Fast Minor Testing in Planar Graphs *

Isolde Adler[1], Frederic Dorn[2], Fedor V. Fomin[2],
Ignasi Sau[3], and Dimitrios M. Thilikos[4]

[1] Institut für Informatik, Goethe-Universität, Frankfurt, Germany.
iadler@informatik.uni-frankfurt.de
[2] Department of Informatics, University of Bergen, Norway.
{frederic.dorn,fedor.fomin}@ii.uib.no
[3] AlGCo team, CNRS, LIRMM, Montpellier, France
ignasi.sau@lirmm.fr
[4] Department of Mathematics, National and Kapodistrian
University of Athens, Greece.
sedthilk@math.uoa.gr

Abstract. Minor containment is a fundamental problem in Algorithmic Graph Theory used as a subroutine in numerous graph algorithms. A model of a graph H in a graph G is a set of disjoint connected subgraphs of G indexed by the vertices of H, such that if $\{u, v\}$ is an edge of H, then there is an edge of G between components C_u and C_v. Graph H is a minor of G if G contains a model of H as a subgraph. We give an algorithm that, given a planar n-vertex graph G and an h-vertex graph H, either finds in time $\mathcal{O}(2^{\mathcal{O}(h)} \cdot n + n^2 \cdot \log n)$ a model of H in G, or correctly concludes that G does not contain H as a minor. Our algorithm is the first *single-exponential* algorithm for this problem and improves all previous minor testing algorithms in planar graphs. Our technique is based on a novel approach called *partially embedded dynamic programming*.

Keywords: graph minors, planar graphs, branchwidth, parameterized complexity, dynamic programming.

1 Introduction

For two input graphs G and H, the Minor Containment problem is to decide whether H is a minor of G. This is a classical NP-complete problem [18], and remains NP-complete even when both graphs G and H are planar, as it is a generalization of the Hamiltonian Cycle problem. When H is fixed, by the celebrated result of Robertson and Seymour [30], there is an algorithm to decide if H is a minor of an input graph G that runs in time $f(h) \cdot n^3$, where n is the number of vertices of G, h is the number of vertices in H, and f is some recursive function. One of the significant algorithmic implications of this result is that, combined with the Graph Minor Theorem

* An extended abstract of this work appeared in the proceedings of ESA'10 [2].

of Robertson and Seymour [32], it shows the polynomial-time solvability of many graph problems, some of which were previously not even known to be decidable [17]. However, these algorithmic results are highly *non-practical*. This triggered an ongoing quest in the Theory of Algorithms since then –next to on simplifying the 20-papers proof of the Graph Minors Theorem– for making Graph Minors constructive and for making its algorithmic proofs practical for a wide range of applications (e.g., [10, 23]).

Unfortunately, in the minor testing algorithm of Robertson and Seymour [30], the function $f(h)$ has an *immense* exponential growth, which makes the algorithm absolutely impractical even for very simple patterns (see [24] for recent theoretical improvements of this function). There were several attempts to improve the running time of the algorithm of Robertson and Seymour. One direction of such improvements is decreasing the degree of the polynomial in n. For example, Reed and Li gave a linear time algorithm solving K_5-minor containment [29]. The second direction of improvements is towards reducing the exponential dependency in the function $f(h)$, which is a natural direction of study for Parameterized Complexity [16]. A significant step in this direction was done by Hicks [21], who provided in graphs of branchwidth k and m edges an $\mathcal{O}(3^{k^2} \cdot (h+k-1)! \cdot m)$ time algorithm, following the algorithm sketched by Robertson and Seymour [30]. Recently, this was improved to $\mathcal{O}(2^{(2k+1)\log k} \cdot h^{2k} \cdot 2^{2h^2} \cdot m)$ on general graphs, and in planar, and more generally, in graphs of bounded genus, to $2^{\mathcal{O}(k)} \cdot h^{2k} \cdot 2^{\mathcal{O}(h)} \cdot n$ [1].

In this paper we focus on the case where the input graph G is planar.

Planar H-Minor Containment
Input: A planar graph G.
Objective: Either find a model M of H in G, or conclude that G does not contain such a model.

Over the last four decades, many different algorithmic techniques in planar graphs were developed for different type of problems and algorithms, including approximation [5,9], exact [14,26], and parameterized algorithms [3, 8, 15]. However, it seems that none of these approaches can be used to speed up the algorithm for Planar H-Minor Containment.

Our results and key ideas. By arguments inspired by Bidimensionality Theory [7], we first show that the $2^{\mathcal{O}(k)} \cdot h^{2k} \cdot 2^{\mathcal{O}(h)} \cdot n$ time algorithm from [1], combined with the grid minor Theorem of Robertson, Thomas, and Seymour [31], can be used to solve Planar H-Minor Containment in time $\mathcal{O}(2^{\mathcal{O}(h \log h)} \cdot n + n^2 \cdot \log n)$. This directly sets up the challenge of designing a *single-exponential* (on the size h of the pattern H) algorithm for this problem.

Our main result is the following theorem.

Theorem 1. *Given a planar graph G on n vertices and a graph H on h vertices,* Planar H-Minor Containment *is solvable in time* $\mathcal{O}(2^{\mathcal{O}(h)} \cdot n + n^2 \cdot \log n)$.

That is, we prove that when G is planar the behaviour of the function $f(h)$ can be made *single-exponential*, improving over all previous results for this problem [1, 21, 30]. In addition, we can *enumerate* and *count* the number of models within the same time bounds. Let us remark that by Theorem 1, Planar H-Minor Containment is solvable in polynomial time when the size of the pattern graph H is $\mathcal{O}(\log n)$, therefore substantially improving the existing algorithms for small patterns [12].

In order to prove Theorem 1, we introduce a novel approach of dynamic programming in planar graphs of bounded branchwidth, namely *partially embedded dynamic programming*. This approach is extremely helpful in computing graph minors but we believe that this technique can be used in many related problems including Planar Disjoint Paths. Our technique is inspired by the technique of *embedded dynamic programming* introduced in [13] for solving Planar Subgraph Isomorphism for a pattern of size h and an input graph of size n in time $2^{\mathcal{O}(h)} \cdot n$. There, one controls the partial solutions by the ways the separators of G can be routed through the pattern. The difference (and difficulty) concerning Planar H-Minor Containment is that we look for a model M of size $\mathcal{O}(n)$ out of $2^{\mathcal{O}(n)}$ possible non-isomorphic models of H in G. In partially embedded dynamic programming, we look for potential models of H in G with a "magnifying glass" only at a given separator S of G. That is, we consider a collection $\mathcal{A}$ of graphs A arising from 'inflating' a part of H, namely the part interacting with S. Thus, each A behaves like a subgraph of G inside the intersection with S, and outside that inter-

section A behaves like a minor of G; this is why we call our dynamic programming technique "*partially embedded*".

