color a is available for the three triples. We would like to stress that there are other choices of a maximum-cardinality set of cycles in the bifurcate gadget of Fig. 9(b) but all of them yield color a available. For each vertex v_i, we need as many triples $\{a_{i,k}, b_{i,k}, c_{i,k}\}$ as $\deg_G(v_i)$. For that, we concatenate the bifurcate gadgets $deg_G(v_i) - 1$ times in the following way. Inductively, we consider the triple $\{a_{i,2}, b_{i,2}, c_{i,2}\}$ of Fig. 9(a) as the original triple $\{a_i, b_i, c_i\}$, and plug another bifurcate gadget starting from this triple.

With the gadgets defined so far, we have a representation of the colored vertices of G in H. We now proceed to capture the edges of G in H. For this, we introduce for each $\{v_i, v_j\} \in E$, $i, j \in [n]$, an *edge* gadget depicted in Fig. 10, where all the 12 edge-crossings should be replaced by a path-crossing gadget. We ask in this gadget 51 new cycles (3 for the expel gadgets and 48 for the path-crossing gadgets). We plug one side of this gadget to an $\{a_{i,k}, b_{i,k}, c_{i,k}\}$ triple defining a color output of v_i and the other side to an $\{a_{j,k'}, b_{j,k'}, c_{j,k'}\}$ triple defining a color output of v_j. This gadget ensures that the intersection of the two color outputs is empty.

Claim. If we ask for the maximum number of cycles inside H, then each expel gadget, double-expel gadget, and SC-gadget contains a cycle and each cycle is contained inside such a gadget.

Proof: In this proof, we say that a cycle C *kills* an another cycle C' if, for all set S of vertex-disjoint cycles containing C, then $(S \backslash \{C\}) \cup \{C'\}$ is also a set of vertex-disjoint cycles. First note that any cycle entirely contained in an expel and or a double-expel gadget should use both vertices v and v'. Also note that if some external cycle that is not entirely contained in an

expel gadget or a double-expel gadget uses the vertex v or v' of an expel or a double-expel gadget, then it also use the vertex u (or u and u'') and we are not able to find an internal cycle anymore. Therefore, any external cycle containing v or v'kills the cycle $\{u, v, v'\}$ or $\{u', u'', v', v''\}$.

Note that if an external cycle of a path-crossing gadget turn inside it, then without loss of generality it uses a path of the form $pc_1, w_{1,1}, w_{1,2}, w_0, w_{2,2}, w_{2,1}, pc_2$ inside the path-crossing gadget. This external cycle kills the cycle inside the expel gadget between $w_{1,1}$ and $w_{2,2}$. Moreover, another disjoint external cycle turning in the same path-crossing gadget kills the cycle inside the expel gadget between $w_{3,1}$ and $w_{4,2}$.

Let C be a cycle in H that is not contained in only one expel gadget, double-expel gadget, or SC_i-gadget. Because of the previous remarks, we have that C cannot turn in two path-crossing gadgets and if it turns in any path-crossing gadget then it uses at least two expel or double-expel gadgets and kills their internal cycles. In both configurations, the introduction of C in the solution decreases the number of cycles that we can find in H.

The only choice remaining for C is to turn only once in a path-crossing gadget. If it happens inside a bifurcate gadget, then C uses vertices of two expel gadgets $expel_1$ and $expel_2$ corresponding to two different colors. The only way to connect vertices corresponding ti different colors outside the path-crossing gadget is by using an SC_i-gadget. So C kills the cycles of $expel_1$ and $expel_2$, or it could also use a path leading to an edge gadget. If C turns in a path-crossing gadget inside an edge gadget, then the the analysis is similar but there is an extra case where the edge gadget representing the edge between v_i and v_j is directly plugged in the SC_j-gadget. Just note that in this case none of the vertices $\{a_i, b_i, c_i\}$ can be a selected vertex with the current set of cycles we ask for,, and therefore in order to allow it we need to decrease the number of cycles in the solution. □

If we are given a solution of Planar Cycle Packing in H, then for each $i \in [n]$, the selection of a cycle in the SC_i-gadget selects a color for v_i, that can be any color that is shared by all color outputs of v_i, and the edge gadgets ensure that two adjacent vertices are in two different color classes. So in this way we obtain a solution of Planar 3-colorability in G.

Conversely, given a solution of Planar 3-colorability in G, we construct a solution of Planar Cycle Packing in H as follows. For each $i \in [n]$ we choose in the SC_i-gadget the cycle of length 4 that contains $u_{i,0}$ and the vertex in $\{a_i, b_i, c_i\}$ that corresponds to the color of v_i. We also choose in the bifurcates gadgets the cycle that select vertices in $\{a_{i,1}, b_{i,1}, c_{i,1}, a_{i,2}, b_{i,2}, c_{i,2}\}$ that lead to two identical color outputs that is exactly the color output of $\{a_i, b_i, c_i\}$. This selection keeps the property that the color output of $\{a_i, b_i, v_i\}$ is contain in the color output of $\{a_{i,1}, b_{i,1}, c_{i,1}\}$

and in the color output of $\{a_{i,2}, b_{i,2}, c_{i,2}\}$, and leaves free as many vertices as possible for other cycles in other gadgets. Inside the edge gadget representing $\{v_i, v_j\} \in E$, we select the three cycles that are allowed by the free vertices. We complete our cycle selection by selecting a cycle in each expel gadget contained in a path-crossing gadget. By the claim, this choice leads to an optimal solution of Planar Cycle Packing in H.

As the degree of each vertex in G is bounded by 5, the number of gadgets we introduce for each $v_i \in V(G)$ to construct H is also bounded by a constant, so the total number of vertices of H is linear in the number of vertices of G. Therefore if we could solve Planar Cycle Packing in time $2^{o(\sqrt{n})} \cdot n^{O(1)}$ then we could also solve Planar 3-coloring in time $2^{o(\sqrt{n})} \cdot n^{O(1)}$, which is impossible by Theorem 1 unless the ETH fails. The theorem follows. □

From Theorem 3 we obtain the following corollary.

Corollary 2. *The* Planar Cycle Packing *problem cannot be solved in time* $2^{o(\mathbf{tw})} \cdot n^{O(1)}$ *unless the ETH fails.*

5.3 Maximally Disconnected Feedback Vertex Set

We note that we can construct other problem of Type 2 that the ones introduced in [4]. We create Maximally Disconnected Feedback Vertex Set and prove that it is a problem of Type 2.

Maximally Disconnected Feedback Vertex Set
Input: A graph $G = (V, E)$ and integers ℓ and r.
Parameter: tw.
Question: Does G contain a feedback vertex set of size at most ℓ that induces at least r connected components?

Lemma 1. Maximally Disconnected Feedback Vertex Set *cannot be soved in time* $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{O(1)}$ *unless the ETH fails.*

Proof: We reduce from $k \times k$ hitting set. The reduction is the same that the one given in Annex C.1 p60 of [4] where we just have to redefine the force gadget and the one-in-many gadget. □

The use of previous dynamic programming techniques describe in the introduction and in Theorem 2 gives the two following lemma.