After giving some preliminaries in Section 2, we first show in Section 3 how Planar H-Minor Containment can be solved in polynomial time for input graphs of large branchwidth (in comparison to the pattern size). If the branchwidth is small, we compute the collection $\mathcal{A}$ in Section 3.1 and give the partially embedded dynamic programming approach in Section 3.2.

2 Preliminaries

We use standard graph terminology, see for instance [11].

Graphs and graph minors. All graphs considered in this article are simple and undirected. Given a graph G, we denote by $V(G)$ and $E(G)$ the vertex set and the edge set of G, respectively. A graph H is a *subgraph* of a graph G, $H \subseteq G$, if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. We define graph operation *contracting* edge $e = \{x, y\} \in G$ by removing e, including x and y, and making a new vertex v_e adjacent to the former neighbors of x and y (excluding x and y).

Graph H is a *minor* of graph G (denoted by $H \preceq G$), if H can be obtained from a subgraph of G by a (possibly empty) sequence of edge contractions. In this case we also say that G is a *major* of H. Graph H is a *contraction minor* of graph G (denoted by $H \preceq_c G$), if H can be obtained from G by a (possibly empty) sequence of edge contractions.

A *model* M of minor H in G is a subgraph of G, where the edge set $E(M)$ is partitioned into *c-edges (contraction edges)* and *m-edges (minor edges)* such that the graph resulting from contracting all c-edges is isomorphic to H.

For an illustration, see Fig. 1.

Branchwidth. A *branch decomposition* (T, μ) of a graph G consists of an unrooted ternary tree T (i.e., all internal vertices are of degree three) and a bijection $\mu : L \to E(G)$ 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 $T \setminus \{e\}$. Then let G_i be the graph induced by the edge set $\{\mu(f) : f \in L \cap V(T_i)\}$ for $i \in \{1, 2\}$. The *middle set* is the

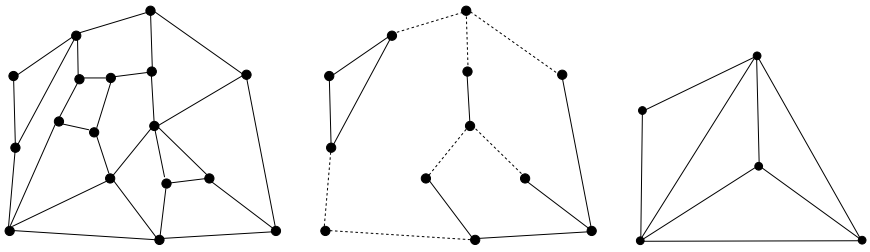

Fig. 1. On the left hand side we illustrate a graph G. In the middle we have a subgraph M of G. Contracting the edges of M indicated by the dashed lines, we obtain minor H illustrated on the right hand side. M is a model of H in G. The dashed lines illustrate its c-edges, the solid lines its m-edges.

intersection of the vertex sets of G_1 and G_2, i.e., $\mathbf{mid}(e) = V(G_1) \cap V(G_2)$. The *width* of (T, μ) is the maximum order of the middle sets over all edges of T, i.e., $\mathit{width}(T, \mu) := \max\{|\mathbf{mid}(e)| \colon e \in E(T)\}$. The *branchwidth* of G is defined as $\mathbf{bw}(G) := \min\{\mathit{width}(T, \mu) \mid (T, \mu)$ branch decomposition of $G\}$. Note that for each $e \in E(T)$, $\mathbf{mid}(e)$ is a separator of G, unless $\mathbf{mid}(e) = \emptyset$.

Remark 1 *For every two edges $e, f \in E(T)$ with $e \cap f \neq \emptyset$, we have $|\mathbf{mid}(e) \cup \mathbf{mid}(f)| \leq 1.5 \cdot \mathit{width}(T, \mu)$.*

Intuitively, a graph G has small branchwidth if G is close to being a tree. The fundamental grid minor Theorem says that, roughly, a graph has either small branchwidth, or it contains a large grid as a minor. We use the variant for planar graphs.

Proposition 1 ([6, 20, 31]) *Given a planar graph G on n vertices with $\mathbf{bw}(G) \geq k$, a model of the $(\lfloor k/3 \rfloor \times \lfloor k/3 \rfloor)$-grid in G can be found in time $\mathcal{O}(n^2 \cdot \log n)$.*

On the other hand, every planar graph is minor of a large enough grid.

Proposition 2 ([31]) *If H is a planar graph with $|V(H)|+2|E(H)| \leq \ell$, then H is isomorphic to a minor of the $(2\ell \times 2\ell)$-grid.*

Planar graphs and equivalent drawings. Let Σ be the unit sphere. A *planar drawing* Φ, or simply *drawing* Φ, of a graph G with vertex set $V(G)$ and edge set $E(G)$ maps vertices to points in the

sphere, and edges to simple curves between their end-vertices, such that edges do not cross, except in common end-vertices. A *plane graph* $\langle G, \Phi \rangle$ is a graph G together with a planar drawing Φ[1]. A *planar graph* is a graph that admits a planar drawing. The set of *faces* $F(G)$ of a plane graph G is defined as the union of the connected regions of $\Sigma \setminus G$. A subgraph of a plane graph G, induced by the vertices and edges incident to a face $f \in F(G)$, is called a *bound* of f (for further reading, see e.g. [11]). Consider any two drawings Φ_1 and Φ_2 of a planar graph G. A *homeomorphism* of Φ_1 onto Φ_2 is a homeomorphism of Σ onto itself which maps vertices, edges, and faces of Φ_1 onto vertices, edges, and faces of Φ_2, respectively. We call two planar drawings *equivalent*, if there is a homeomorphism from one onto the other.

Proposition 3 (e.g. [28]) *Every planar n-vertex graph has $2^{\mathcal{O}(n)}$ non-equivalent drawings. A set of all such drawings can be computed in time $2^{\mathcal{O}(n)}$.*

Proposition 4 ([36]) *The number of non-isomorphic edge-maximal planar graphs on n vertices is $2^{\mathcal{O}(n)}$.*

Nooses and combinatorial nooses. A *noose* of a Σ-plane graph G is a simple closed curve in Σ that meets G only in vertices. From the Jordan Curve Theorem (e.g. [27]), it follows that nooses separate Σ into two regions.

Let $V(N) = N \cap V(G)$ be the vertices and $F(N)$ be the faces intersected by a noose N. The *length* of N is $|V(N)|$, the number of vertices in $V(N)$. The clockwise order in which N meets the vertices of $V(N)$ is a cyclic permutation π on the set $V(N)$.
A *combinatorial noose* $N_C = [v_0, f_0, v_1, f_1, \ldots, f_{\ell-1}, v_\ell]$ in a plane graph G is an alternating sequence of vertices and faces of G, such that

- f_i is a face incident to both v_i, v_{i+1} for all $i < \ell$;
- $v_0 = v_\ell$ and the vertices $v_1, \ldots, v_\ell$ are mutually distinct; and
- if $f_i = f_j$ for any $i \neq j$ and $i, j = 0, \ldots, \ell - 1$, then the vertices v_i, v_{i+1}, v_j, and v_{j+1} do not appear in the order $(v_i, v_j, v_{i+1}, v_{j+1})$ on the bound of face $f_i = f_j$.

The *length* of a combinatorial noose $[v_0, f_0, v_1, f_1, \ldots, f_{\ell-1}, v_\ell]$ is ℓ.

[1] If the drawing Φ is clear from the context, we may also simply write *plane graph* G.

Remark 2 *The order in which a noose N intersects the faces $F(N)$ and the vertices $V(N)$ of a plane graph G gives a unique alternating face-vertex sequence of $F(N) \cup V(N)$ which is a combinatorial noose N_C. Conversely, for every combinatorial noose N_C there exists a noose N with face-vertex sequence N_C.*

We will refer to combinatorial nooses simply as nooses if it is clear from the context.

Proposition 5 ([13]) *Every plane n-vertex graph has $2^{\mathcal{O}(n)}$ combinatorial nooses.*

Sphere cut decompositions. For a plane graph G, we define a *sphere cut decomposition* (or *sc-decomposition*) $\langle T, \mu, \pi \rangle$ as a branch decomposition, which for every edge e of T has a noose N_e that divides Σ into two regions Δ_1 and Δ_2 such that $G_i \subseteq \Delta_i \cup N_e$, where G_i is the graph induced by the edge set $\{\mu(f) : f \in L \cap V(T_i)\}$ for $i \in \{1, 2\}$ and $T_1 \dot\cup T_2 = T \setminus \{e\}$. Thus N_e meets G only in $V(N_e) = \mathbf{mid}(e)$ and its length is $|\mathbf{mid}(e)|$. The vertices of $\mathbf{mid}(e) = V(G_1) \cap V(G_2)$ are ordered according to a cyclic permutation $\pi = (\pi_e)_{e \in E(T)}$ on $\mathbf{mid}(e)$.

Theorem 2 ([15, 35]). *Let G be a planar graph of branchwidth at most k without vertices of degree one embedded on a sphere. Then there exists a sc-decomposition of G of width at most k.*

3 Minor testing in planar graphs

For solving Planar H-Minor Containment in single-exponential time $\mathcal{O}(2^{\mathcal{O}(h)} \cdot n + n^2 \cdot \log n)$, we introduce in this section the method of *partially embedded dynamic programming*. We present Algorithm 3.1 as a roadmap on how we proceed in proving our main Theorem 1.

We divide Algorithm 3.1 into three parts, presented in the following sections.

From the next proposition we can find a model of H in G in the case of G having large branchwidth.

Proposition 6 *Let G and H be planar graphs with $|V(G)| = n$ and $|V(H)| = h$. There exists constant $c \leq 42$ such that if $\mathbf{bw}(G) > c \cdot h$, then G contains a model of H, which can be found in time $\mathcal{O}(n^2 \cdot \log n + h^4)$.*

Algorithm 3.1: The main routine for PLANAR H-MINOR CONTAINMENT.

Input : A planar graph G.
Output : A model M of H in G, if it exists.
Compute sc-decomposition $\langle T, \mu, \pi \rangle$ of G of width $\mathbf{bw}(G)$.
if $\mathbf{bw}(G) > 42 \cdot h$ **then** Compute M (Proposition 6)
else Run PRE-PROC(H) to produce a collection $\mathcal{A}$ of plane majors of H (Section 3.1, Algorithm 3.2);
Run partially embedded dynamic programming on $\langle T, \mu, \pi \rangle$ to find a model M of H in G by using $\mathcal{A}$ (Section 3.2).

Proof: Since any planar graph H with at least three vertices satisfies $|E(H)| \leq 3|V(H)| - 6$, by Proposition 2 any planar graph on h vertices is isomorphic to a minor of the $(14h \times 14h)$-grid.

With the algorithm of [35], we can find $\mathbf{bw}(G)$ in time $\mathcal{O}(n^2)$. If $\mathbf{bw}(G) > 42h$, then by Proposition 1, we can find in time $\mathcal{O}(n^2 \log n)$ a model of a $(14h \times 14h)$-grid in G. From this grid we find a model of H in G using Proposition 2. To conclude, let us discuss how the proof of Proposition 2 provided in [31] can be easily made constructive. For the sake of presentation, we omit many details that can be found in [31]. Indeed, the proof of [31, Fact (1.5)] consists in subdividing H to obtain an auxiliary planar graph H_1 with $|V(H_1)| \leq 7h$, which is isomorphic to a minor of a planar Hamiltonian graph H_2 with $|V(H_2)| \leq 14h$. It is then proved that such a graph H_2 is isomorphic to a minor of a $(|V(H_2)| \times |V(H_2)|)$-grid. Then the proof uses simple operations on vertices, edges, and separating triangles. At one point it uses a Hamiltonian cycle, which existence is guaranteed by Whitney's Theorem [37][2] There exist a constructive version of Whitney's Theorem, i.e. an algorithm finding a Hamiltonian cycle in triangulated planar graph in linear time [34]. One can then check that the overall running time is dominated by the inductive proof of [31, Fact (1.4)], in which one must find $\mathcal{O}(h)$ times a separating triangle in a graph on $7h$ vertices. This procedure can be naïvely done in time $\mathcal{O}(h^4)$. Therefore, a rough upper bound for the algorithm that follows from Proposition 2 is $\mathcal{O}(h^4)$. □

[2] Whitney's Theorem states that each triangulated planar graph without separating triangles is Hamiltonian.