Lemma 2. Maximally Disconnected Feedback Vertex Set *can be soved in time* $2^{O(\mathbf{tw} \log \mathbf{tw})} \cdot n^{O(1)}$ *unless the ETH fails.*

Lemma 3. Planar Maximally Disconnected Feedback Vertex Set *can be soved in time* $2^{O(\mathbf{tw})} \cdot n^{O(1)}$ *unless the ETH fails.*

6 Problems of Type 3

In this section we prove that there exist problems of Type 3. More precisely, we prove that Monochromatic Disjoint Paths is of Type 3. For this problem, we need to introduce some definition. Let $G = (V, E)$ be a graph, let k be an integer, and $c : V \to \{0, \ldots, k\}$ a color function. Two colors c_1 and c_2 in $\{0, \ldots, k\}$ are *compatible* and we note $c_1 \equiv c_2$ if $c_1 = 0$, $c_2 = 0$, or $c_1 = c_2$. Let $P = x_0 \ldots x_m$ be a path in G. We say that P is a *monochromatic path* if for all $i \neq j$, $c(x_i)$ and $c(x_j)$ are two compatible colors. We let $c(P) = \max_{i \in [m]}(c(x_i))$. We say that P is *colored* x if $x = c(P)$. We say that two monochromatic paths P and P' are *color-compatible* if $c(P) \equiv c(P')$.

Monochromatic Disjoints Paths
Input: A graph $G = (V, E)$ of treewidth $\mathbf{tw}$, a color function $\gamma : V \to \{0, \ldots, \mathbf{tw}\}$, $m \in \mathbb{N}$ and $\mathcal{N} = \{\mathcal{N}_i = \{s_i, t_i\} | i \in \{1, \ldots, m\}, s_i, t_i \in V\}$.
Parameter: tw.
Question: Does G contain m pairwise vertex-disjoint monochromatic paths from s_i to t_i, $i \in \{1, \ldots, m\}$?

6.1 Algorithm for Monochromatic Disjoint Paths

Lemma 4. Monochromatic Disjoint Paths *can be solved in time* $2^{O(\mathbf{tw} \log \mathbf{tw})} \cdot n^{O(1)}$.

Proof: The following algorithm is an adaptation of the algorithm given in [23] for Disjoint Path.

Let G be a colored graph and let $\gamma : V(G) \to \{0, \ldots, \mathbf{tw}\}$ be the coloring of the graph. Let $(\mathcal{N}_i = \{s_i, t_i\})_{i \in \{1 \ldots m\}}$ be the m paths we are looking for and let (T, μ, π) a sc-decomposition of G of width $\mathbf{bw}(G)$. As in [10], we root T by arbitrarily choosing an edge e and subdivide it by insering a new node s. Let e' and e'' be the new edges and set $\mathbf{mid}(e') = \mathbf{mid}(e'') = \mathbf{mid}(e)$. Create a new node root r, connect it to s, by an edge e_r, and set $\mathbf{mid}(e_r) = \emptyset$. The root e_r is not consider as a leaf.

Let now e be an edge of T, let $X, P \subseteq \mathbf{mid}(e)$ with $X \cap P = \emptyset$, and M, L be two disjoint matchings of $\mathbf{mid}(e) \backslash (X \cup P)$. Let $\gamma_0 : P \cup V[M] \cup V[L] \to \{0, \ldots, \mathbf{tw}\}$ be color functions. Let $\varphi : P \to \{1, \ldots, m\}$ an injective function. Intuitively, we want to keep memories of path inside G_e, so P stands for virtual source of a terminals, M stands for pair of virtual sources that should be links, L stands for pair $\{x, y\}$ such that there is a path in G_e that link x and y, and X stands for vertices that are already inside a path or both an endpoint and a terminal. We say that

$mdp(G_e, \mathbf{mid}(e), X, P, M, L, c_0, \varphi) = true$ if all the following conditions are fulfilled.

- For all $\{s_i, t_i\}$ in $\mathcal{N} \cap V(G_e)^2$
  - There exists a monochromatic path $s_i \dots t_i$ in G_e or
  - There exist $\{s'_i, t'_i\} \in M$ and two monochromatic paths in G_e $s_i \dots s'_i$ colored $\gamma_0(s'_i)$ and $t_i \dots t'_i$ colored $\gamma_0(t'_i)$ with $\gamma_0(s'_i) \equiv \gamma_0(t'_i)$.
- For all $\{s_i, t_i\}$ in $\mathcal{N}$, such that $s_i \in V(G_e)$ and $t_i \notin V(G_e)$ or vice-versa,
  - There exist $s'_i \in P$ such that $\varphi(s'_i) = i$ and a monochromatic path $s_i = v_0 \dots v_k = s'_i$ colored $c_0(s'_i)$.
- For all $\{x_i, y_i\}$ in L
  - There exists a monochromatic path $x_i \dots y_i$ colored $\max(\gamma_0(x_i), \gamma_0(y_i))$ in G_e.
- All these paths are vertex-disjoint and all vertices in S with degree at least 2 are in X.

Let $S_1 = (X_1, P_1, M_1, L_1, \gamma_1, \varphi_1)$, and $S_2 = (X_2, P_2, M_2, L_2, \gamma_2, \varphi_2)$ with $X_1, X_2, P_1, P_2, \dots$ defined as above. We define $G[S_1] = (P_1 \cup V[M_1] \cup V[L_1], \{\{x, y\} \in L_1\})$ and colored by γ_1 and similarly define $G[S_2]$. We say that $G[S_1, S_2]$ is *defined* if for all $x \in V(G[S_1]) \cap V(G[S_2])$, $c_1(x) \equiv c_2(x)$, $X_1 \cap V(G[S_2]) = X_2 \cap V(G[S_1]) = X_1 \cap X_2 = \emptyset$, and we define $G[S_1, S_2] = G[S_1] \cup G[S_2]$ and colored by γ_{12} defined such that for all $x \in V(G[S_1, S_2])$, $\gamma_{12} = \max(\gamma_1(x), \gamma_2(x))$. Otherwise we say that $G[S_1, S_2]$ is *undefined*.

Let $e \in E(T)$, we define $\mathcal{R}_e = \{(X, P, M, L, \gamma, \varphi) | X \subseteq \mathbf{mid}(e), P \subseteq \mathbf{mid}(e), X \cap P = \emptyset$, M and L are disjoint matchings on $\mathbf{mid}(e) \backslash (X \cup P)$, $V[M] \cap V[L] = \emptyset$ and $mdp(G_e, \mathbf{mid}(e), X, P, M, L, \gamma, \varphi) = true\}$. We want to know if $(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset) \in \mathcal{R}_{e_r}$. We can compute $\mathcal{R}_e$, for each $e \in E(T)$, as follows :