Otherwise, let us assume that $\mathbf{bw}(G) \leq c \cdot h$. In this case, PREPROC($H$) (basically) computes a list of all plane majors A of H up to a fixed size linear in h. This "preprocessing step" is presented in Section 3.1. In the sequel, we are interested in graphs A of our list, if A is a minor of G obtained from H by "uncontracting" some part, such that on a given subset $S \subseteq V(G)$, our graph A looks like a subgraph of G. That is, we consider such A that have a model M_A in G such that every edge of $M_A[S]$ is an m-edge and for every pair of vertices $u, v \in V(M_A[S])$ there is no path of c-edges in M_A connecting u and v. Finally, in Section 3.2, we proceed by partially embedded dynamic programming bottom-up along a sphere cut decomposition of G. Here we make use of the fact that every middle set S yields a separating noose in an embedding of G. If H has a model $M \subseteq G$ that intersects S, then the noose comes from a noose in M, which in turn is present in some major A of H of our list. We use this fact to restrict the number of candidates A we need to consider in every single dynamic programming step.

3.1 Preprocessing

If the branchwidth of G is at most $c \cdot h$, then we compute a sphere cut decomposition of width $\mathcal{O}(h)$ in time $\mathcal{O}(n^2)$ by using the algorithm of [19], and we continue with dynamic programming.

In the first step we do preprocessing. Namely, we compute for H a list of auxiliary graphs A with $H \preceq A$ and $|V(A)| = \mathcal{O}(h)$, such that A is a candidate for a model M in G. We will need this collection in the dynamic programming algorithm described in Section 3.2. To be precise, we compute a collection $\mathcal{A}$ of edge-colored plane graphs, each consisting of

- a planar graph $A_{m,c}$ with $|V(A_{m,c})| \leq h + 1.5 \cdot \mathbf{bw}(G)$, such that H is a contraction minor of $A_{m,c}$;
- a bipartition of the edge set $E(A_{m,c})$ into m-edges and c-edges such that contracting the c-edges of $A_{m,c}$ creates a graph isomorphic to H; and
- a drawing Φ of $A_{m,c}$.

We describe the preprocessing in Algorithm 3.2: the routine PREPROC takes as input H and outputs the collection $\mathcal{A}$ of edge-colored plane graphs $\langle A_{m,c}, \Phi \rangle$.

Algorithm 3.2: The preprocessing algorithm: PRE-PROC.

Input : Planar graph H on h vertices and ℓ edges.
Output : Collection $\mathcal{A}$ of edge-colored plane graphs .
Compute all non-isomorphic planar graphs A satisfying $h \leq |V(A)| \leq 1.5 \cdot \mathbf{bw}(G) + h$ and $|E(A)| \geq \ell$.
for *every such graph* A **do**
  for *every subset* X *of* $E(A)$ *of size* ℓ **do**
    mark the edges in X as m-edges and the edges in $E(A) \setminus X$ as c-edges, resulting in a edge-colored graph $A_{m,c}$.
for *every graph* $A_{m,c}$ **do**
  if *the graph resulting from contracting the c-edges of* $A_{m,c}$ *is isomorphic to* H **then** compute its non-equivalent drawings $\Phi_1, \ldots, \Phi_q$, and add the plane graphs $\langle A_{m,c}, \Phi_i \rangle$ for all $1 \leq i \leq q$ to $\mathcal{A}$.

When doing dynamic programming in Section 3.2, we compute in each dynamic programming step a subset of collection $\mathcal{A}$ consisting of minors of M which represent both H and M.

Lemma 1. *For every planar graph H on h vertices and every constant d, the cardinality of the collection $\mathcal{A}$ of non-isomorphic edge-colored plane graphs on $d \cdot h$ vertices containing a minor isomorphic to H is $2^{\mathcal{O}(h)}$. Furthermore, we can compute $\mathcal{A}$ in time $2^{\mathcal{O}(h)}$.*

Proof: By Proposition 4, the number of non-isomorphic planar graphs A on $d \cdot h$ vertices (for a constant d) is $2^{\mathcal{O}(h)}$. We compute this set in time $2^{\mathcal{O}(h)}$ using the algorithm of [25]. We partition the edge set of each A into three subsets: the edges that we need to delete, the c-edges, and the m-edges. There are again $2^{\mathcal{O}(h)}$ possible such partitions, which can be computed in time $2^{\mathcal{O}(h)}$. We use the linear time algorithm for planar graph isomorphism [22] to check if after applying the graph operations the resulting graph A' is isomorphic to H. If so, we generate all non-equivalent drawings of A and add them to $\mathcal{A}$ by using Proposition 3 and Algorithm 4.1 in [13], again in time $2^{\mathcal{O}(h)}$. □

Using Lemma 1, we get the following corollary.

Corollary 1. *Algorithm 3.2 is correct and runs in time $2^{\mathcal{O}(\mathbf{bw}(G)+h)}$.*

3.2 Partially embedded dynamic programming

From now on, we will refer to an edge-colored plane graph $\langle A_{m,c}, \Phi \rangle \in \mathcal{A}$ simply as A. In this section, we present the technique of partially embedded dynamic programming. Before proceeding to a formal description, we provide the basic intuition behind our algorithm. Towards this, let us consider graphs $A \in \mathcal{A}$ satisfying $H \preceq_c A$ and $A \preceq G$.

We define subgraphs PAST, PRESENT, and FUTURE of A with

- $V(A) = V(\text{PAST}) \cup V(\text{PRESENT}) \cup V(\text{FUTURE})$;
- $E(A) = E(\text{PAST}) \dot\cup E(\text{PRESENT}) \dot\cup E(\text{FUTURE})$;
- PRESENT $\subseteq G$, (i.e., we can obtain A as a minor of G with PRESENT being subgraph of G); and
- $E(\text{PAST}) \subseteq E(H)$, (i.e., we can obtain H as a contraction minor of A without contracting edges in PAST).

Here, we slightly abuse notation by assuming that edge sets in different graphs are actually the same, instead of introducing bijective mappings. Note that we make no assumption about the edges in FUTURE. Intuitively speaking, in partially embedded dynamic programming, we look for potential models M of H with a magnifying glass only in the separators of the sc-decomposition of G. By decontracting H at the separators, we obtain the part PRESENT, which yields a subgraph of G for which we are enabled to apply embedded dynamic programming. For memorizing the rest of the potential model M, we contract all necessary edges to PAST in the processed graph and (almost) all edges to FUTURE in the graph remainder. The picture will be concretized in what follows.

Given a sc-decomposition of G, we proceed with dynamic programming: Every edge e of the sc-decomposition defines a separator $\mathbf{mid}(e) \subseteq V(G)$ and an associated noose N_e, which separates the graph $G_{\text{sub}} \subseteq G$ processed so far from $G \setminus G_{\text{sub}}$. At every edge e of the sc-decomposition, we check for every graph A of $\mathcal{A}$ all the ways in which the graph G_{sub} can be obtained as a major of $A_{\text{sub}} \subseteq A$ with $A_{\text{sub}} = (V(\text{PAST}) \cup V(\text{PRESENT}), E(\text{PAST}) \cup E(\text{PRESENT}))$, where $\mathbf{mid}(e)$ determines $V(\text{PRESENT})$. The noose N_e comes from a noose in A, and this is controlled by the ways in which N_e can be routed through the vertices of A. The number of solutions we get—the *valid partial solutions*—is bounded by the number of combinatorial nooses in A onto which we can map N_e. When updating the valid partial

solutions at two incident edges of the sc-decomposition, we unite PRESENT and PAST of two solutions and set the graph remainder to FUTURE. In a post-processing step, we contract part of PRESENT, namely those edges with at most one endpoint in the newly obtained separator of the sc-decomposition; this part becomes PAST. We then decontract some edges of FUTURE for the next updating step. This concludes the informal description of the algorithm.

In the remaining part of this section, we will precisely describe and analyze the dynamic programming routine with which we achieve the following result:

Lemma 2. *For a plane graph G with a given sc-decomposition $\langle T, \mu, \pi \rangle$ of G of width $\mathbf{bw}(G)$ and a planar graph H on h vertices, we can decide in time $2^{\mathcal{O}(\mathbf{bw}(G)+h)} \cdot n$ whether G contains a model M of H.*

Dynamic programming. We root the sc-decomposition $\langle T, \mu, \pi \rangle$ at some node $r \in V(T)$. For each edge $e \in T$, let L_e be the set of leaves of the subtree rooted at e. The subgraph G_e of G is induced by the edge set $\{\mu(v) \mid v \in L_e\}$. The vertices of $\mathbf{mid}(e)$ form a combinatorial noose N that separates G_e from the residual graph.

Let A be a given plane graph in $\mathcal{A}$. If A is a minor of G, then there exists a plane model M of A in G. Furthermore, for above noose N the intersection $M \cap N$ forms a noose, both in model M and in candidate A. One basic point of partially embedded dynamic programming is to check how the vertices of the combinatorial noose N are mapped to faces and vertices of A. For a combinatorial noose N_A in A, we can map N to N_A bounding (clockwise) a unique subgraph A_{sub} of A.

In each step of the algorithm, we compute the solutions for a subproblem in G_e, where each solution consists of three parts, namely

- a plane edge-colored graph $A \in \mathcal{A}$;
- a combinatorial noose N_A in A; and
- a mapping γ from combinatorial noose N to N_A (defined below).

N_A has the properties that a) it bounds (clockwise) a subgraph $A_{\text{sub}} \subseteq A$ and b) no vertex in $V(A_{\text{sub}}) \setminus V(N_A)$ is incident to a c-edge. The subgraph A_{sub} is representing the part of model M already computed, whereas the residual graph of A represents the part

of M which still has to be verified. For every middle set, we store this information in an array of triples $\langle A, N_A, \gamma \rangle$.

We define now valid mappings between combinatorial nooses and describe how partial solutions are stored in the dynamic programming. Then, we give the different DP-steps and finally verify the approach.

Valid partial solutions. For a middle set $\mathbf{mid}(e)$ of the rooted sc-decomposition $\langle T, \mu, \pi \rangle$ of plane graph G, $N = N_e$ is the associated combinatorial noose in G with face-vertex sequence of $F(N) \cup V(N)$ separating G_e from the residual graph. Let $\mathfrak{N}$ denote the set of all combinatorial nooses of A whose length is at most the length of N and which bound (clockwise) a subgraph $A_{\text{sub}} \subseteq A$ such that no vertex in $V(A_{\text{sub}}) \setminus V(N_A)$ is an end-vertex of a c-edge. We now map N to nooses $N_A \in \mathfrak{N}$, preserving the order. More precisely, we map vertices of N to both vertices and faces of A. Therefore, we consider partitions of $V(N) = V_1(N) \dot\cup V_2(N)$ where vertices in $V_1(N)$ are mapped to vertices of $V(A)$ and vertices in $V_2(N)$ to faces of $F(A)$.

We define a mapping $\gamma : V(N) \cup F(N) \to V(A) \cup F(A)$ relating N to the combinatorial nooses in $\mathfrak{N}$. For every $N_A \in \mathfrak{N}$ on faces and vertices of set $F(N_A) \cup V(N_A)$ and for every partition $V_1(N) \dot\cup V_2(N)$ of $V(N)$, mapping γ *is valid* if

a) γ restricted to $V_1(N)$ is a bijection to $V(N_A)$;
b) every $v \in V_2(N)$ and $f \in F(N)$ satisfy $\gamma(v) \in F(N_A)$ and $\gamma(f) \in F(N_A)$;
c) for every $v_i \in V(N)$ and subsequence $[f_{i-1}, v_i, f_i]$ of N: if $v_i \in V_2(N)$, then face $\gamma(v_i)$ is equal to both $\gamma(f_{i-1})$ and $\gamma(f_i)$, and if $v_i \in V_1(N)$, then vertex $\gamma(v_i)$ is incident to both $\gamma(f_{i-1})$ and $\gamma(f_i)$; and
d) A_{sub} is a minor of G_e with respect to $a) - c)$.