- if e is a leaf, then $G_e = (\{x, y\}, \{(x, y)\}$
  - if $\{x, y\} \in \mathcal{N}$ then $\mathcal{R}_e = \{(\{x, y\}, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)\}$
  - if $x \in \mathcal{N}_i, y \in \mathcal{N}_j$, $i \neq j$ then $\mathcal{R}_e = \{(\emptyset, \{x, y\}, \emptyset, \emptyset, \{(x, \gamma(x)), (y, \gamma(y))\}, \{(x, i), (y, j)\})\}$
  - if $x \in \mathcal{N}_i$ and $\forall j \in \{1, \dots, m\}, y \notin \mathcal{N}_j$ and $\gamma(x) \not\equiv \gamma(y)$ then $\mathcal{R}_e = \{(\emptyset, \{x\}, \emptyset, \emptyset, \{(x, \gamma(x)\}, \{(x, i)\})\}$
  - if $x \in \mathcal{N}_i$ and $\forall j \in \{1, \dots, m\}, y \notin \mathcal{N}_j$ and $\gamma(x) \equiv \gamma(y)$ then $\mathcal{R}_e = \{(\emptyset, \{x\}, \emptyset, \emptyset, \{(x, \gamma(x)\}, \{(x, i)\}), (\{x\}, \{y\}, \emptyset, \emptyset, \{(y, \max(\gamma(x), \gamma(y)))\}, \{(y, i)\})\}$
- if e is not a leaf, let e_1 and e_2 be the two children of e in $E(T)$. We construct $\mathcal{R}_e$ as the set of all 6-uplets $(X, P, M, L, \gamma_0, \varphi)$ such that there exist $S_1 = (X_1, P_1, M_1, L_1, \gamma_1, \varphi_1) \in \mathcal{R}_{e_1}$, and $S_2 = (X_2, P_2, M_2, L_2, \gamma_2, \varphi_2) \in \mathcal{R}_{e_2}$ fulfilling the following properties:
  - $H = G[S_1, S_2]$ is defined

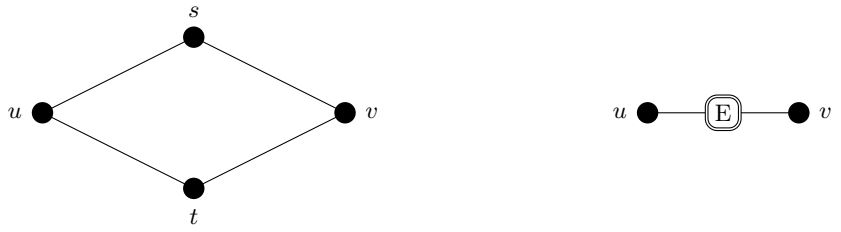
- For all $\{x_i, y_i\}$ in L, there exists a monochromatic path $x_i \dots y_i$ in H and we have $\gamma_0(x_i) = \gamma_0(y_i) = \gamma_{12}(x_i \dots y_i)$.
- All vertices in $\mathbf{mid}(e)$ of degree at least 2 in $G[S_1, S_2]$ are in X.
- For all $\{v, w\}$ in M_i, $i \in \{1, 2\}$, there is a monochromatic color compatible path from v to w in $G[S_1, S_2]$ or two vertices $\{v', w'\} \in M$, and two monochromatic color compatible paths $v \dots v'$ and $w \dots w'$ and we have $\gamma_0(v') = \gamma_0(w') = \max(\gamma_{12}(v \dots v'), \gamma_{12}(w \dots w'))$.
- For all i in $\{1, 2\}$, For all v in P_i, there exist $w \in P$ and a monochromatic color compatible path $v \dots w$ or there are $w \in P_{3-i}$ such that $\varphi_i(v) = \varphi_{3-i}(w)$ and a monochromatic path from $v \dots w$ such that $\gamma_0(w) = \gamma_{12}(v \dots w)$.
- All these paths are pairwise vertex-disjoint.

As in the graph $G[S_1, S_2]$, by construction, all vertices have degree at most two, we can check all the previous properties in polynomial time because we just have to compare two sets or follow a path in $G[S_1, S_2]$ to check each property. So we can compute each element of $\mathcal{R}_e$ in $poly(\mathbf{mid}(e))$ steps. As $(X, P, V[M], V[L])$ forms a partition of a subset of $\mathbf{mid}(e)$, there are at most $5^{\mathbf{mid}(e)}$ such sets. There are at most $\mathbf{tw} + 1$ colors and at most $(\mathbf{tw} + 1)^{\mathbf{mid}(e)}$ choices for γ_0. As $|\{\varphi(x) | x \in P\}| \leqslant |P| \leqslant \mathbf{mid}(e)$, then there are at most $\mathbf{mid}(e)^{\mathbf{mid}(e)}$ such different color functions φ possible. As $\mathbf{bw} - 1 \leqslant \mathbf{tw}$ we have that for all e in $E(T)$, $|\mathbf{mid}(e)| \leqslant \mathbf{tw}$ and so for all e in $E(T)$, $|\mathcal{R}_e| \leqslant 5^{\mathbf{tw}} \cdot (\mathbf{tw} + 1)^{2\mathbf{tw}}$. As for each $e \in E(T)$ such that e is not a leaf, we have to merge the tables of the two children e_1 and e_2 of e then the above dynamic programming algorithm can solve Monochromatic Disjoint Paths in $O(25^{\mathbf{tw}} \cdot (\mathbf{tw} + 1)^{4\mathbf{tw}} \cdot |V(G)|)$ steps. The lemma follows. Note that the constant can probably be optimized with fast matrix multiplication for example but it is outside the scope of this paper. □

6.2 Lower bound for Planar Monochromatic Disjoint Paths

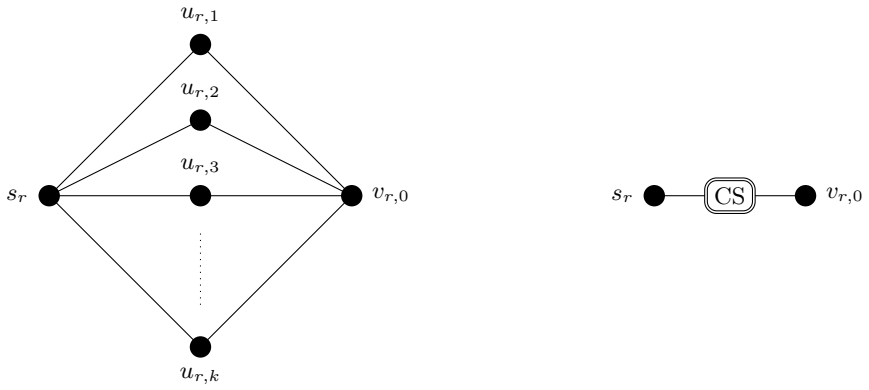
Theorem 4. Planar Monochromatic Disjoint Paths *cannot be solved in time* $2^{o(\mathbf{pw} \log \mathbf{pw})} \cdot n^{O(1)}$ *unless the ETH fails, where* **pw** *stands for the pathwidth of the input graph.*

Proof: We reduce from the $k \times k$ hitting set problem. We do not use the original version but the one considered in [17]:



The gadget. The representation.

Fig. 11. Expel gadget.