Items *a*) and *b*) say where to map the faces and vertices of N to. Item *c*) (with *a*)) makes sure that if two vertices v_h, v_j in sequence $N = [\ldots, v_h, \underline{\ldots}, v_j, \ldots]$ are mapped to two vertices w_i, w_{i+1} that appear in sequence N_A as $[\ldots, w_i, f_i, w_{i+1}, \ldots]$, then every face and vertex between v_h, v_j in sequence N (here underlined) is mapped to face f_i. Item *d*) simply makes clear that the boundary of G_e is mapped to the boundary A_{sub} such that contracting some part of $G_e \setminus N$ leads to A_{sub}. For an illustration, see Fig. 2.

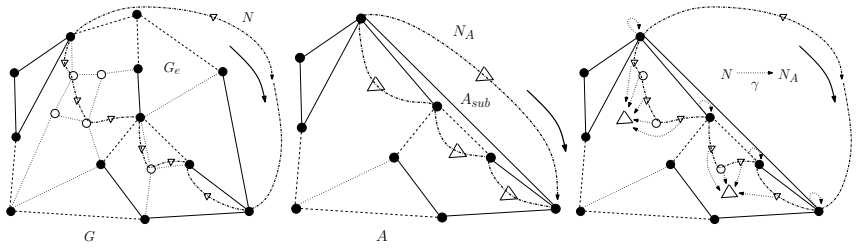

Fig. 2. The figure on the left hand side illustrates the graph G of Fig. 1 with an oriented noose N (dashed) enclosing subgraph G_e. The figure in the middle shows a graph A of collection $\mathcal{A}$ (that is, A is a major of the graph H of Fig. 1) with a noose N_A enclosing A_{sub}. The figure on the right hand side shows a mapping γ from the faces and vertices of N to N_A.

We assign an array Ψ_e to each $\mathbf{mid}(e)$ consisting of triples, where each triple $\langle A, N_A, \gamma\rangle$ represents a minor candidate A together with a valid mapping γ from a combinatorial noose N corresponding to $\mathbf{mid}(e)$ to a combinatorial noose $N_A \in \mathfrak{N}$. The vertices and faces of N are oriented clockwise around the drawing of G_e. Without loss of generality, we assume for every $\langle A, N_A, \gamma\rangle$ the orientation of N_A to be clockwise around the drawing of subgraph A_{sub} of A.

Step 0: Initializing the leaves. For every parent edge e of a leaf v of T, we initialize for every $A \in \mathcal{A}$ the valid mappings from the combinatorial noose bounding the edge $\mu(v)$ of G to every combinatorial noose of length at most two in A (clockwise bounding at most one edge of A).

Step 1a): Update process. We update the arrays of the middle sets bottom-up in post-order manner from the leaves of T to root r. During this updating process it is guaranteed that the *local* solutions for each minor associated with a middle set of the sc-decomposition are combined into a *global* solution for the overall graph G.

In each dynamic programming step, we compare the arrays of two middle sets $\mathbf{mid}(e)$ and $\mathbf{mid}(f)$ in order to create a new array assigned to the middle set $\mathbf{mid}(g)$, where e, f, and g have a vertex of T in common. From [15] we know that the combinatorial noose N_g is formed by the symmetric difference of the combinatorial nooses N_e, N_f and that $G_g = G_e \cup G_f$. In other words, we are ensured that if two solutions on G_e and G_f bounded by N_e and N_f *fit together*, then

they form a new solution on G_g bounded by N_g. We now determine when two solutions represented as tuples in the arrays Ψ_e and Ψ_f fit together. We update two triples $\langle A^1, N_A^1, \gamma_1\rangle \in \Psi_e$ and $\langle A^2, N_A^2, \gamma_2\rangle \in \Psi_f$ to a new triple in Ψ_g if

- $A^1 = A^2 =: A \in \mathcal{A}$ and every edge of A with no endpoint in $V(N_A^1) \cup V(N_A^2)$ is an m-edge;
- for every $x \in (\, V(N_e) \cup F(N_e)\,) \cap (\, V(N_f) \cup F(N_f)\,)$, we have $\gamma_1(x) = \gamma_2(x)$; and
- for the subgraph A^1_{sub} of A separated by N_A^1 and the subgraph A^2_{sub} of A separated by N_A^2, we have that $E(A^1_{\text{sub}}) \cap E(A^2_{\text{sub}}) = \emptyset$ and $V(A^1_{\text{sub}}) \cap V(A^2_{\text{sub}}) \subseteq \{\gamma(v) \mid v \in V(N_e) \cap V(N_f)\}$.

That is, we only update solutions with the same graph A and with the two nooses N_A^1 and N_A^2 bounding (clockwise) two edge-disjoint parts of A and intersecting in a consecutive subsequence of both N_A^1 and N_A^2. If the two solutions on N_e and N_f fit together, we get a valid mapping $\gamma_3 : N_g \to N_A^3$ to a noose N_A^3 of A as follows:

- for every $x \in (\, V(N_e) \cup F(N_e)\,) \cap (\, V(N_f) \cup F(N_f)\,) \cap (\, V(N_g) \cup F(N_g),\,)$ we have $\gamma_1(x) = \gamma_2(x) = \gamma_3(x)$;
- for every $y \in (\, V(N_e) \cup F(N_e)\,) \setminus (\, V(N_f) \cup F(N_f)\,)$ we have $\gamma_1(y) = \gamma_3(y)$; and
- for every $z \in (\, V(N_f) \cup F(N_f)\,) \setminus (\, V(N_e) \cup F(N_e)\,)$ we have $\gamma_2(z) = \gamma_3(z)$.

We have that γ_3 is a valid mapping from N_g to the combinatorial noose N_A^3 that bounds subgraph $A^3_{\text{sub}} = A^1_{\text{sub}} \cup A^2_{\text{sub}}$.

Step 1b): Post-processing. Before adding a triple $\langle A, N_A^3, \gamma_3\rangle$ to array Ψ_g, we need to manipulate A so that *a*) it does not grow too big and *b*) it is suitable for future update operations. In A restricted to subgraph A^3_{sub}, we contract all c-edges with at least one end-vertex not in N_A^3 in order to fulfill *a*). Concerning *b*), for every $B \in \mathcal{A}$ we check for all its nooses N_B with $|V(N_B)| = |V(N_A^3)|$ if there is a bijection β from N_A^3 to N_B such that the following holds. If in a copy of B we contract those c-edges which *i*) are in the subgraph *counter-clockwise* bounded by N_B and *ii*) have at least one end-vertex not in N_B, then we obtain a edge-colored graph isomorphic to A. We define $\delta = \gamma_3 \circ \beta$ and we replace $\langle A, N_A^3, \gamma_3\rangle$ in array Ψ_g by those triples $\langle B, N_B, \delta\rangle$ which validate properties *i*) and *ii*).