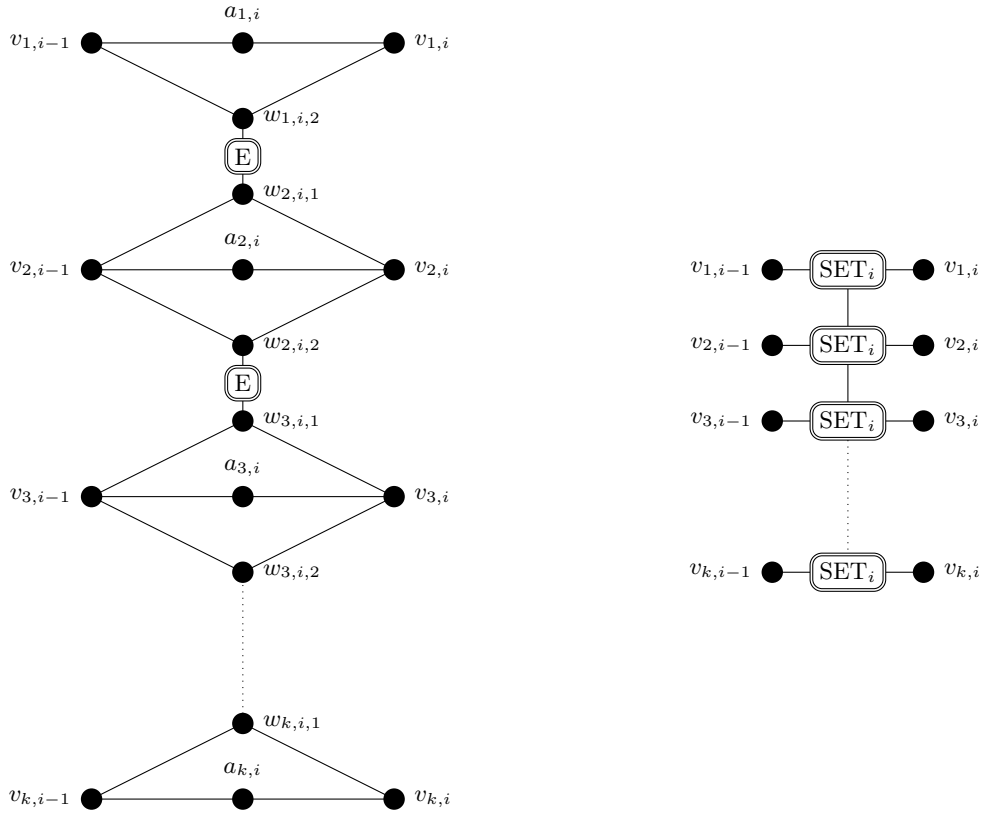
The gadget. The representation.

Fig. 12. Color-selection gadget where $u_{r,i}$ is colored c_i for each $i \in [k]$.

$k \times k$ HITTING SET
Input: A family of sets $S_1, S_2, \ldots, S_m \subseteq [k] \times [k]$, such that each set contains at most one element from each row of $[k] \times [k]$.
Parameter: k.
Question: Is there a set S containing exactly one element of each row such that $S \cap S_i \neq \emptyset$ for any $1 \leqslant i \leqslant m$?

Let k be an integer and $S_1, S_2, \ldots, S_m \subseteq [k] \times [k]$ such that each set contains at most one element from each row of $[k] \times [k]$. We will first present an overview of the reduction with all the involved gadgets and then we will provide a formal definition of the constructed planar graph G.

We construct a gadget for each row $\{r\} \times [k]$, $r \in [k]$, that selects the unique pair p of S of this row. First, for each $r \in [k]$, we introduce two new vertices s_r and t_r, a request $\{s_r, t_r\}$, $m+1$ vertices $v_{r,i}$, $i \in \{0, \ldots, m\}$ and $m+2$ edges $\{e_{r,0} = (s_r, v_{r,0})\} \cup \{e_{r,i} = (v_{r,i-1}, v_{r,i}) | i \in \{1, \ldots, m\}\} \cup \{e_{r,m+1} = (v_{r,m}, t_r)\}$. So we have a path with $m+2$ edges between s_r and t_r.



The gadget. The representation.

Fig. 13. Set gadget.

Each edge of these paths, except the last one, will be replaced by an appropriate gadget. Namely, for each $r \in [k]$, we replace the edge $e_{r,0}$ with the gadget depicted in Fig. 12, which we call *color-selection* gadget. In this figure, $u_{r,i}$ is colored i. The color used by the path from s_r to t_r in the color-selection gadget will define the pair of the solution of S in the row $\{r\} \times [k]$.

Now that we have described the gadgets that allow to define S, we need to ensure that $S \cap S_i \neq \emptyset$ for any $i \in [m]$. For this, we need the gadget depicted in fig. 11, which we call *expel* gadget. Each time we introduce this gadet, we add the request $\{s, t\}$. This new path requested uses either vertex u or vertex v, so only one of these vertices can be used by other paths. For each $i \in [m]$, we replace all the edges $\{e_{r,i} | r \in [k]\}$ with the gadget depicted in Fig. 13, which we call *set* gadget. In this figure, $a_{r,i}$ is such that if $(\{r\} \times [k]) \cap S_i = \{\{r, c_{r,i}\}\}$ then $a_{r,i}$ is colored $c_{r,i}$, and if $(\{r\} \times [k]) \cap S_i = \emptyset$ then vertex $a_{r,i}$ is removed from the gadget. This

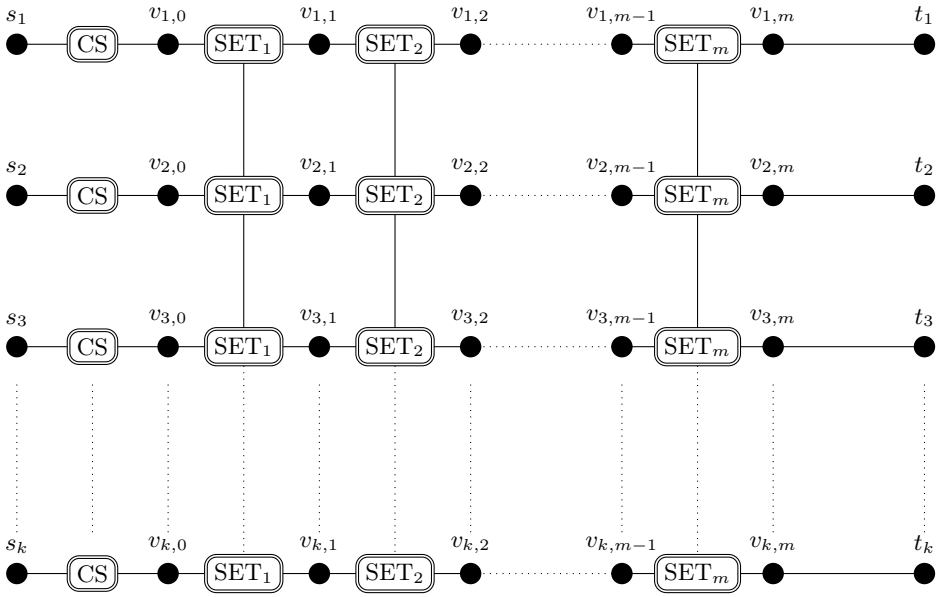


Fig. 14. Final graph G in the reduction of Theorem 4

completes the construction of the graph G, which is depicted in Fig. 14. Note that G is indeed planar.