Step 2: Termination. If, at some step, we have a solution where the entire minor H is formed, we terminate the algorithm accepting. That is the case, if for some triple we have that $H \preceq A_{\text{sub}} \preceq A$ and A_{sub} is bounded by N_A. We output model M of H in G represented by this A by reconstructing a solution top-down in $\langle T, \mu, \pi \rangle$. If at root r no $A \in \mathcal{A}$ has been computed, we reject.

Correctness of the algorithm. In Corollary 1, we already showed that the preprocessing correctly computes the collection $\mathcal{A}$ of pairwise non-isomorphic edge-colored plane graphs A on at least h and at most $1.5 \cdot \mathbf{bw}(G) + h$ vertices containing H as a minor. In the update process on the nooses N^1, N^2 of two incident edges of the sc-decomposition, we produce graphs with as many vertices since we have for candidate A of H in G that A intersects $V(N^1) \cup V(N^2)$, and by Remark 1 that $|V(N^1) \cup V(N^2)| \leq 1.5 \cdot \mathbf{bw}(G)$ and up to h vertices of A might be outside N^1_A and N^2_A.

We have already seen how to map every combinatorial noose of G that identifies a separation of G via a valid mapping γ to a combinatorial noose of A determining a separation of A. Step 0 ensures that every edge of A is bounded by a combinatorial noose N_A of length two, which is determined by triple $\langle A, N_A, \gamma \rangle$ in an array assigned to a leaf edge of T. We need to show that Step 1a) and 1b) compute a valid solution for N_g from N_e and N_f, given incident edges e, f, g. We note that the property that the symmetric difference of the combinatorial nooses N_e and N_f forms a new combinatorial noose N_g is passed on to the combinatorial nooses N^1_A, N^2_A, and N^3_A of A, too. If the two solutions fit together, then A^1_{sub} of A separated by N^1_A and subgraph A^2_{sub} of A separated by N^2_A only intersect in the image of $V(N_e) \cap V(N_f)$. We observe that N^1_A and N^2_A intersect in a consecutive alternating subsequence with order reversed to each other, i.e., $N^1_A \mid_{N_e \cap N_f} = \overline{N^2_A} \mid_{N_e \cap N_f}$, where $\overline{N_A}$ means the reversed sequence N_A. Since every oriented N_A uniquely identifies a separation of $E(A)$, we can easily decide whether two triples $\langle A, N^1_A, \gamma \rangle \in \Psi_e$ and $\langle A, N^2_A, \gamma \rangle \in \Psi_f$ fit together and bound a new subgraph A^3_{sub} of A. In Step 1b), we contract all c-edges in A^3_{sub} with an end-vertex in $V(A^3_{\text{sub}}) \setminus V(N^3_A)$ such that A^3_{sub} represents a subgraph of H. Next we blow up $\overline{A^3_{\text{sub}}}$, that is the subgraph of A on the other side which N^3_A bounds counter-clockwise. From $\mathcal{A}$ we compute every possible graph partitioned into one subgraph isomorphic to A^3_{sub} and one subgraph

major of $\overline{A^3_{\text{sub}}}$ separated by a noose bijectively mapped from N^3_A. If there exists an $A \in \mathcal{A}$ which is a minor of G, then at some point we will enter Step 2 and produce the entire model M.

Running time of the algorithm. By Corollary 1, computing the collection $\mathcal{A}$ in the preprocessing step can be done in time $2^{\mathcal{O}(\mathbf{bw}(G)+h)}$. We now give an upper bound on the size of each array Ψ. For each $A \in \mathcal{A}$, the number of combinatorial nooses in $\mathfrak{N}$ we are considering is bounded by the total number of combinatorial nooses in A, which is $2^{\mathcal{O}(\mathbf{bw}(G)+h)}$ by Proposition 5. The number of partitions of vertices of any combinatorial noose N is bounded by $2^{|V(N)|}$. Since the order of both N_A and N is given, we only have $2 \cdot |V(A)|$ possibilities to map vertices of N to N_A, once the vertices of N are partitioned. Thus, in an array Ψ_e we may have up to $2^{\mathcal{O}(|V(A)|)} \cdot 2^{|V(N)|} \cdot |V(A)|$ triples $\langle A, N_A, \gamma \rangle$. We first create all triples in the arrays assigned to the leaves. Since middle sets of leaves only consist of an edge in G, we get arrays of size $\mathcal{O}(|V(A)|^2)$, which we compute in the same asymptotic running time. When updating middle sets $\mathbf{mid}(e), \mathbf{mid}(f)$, we compare every triple of array Ψ_e to every triple in array Ψ_f to check if two triples fit together. We can compute the unique subgraph A^1_{sub} (resp. A^2_{sub}) described by a triple in Ψ_e (resp. Ψ_f), compare two triples in Ψ_e and Ψ_f, and create a new triple in time linear in the order of $V(N)$ and $V(H)$. For adding a new triple to Ψ_g in the post-processing, we apply the color coding technique [4] for computing each of the $2^{\mathcal{O}(\mathbf{bw}(G)+h)}$ nooses in $\mathfrak{N}$ in the same asymptotic running time.

This completes the proof of Lemma 2.

Proof of Theorem 1. We put everything together by verifying Algorithm 3.1. We produce in time $\mathcal{O}(n^2 \cdot \log n)$ a sc-decomposition of input graph G [19]. Next, either we can immediately compute a minor model of G in time $\mathcal{O}(n^2 \cdot \log n + h^4)$ (Proposition 6) or we run our 2-step-algorithm: we produce all majors of the minor pattern (Lemma 1) with Algorithm 3.2 in time $2^{\mathcal{O}(h)}$, and run partially embedded dynamic programming in time $2^{\mathcal{O}(h)} \cdot n$ (Lemma 2). □

4 Conclusions and further research

In this paper we showed that Planar H-Minor Containment is solvable in time $\mathcal{O}(2^{\mathcal{O}(h)} \cdot n + n^2 \cdot \log n)$ for a host graph on n vertices and a pattern H on h vertices. That is, we showed that the problem can be solved in *single-exponential* time in h, significantly improving all previously known algorithms. Similar to [13], we can enumerate and count the number of models within the same time bounds.