Formally, the graph we obtain is $G = (V, E)$ where $V = \{s_r | r \in [k]\} \cup \{t_r | r \in [k]\} \cup \{v_{r,i} | r \in [k], i \in \{0, m\}\} \cup \{u_{r,c} | r \in [k], c \in [k]\} \cup (\{w_{r,i,b} | r \in [k], i \in [m], i \in \{1, 2\}\} \backslash \{w_{r,i,b} | i \in [m], (r, b) \in \{(1, 1), (k, 2)\}\}) \cup \{s_{r,i} | r \in [k-1], i \in [m]\} \cup \{t_{r,i} | r \in [k-1], i \in [m]\} \cup \{a_{r,i} | \exists c \in [k], (r, c) \in S_i$ and $E = \{\{s_r, u_{r,c}\} \in V^2 | r \in [k], c \in [k]\} \cup \{\{u_{r,c}, v_{r,0}\} \in V^2 | r \in [k], c \in [k]\} \cup \{\{v_{r,i-1}, w_{r,i,b}\} \in V^2 | r \in [k], i \in [m], b \in \{1, 2\}\} \cup \{\{w_{r,i,b}, v_{r,i}\} \in V^2 | r \in [k], i \in [m], b \in \{1, 2\}\} \cup \{\{v_{r,i-1}, a_{r,i}\} \in V^2 | r \in [k], i \in [m]\} \cup \{\{a_{r,i}, v_{r,i}\} \in V^2 | r \in [k], i \in [m]\} \cup \{\{v_{r,m}, t_r\} \in V^2 | r \in [k]\} \cup \{\{s_{r,i}, w_{r,i,2}\} \in V^2 | r \in [k-1], i \in [m]\} \cup \{\{s_{r,i}, w_{r+1,i,1}\} \in V^2 | r \in [k-1], i \in [m]\} \cup \{\{t_{r,i}, w_{r,i,2}\} \in V^2 | r \in [k-1], i \in [m]\} \cup \{\{t_{r,i}, w_{r+1,i,1}\} \in V^2 | r \in [k-1], i \in [m]\}$. The color function γ of G is defined such that for each $r \in [k]$ and $c \in [k]$, $\gamma(u_{r,c}) = c$, and for each $i \in [m]$ and $(r, c) \in S_i$, $\gamma(a_{r,i}) = c$. For any other vertex $v \in V(G)$, we set $\gamma(v) = 0$.

The input of Planar Monochromatic Disjoint Paths is the planar graph G, the color function γ and the $k + (k-1) \cdot m$ requests $\mathcal{N} = \{\{s_r, t_r\} | r \in [k]\} \cup \{\{s_{r,i}, t_{r,i}\} | r \in [k-1], i \in [m]\}$, the second set corresponding to the requests introduced by the expel gadgets.

Note that because of the expel gadgets, the request $\{s_r, t_r\}$ imposes a path between $v_{r,i-1}$ and $v_{r,i}$ for each $r \in [k]$. Note also that because of the expel gadgets, at least one of the paths between $v_{r,i-1}$ and $v_{r,i}$ should use an $a_{r,i}$ vertex, as otherwise at least two paths would intersect. Conversely,

if one path uses an $a_{r,i}$ vertex, then we can find all the desired paths in the corresponding set gadgets by using the $w_{r,i,b}$ vertices.

Given a solution of Planar Monochromatic Disjoint Paths in G, we can construct a solution of $k \times k$ hitting set $S = \{(r,c)|r \in [k]$, such that the path from s_r to t_r is colored with color $c\}$. We have that S contains exactly one element of each row, so we just have to check if $S \cap S_i \neq \emptyset$ for each $i \in [m]$. Because of the property of the set gadgets mentioned above, for each $i \in [m]$, the set gadget labeled i ensures that $S \cap S_i \neq \emptyset$.

Conversely, given a solution S of $k \times k$ hitting set, for each $\{r,c\} \in S$ we color the path from s_r to t_r with color c. We assign an arbitrary coloring to the other paths. For each $i \in [m]$, we take $\{r,c\} \in S \cap S_i$ and in the set gadget labeled i, we impose that the path from $v_{r,i-1}$ to $v_{r,i}$ uses vertex $a_{r,i}$. By using the $w_{r,i,b}$ vertices for the other paths, we find the desired $k + (k-1) \cdot m$ monochromatic paths.

Let us now argue about the pathwidth of G. We define for each $r, c \in [k]$ the bag $B_{0,r,c} = \{s_{r'}|r' \in [k]\} \cup \{v_{r',0}|r' \in [k]\} \cup \{u_{r,c}\}$, for each $i \in [m]$, the bag $B_i = \{v_{r,i-1}|r \in [k]\} \cup \{v_{r,i}|r \in [k]\} \cup \{a_{r,i} \in V(G)|r \in [k]\} \cup \{w_{r,i,b} \in V(G)|r \in [k], b \in [2]\} \cup \{s_{r,i}|r \in [m-1]\} \cup \{t_{r,i}|r \in [m-1]\}$, and the bag $B_{m+1} = \{v_{r,m}|r \in [k]\} \cup \{t_r|r \in [k]\}$. We note that the size of each bag is at most $2 \cdot (k-1) + 5 \cdot k - 2 = O(k)$. A path decomposition of G consists of all bag $B_{0,r,c}$, $r, c \in [k]$ and B_i, $i \in \{1, \ldots, m+1\}$ and edges $\{B_i, B_{i+1}\}$ for each $i \in [m]$, $\{B_{0,r,c}, B_{0,r,c+1}\}$ for $r \in [k]$, $c \in [k-1]$, $\{B_{0,r,k}, B_{0,r+1,1}\}$ for $r \in [k]$, and $\{B_{0,k,k}, B_1\}$. Therefore, as we have that $\mathbf{pw}(G) = O(k)$, the theorem follows. □

As any graph G satisfies $\mathbf{tw}(G) \leqslant \mathbf{pw}(G)$, from theorem 4 we obtain the following corollary

Corollary 3. Planar Monochromatic Disjoint Paths *cannot be solved in time* $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{O(1)}$ *unless the ETH fails.*

7 Other problems with lower bound in $2^{o(\mathbf{tw} \log \mathbf{tw})}$

Using ideas similar to those in the reduction for Monochromatic Disjoint Paths in Theorem 4, we can prove that an other planar problem cannot be solved in time $2^{o(\mathbf{tw} \log \mathbf{tw})}$ unless the ETH fails. Intuitively, we can prove lower bounds along the same ideas for problems where some information can be carried by paths that belong to the solution. For example, this is the case of the Planar Subgraph Isomorphism problem, which is defined as follow:

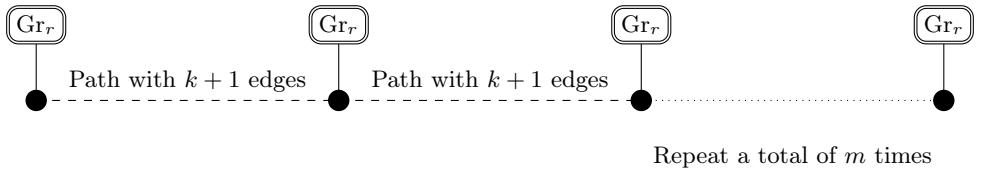


Fig. 15. Req_r with $m+1$ elements Gr_r.

Fig. 16. Expel gadget.

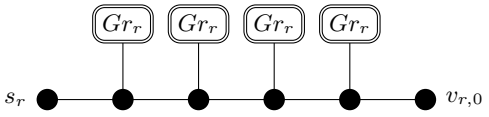


Fig. 17. The color-selection gadget.

Planar Subgraph Isomorphism
Input: Two planar graphs G and H.
Parameter: $\mathbf{tw} = \mathbf{tw}(G)$.
Question: Does G contains a subgraph isomorphic to H?

Lemma 5. Planar Subgraph Isomorphism *cannot be solved in time* $2^{o(\mathbf{tw}\log\mathbf{tw})} \cdot n^{O(1)}$ *unless ETH fails.*

Proof: We reduce again from $k \times k$ hitting set. Let k be an integer with $k \geqslant 3$ and $S_1, S_2, \ldots, S_m \subseteq [k] \times [k]$ such that each set contains at most one element from each row of $[k] \times [k]$. We construct a graph G similar to the one for Monochromatic Disjoint Path. We construct in the same time the graph H. In this section, we call each maximal connected subgraph of H a *request*. So H will be the union of all the requests we made. We modify the expel gadget, the color selection gadget, and the avoid path gadget. As in Monochromatic Disjoint Paths, we create a gadget for each row $\{r\} \times [k]$ and make a request, Req_r, depicted in Fig. 15, for each of these gadgets, that choose a path and simulates a color for each row. We add the set gadget that ensures that $S \cap S_i \neq \emptyset$ for each $i \in \{1 \ldots m\}$. For each $r \in [k]$, the Req_r simulates the path from s_r to t_r that appears in Monochromatic Disjoint Paths. The way it is insert in the color-

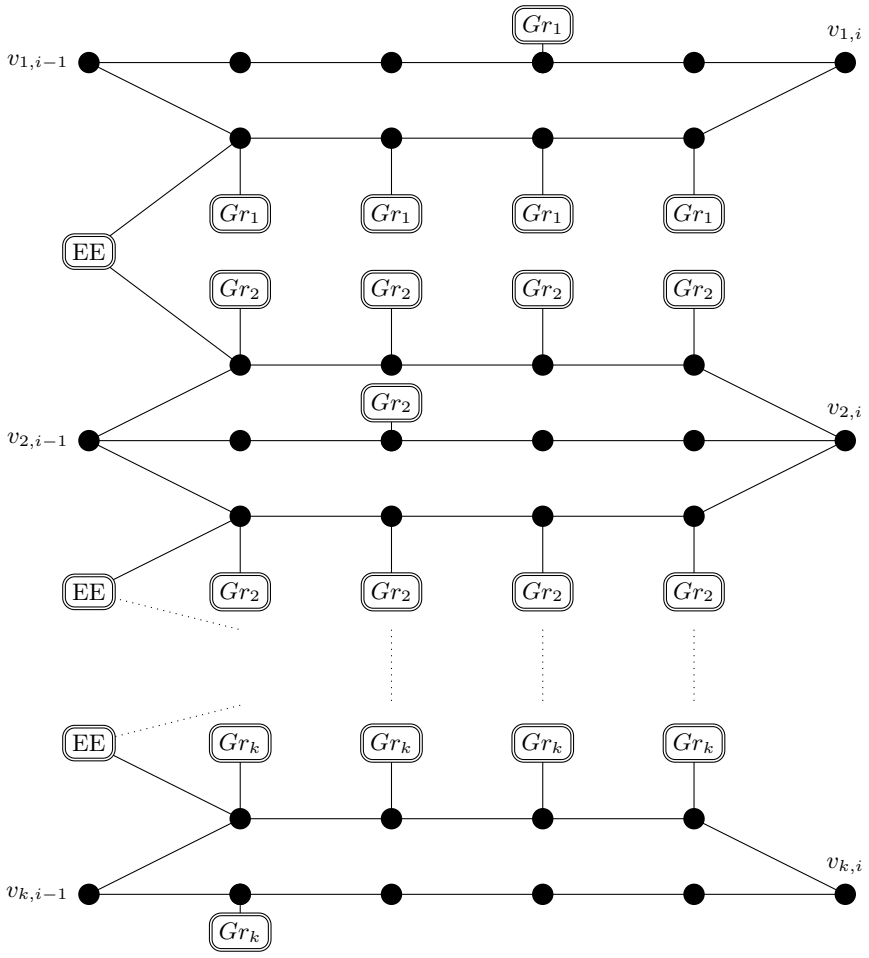


Fig. 18. Set gadget with $(1,3) \in S_1$, $(2,2) \in S_2$, ..., $(k,1) \in S_k$.

selection gadget simulates the color of the path $s_r \dots t_r$ and fix the paths it can use in the set gadgets as in Monochromatic Disjoint Paths.

More precisely, we define $k+1$ graphs that do not appear anywhere else in the graph. For it, we take a grid of size $k+2$ in which we remove a vertex. We name $Gr_1, Gr_2, \dots, Gr_k$ and Gr_g these grids. Gr_i, $i \in [k]$, identifies the subgraph corresponding to the row i, while Gr_g identifies the expel gadget requests. We redefine the *expel* gadget as depicted in Fig. 16 and the *color-selection* gadget as depicted in Fig. 17. We also redefine the *set* gadget as depicted in Fig. 18 similarly to Monochromatic Disjoint Paths but with the new expel gadget and a way to simulate in Planar Subgraph Isomorphism the color we have in Monochromatic Disjoint Paths.

The final graph we construct has the same shape depicted in Fig. 14 but with the new gadgets instead. We define the planar graph H to be the union of the $m \cdot (k-1)$ expel gadget requests shown in Fig. 16 and the request Req_r for each $r \in [k]$.

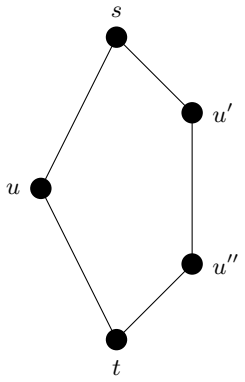


Fig. 19. The double-expel gadget.

All we have to specify is that we can use the path in the set gadget that simulates the colored path that appear in the reduction of Planar Monochromatic Disjoint Paths only if we choose the corresponding color in the color-selection gadget. This property is given by the fact that we look for a path where the identify grids are space by the same number of node each time and so, the color selected in the color-selection gadget is maintain in each set gadget.

If S is a solution of $k \times k$ hitting set, by starting to find the request Req_i that simulates the correct color we find a solution to Planar Subgraph Isomorphism as we have found a solution of Monochromatic Disjoint Paths. For the same reason that appear in the proof of Planar Monochromatic Disjoint Paths lower bound, a solution of Planar Subgraph Isomorphism defines a solution S of $k \times k$ hitting set. □

8 A Lower bounds for Planar Disjoint Paths

In this section we prove that, assuming the ETH, the Planar Disjoint Paths problem cannot be solved in time $2^{o(\mathbf{tw})} \cdot n^{O(1)}$.

Theorem 5. Planar Disjoint paths *cannot be solved in time* $2^{o(\sqrt{n})} \cdot n^{O(1)}$ *unless the ETH fails.*

Proof: We strongly follow the proof of Theorem 3. We reduce from Planar 3-Colorability where the input graph has maximum degree at most 5. Let $G = (V, E)$ be a planar graph with maximum degree at most 5 where $V = \{v_1, \ldots, v_n\}$. We proceed to construct a planar graph H together with a planar embedding. We construct the same graph as in the proof of Theorem 3 but where the gadgets are appropriately modified. We reuse the *expel* gadget depicted in Fig. 11 and for each expel gadget, we ask for a

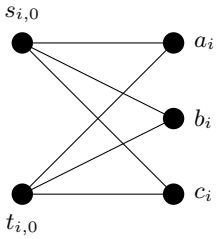


Fig. 20. The SCi-gadget: To keep planarity, there is a path-crossing gadget in each edge intersection.

path between s and t. We redefine the *double-expel* gadget as depicted in Fig. 19 and for each double-expel gadget, we ask for a path between s and t. We reuse the *path-crossing* gadget depicted in Fig. 8 and only ask for the paths contained in the expel gadgets. We can now redefine the SC$_i$-gadget depicted in Fig. 20, where each edge intersection is replaced by a path-crossing gadget. For each SC$_i$-gadget we ask for a path between $s_{i,0}$ and $t_{i,0}$. We also define the *bifurcate* gadget depicted in Fig. 21 and for each bifurcate gadget, we ask for a path between $s_{i,k}$ and $t_{i,k}$ for $k \in [9]$. We finally define the *edge* gadget depicted in Fig. 22 and for each edge gadget, we ask for a path between $s_{i,j,k}$ and $t_{i,j,k}$ for $k \in [3]$. We can easily check that these gadgets preserve the same properties as the corresponding ones in the proof of Theorem 3. Moreover, it is easy to see that a path cannot turn in a path-crossing gadget. This completes the construction of the planar graph H.

Given a solution of Planar Disjoint Path in H, for each $i \in [n]$, the selection of a cycle in the SC$_i$-gadget selects a color for v_i, that can be any color that is shared by all color outputs of v_i, and the edge gadgets ensure that two adjacent vertices are in two different color classes. So in this way we obtain a solution of Planar 3-colorability in G. Conversely, given a solution of Planar 3-colorability in G, it defines a color output for $\{a_i, b_i, c_i\}$ for $i \in [n]$. Therefore, we select in the SC$_i$-gadget the path that uses the vertex in $\{a_i, b_i, c_i\}$ corresponding to the color of v_i. In the bifurcate gadget, we choose the paths that use the vertices in $\{a_{i,1}, b_{i,1}, c_{i,1}, a_{i,2}, b_{i,2}, c_{i,2}\}$ leading to two identical color outputs that are exactly the color output of $\{a_i, b_i, c_i\}$. This selection keeps the property that the color output of $\{a_i, b_i, v_i\}$ is contained in the color output of $\{a_{i,1}, b_{i,1}, c_{i,1}\}$ and in the color output of $\{a_{i,2}, b_{i,2}, c_{i,2}\}$, and leaves free as many vertices as possible for other cycles in other gadgets. Inside each edge gadget representing $\{v_i, v_j\} \in E$, we select the paths that are allowed by the free vertices. We complete our path selection by selecting a free path in each expel gadget contained in the path-crossing gadget.

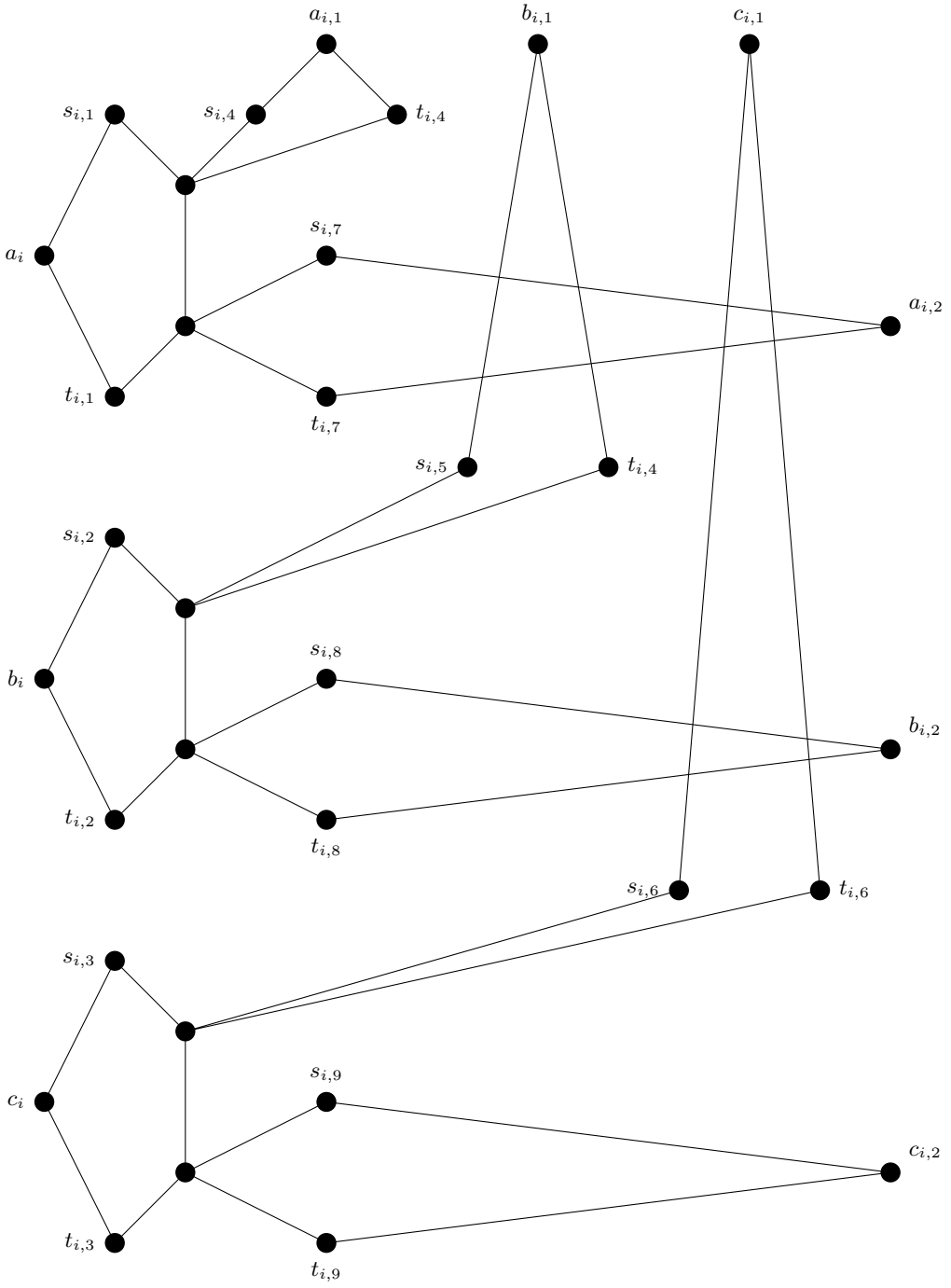


Fig. 21. Bifurcate gadget: To keep planarity, there is a path-crossing gadget in each edge intersection.

As the degree of each vertex in G is bounded by 5, the number of gadgets we introduce for each $v_i \in V(G)$ to construct H is also bounded by a constant, so the total number of vertices of H is linear in the number

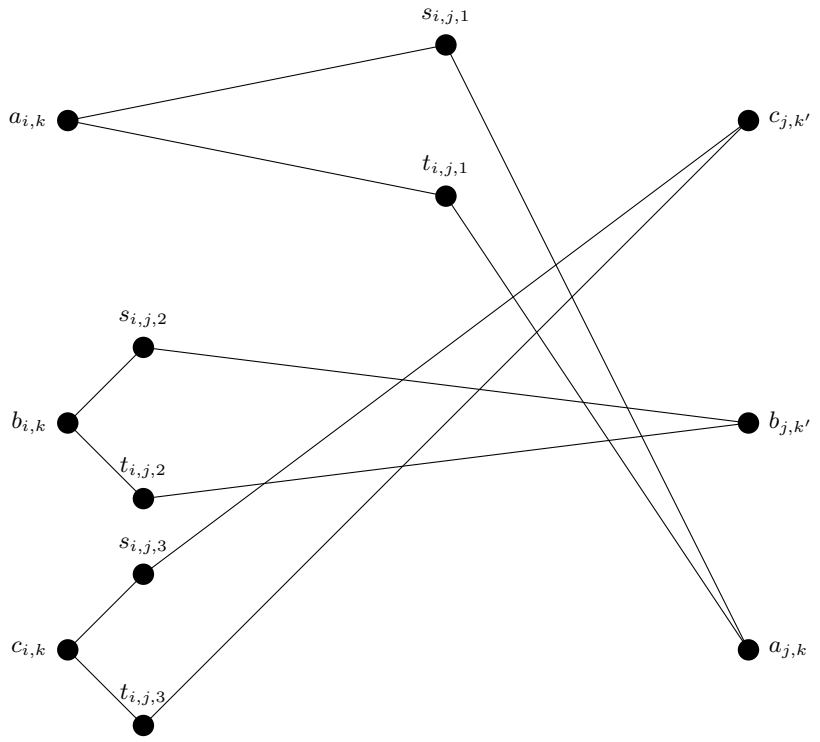


Fig. 22. Edge gadget: To keep planarity, there is a path-crossing gadget in each edge intersection.

of vertices of G. Therefore if we could solve Planar Disjoint Paths in time $2^{o(\sqrt{n})} \cdot n^{O(1)}$, then we could also solve Planar 3-Colorability in time $2^{o(\sqrt{n})} \cdot n^{O(1)}$, which is impossible by Theorem 1 unless the ETH fails. The theorem follows. □

From Theorem 8, we obtain the following corollary.

Corollary 4. *The* Planar Disjoint Paths *problem cannot be solved in time* $2^{o(\mathbf{tw})} \cdot n^{O(1)}$ *unless the ETH fails.*

9 Further research

In this report, we show that planarity plays a role in time complexity for connectivity problems parametrized by treewidth. But it is just the first step and many problems are still open. In particular, we display few interesting problems that we are still not able to solve:

– The Disjoint Path problem is known to be solvable in time $2^{O(\mathbf{tw} \log \mathbf{tw})} \cdot n^{O(1)}$ in general graph [23] and we have not got better algorithm when we restrict the input graph to be planar. The best lower bound we

find for Planar Disjoint Path say that it cannot be solved in time $2^{o(\mathbf{tw})} \cdot n^{O(1)}$ unless ETH fails. We are still not able to define if it is a problem of Type 2 or of Type 3 or even an intermediate type and the question is still open.
- The Subgraph Isomorphism problem is know to be solvable in time $2^{O(h)} \cdot n^{O(1)}$ on planar graphs [6] and graphs on surfaces [2], where h is the number of vertices of a pattern graph H to be found in a host graph G on n vertices. As the Subgraph Isomorphism problem can be expressed in monadic second order logic then an algorithm that compute in time $f(\mathbf{tw}) \cdot n$ exist. To our best knowledge no such algorithm is known, but we prove that we cannot solve the problem in time $2^{o(\mathbf{tw} \log \mathbf{tw})} \cdot n^{O(1)}$.
- Lokshtanov *et al.* [16] have proved that for a number of problems such as Dominating Set or q-Coloring, the best known constant c in algorithms of the form $c^{\mathbf{tw}} \cdot n^{O(1)}$ on general graphs is the best possible unless the Strong ETH fails. Is it possible to provide better constants for these problems on planar graphs? The existence of such algorithms would permit to further refine the problems belonging to Type 1.
- Does a problem that can be solved in time $2^{o(\mathbf{tw})}$ exist?
- Finally, it would be interesting to obtain similar results for problems parameterized by pathwidth, and to extend our algorithms to more general classes of sparse graphs.

References

1. H. L. Bodlaender, M. Cygan, S. Kratsch, and J. Nederlof. Solving weighted and counting variants of connectivity problems parameterized by treewidth deterministically in single exponential time. *CoRR*, abs/1211.1505, 2012.
2. P. Bonsma. Surface split decompositions and subgraph isomorphism in graphs on surfaces. In *Proc. of the 29th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 14 of *LIPIcs*, pages 531–542, 2012.
3. B. Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990.
4. M. Cygan, J. Nederlof, M. Pilipczuk, M. Pilipczuk, J. M. M. van Rooij, and J. O. Wojtaszczyk. Solving Connectivity Problems Parameterized by Treewidth in Single Exponential Time. In *Proc. of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 150–159. IEEE Computer Society, 2011.
5. R. Diestel. *Graph Theory*. Springer-Verlag, 3rd edition, 2005.
6. F. Dorn. Planar subgraph isomorphism revisited. In *Proc. of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 5 of *LIPIcs*, pages 263–274, 2010.
7. F. Dorn, F. V. Fomin, and D. M. Thilikos. Fast Subexponential Algorithm for Non-local Problems on Graphs of Bounded Genus. In *Proc. of the 10th Scandinavian Workshop on Algorithm Theory (SWAT)*, volume 4059 of *LNCS*, pages 172–183, 2006.
8. F. Dorn, F. V. Fomin, and D. M. Thilikos. Catalan Structures and Dynamic Programming in H-minor-free Graphs. In *Proc. of the 19th annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 631–640, 2008.
9. F. Dorn, F. V. Fomin, and D. M. Thilikos. Catalan Structures and Dynamic Programming in H-minor-free graphs. *Journal of Computer and System Sciences*, 78(5):1606–1622, 2012.

10. F. Dorn, E. Penninkx, H. L. Bodlaender, and F. V. Fomin. Efficient Exact Algorithms on Planar Graphs: Exploiting Sphere Cut Decompositions. *Algorithmica*, 58(3):790–810, 2010.
11. J. Flum and M. Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. Springer, 2006.
12. F. V. Fomin and D. M. Thilikos. New upper bounds on the decomposability of planar graphs. *Journal of Graph Theory*, 51(1):53–81, 2006.
13. M. Garey, D. Johnson, and R. E. Tarjan. The Planar Hamiltonian Circuit Problem is NP-Complete. In *SIAM J Comput.*, pages 704–714, 1976.
14. R. Impagliazzo, R. Paturi, and F. Zane. Which Problems Have Strongly Exponential Complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.
15. G. Kreweras. Sur les partitions non croisees d'un cycle. *Discrete Mathematics*, 1(4):333 – 350, 1972.
16. D. Lokshtanov, D. Marx, and S. Saurabh. Known Algorithms on Graphs of Bounded Treewidth are Probably Optimal. In *Proc. of the 22nd annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 777–789, 2011.
17. D. Lokshtanov, D. Marx, and S. Saurabh. Slightly Superexponential Parameterized Problems. In *Proc. of the 22nd annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 760–776, 2011.
18. K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, STOC '87, pages 345–354, New York, NY, USA, 1987. ACM.
19. N. Robertson and P. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309 – 322, 1986.
20. N. Robertson and P. Seymour. Graph minors. x. obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52(2):153 – 190, 1991.
21. J. Rué, I. Sau, and D. M. Thilikos. Dynamic programming for graphs on surfaces. *CoRR*, abs/1104.2486, 2011, to appear in *ACM Transactions on Algorithms (TALG)*. Short version in the *Proc. of ICALP'10*.
22. J. Rué, I. Sau, and D. M. Thilikos. Dynamic Programming for H-minor-free Graphs. In *Proc. of the 18th Annual International Conference on Computing and Combinatorics (COCOON)*, volume 7434 of *LNCS*, pages 86–97, 2012.
23. P. Scheffler. A practical linear time algorithm for disjoint paths in graphs with bounded tree-width. Fachbereich 3 Mathematik, Tech. Report 396/1994, FU Berlin, 1994.