Let us discuss some interesting avenues for further research concerning minor containment problems. First, it seems possible to solve in single-exponential time other variants of planar minor containment using our approach, like looking for a contraction minor, an induced minor, or a topological minor, as it has been recently done in [1] for general host graphs using completely different techniques. Also, it would be interesting to count the number of non-isomorphic models faster than just by enumerating models and removing isomorphic duplicates.

An important question is if, up to some assumption from complexity theory, the running time of our algorithm is tight. In other words, is there a $2^{o(h)} \cdot n^{\mathcal{O}(1)}$ algorithm (i.e., a *subexponential* algorithm) solving Planar H-Minor Containment, or the existence of such an algorithm would imply the failure of, say, the Exponential Time Hypothesis? A first step could be to study the existence of subexponential algorithms when the pattern is further restricted to be a k-outerplanar graph for some constant k, or any other subclass of planar graphs.

Conversely, single-exponential algorithms may exist for host graphs more general than planar graphs. The natural candidates are host graphs embeddable in an arbitrary surface. One possible approach could be to use the framework recently introduced in [33] for performing dynamic programming for graphs on surfaces. The main ingredient of this framework is a new type of branch decomposition of graphs on surfaces, called *surface cut decomposition*, which plays the role of sphere cut decompositions for planar graphs.

References

[1] I. Adler, F. Dorn, F. V. Fomin, I. Sau, and D. M. Thilikos. Faster Parameterized Algorithms for Minor Containment. In *Proc. of the 12th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, volume 6139 of *LNCS*, pages 322–333, 2010.

[2] I. Adler, F. Dorn, F. V. Fomin, I. Sau, and D. M. Thilikos. Fast Minor Testing in Planar Graphs. In *Proc. of the 18th Annual European Symposium on Algorithms (ESA)*, volume 6346 of *LNCS*, pages 97–109, 2010.

[3] J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, and R. Niedermeier. Fixed parameter algorithms for dominating set and related problems on planar graphs. *Algorithmica*, 33:461–493, 2002.

[4] N. Alon, R. Yuster, and U. Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

[5] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41:153–180, 1994.

[6] H. L. Bodlaender, A. Grigoriev, A. M. C. A. Koster. Treewidth Lower Bounds with Brambles. *Algorithmica*, 51(1): 81-98, 2008.

[7] E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos. Subexponential parameterized algorithms on graphs of bounded genus and H-minor-free graphs. *Journal of the ACM*, 52(6):866–893, 2005.

[8] E. D. Demaine and M. Hajiaghayi. Bidimensionality. In M.-Y. Kao, editor, *Encyclopedia of Algorithms*. Springer, 2008.

[9] E. D. Demaine and M. T. Hajiaghayi. Bidimensionality: new connections between FPT algorithms and PTASs. In *Proc. of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 590–601, 2005.

[10] E. D. Demaine, M. T. Hajiaghayi, and K. i. Kawarabayashi. Algorithmic Graph Minor Theory: Decomposition, Approximation, and Coloring. In *Proc. of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 637–646, 2005.

[11] R. Diestel. *Graph Theory*, volume 173. Springer-Verlag, 2005.

[12] M. Dinneen and L. Xiong. The Feasibility and Use of a Minor Containment Algorithm. Computer Science Technical Reports 171, University of Auckland, 2000.

[13] F. Dorn. Planar Subgraph Isomorphism Revisited. In *Proc. of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 263–274, 2010.

[14] F. Dorn, F. V. Fomin, and D. M. Thilikos. Subexponential parameterized algorithms. *Computer Science Review*, 2(1):29–39, 2008.

[15] 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.

[16] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[17] M. R. Fellows and M. A. Langston. On search, decision and the efficiency of polynomial-time algorithms. *J. Comp. Syst. Sc.*, 49:769–779, 1994.

[18] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[19] Q.-P. Gu and H. Tamaki. Constant-factor approximations of branch-decomposition and largest grid minor of planar graphs in $O(n^{1+\varepsilon})$ time. In *Proc. of the 20th International Symposium Algorithms and Computation (ISAAC)*, volume 5878 of *LNCS*, pages 984–993, 2009.

[20] Q. P. Gu and H. Tamaki. Improved bound on the planar branchwidth with respect to the largest grid minor size. Technical Report SFU-CMPT-TR 2009-17, Simon Fraiser University, 2009.
[21] I. V. Hicks. Branch decompositions and minor containment. *Networks*, 43(1):1–9, 2004.
[22] J. E. Hopcroft and J. K. Wong. Linear time algorithm for isomorphism of planar graphs (preliminary report). In *Proc. of the 6th Annual ACM Symposium on Theory of Computing (STOC)*, pages 172–184, 1974.
[23] K. i. Kawarabayashi and B. A. Reed. Hadwiger's conjecture is decidable. In *Proc. of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 445–454, 2009.
[24] K. i. Kawarabayashi and P. Wollan. A shorter proof of the Graph Minor Algorithm - The Unique Linkage Theorem -. In *Proc. of the 42st Annual ACM Symposium on Theory of Computing (STOC)*, 2010. To appear.
[25] Z. Li and S.-I. Nakano. Efficient generation of plane triangulations without repetitions. In *Proc. of the 28th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2076 of *LNCS*, pages 433–443, 2001.
[26] R. J. Lipton and R. E. Tarjan. Applications of a planar separator theorem. *SIAM J. Comput.*, 9:615–627, 1980.
[27] B. Mohar and C. Thomassen. *Graphs on surfaces*. John Hopkins University Press, 2001.
[28] D. Osthus, H. J. Prömel, and A. Taraz. On random planar graphs, the number of planar graphs and their triangulations. *J. Comb. Theory, Ser. B*, 88(1):119–134, 2003.
[29] B. A. Reed and Z. Li. Optimization and Recognition for K_5-minor Free Graphs in Linear Time. In *Proc. of the 8th Latin American Symposium on Theoretical Informatics (LATIN)*, pages 206–215, 2008.
[30] N. Robertson and P. Seymour. Graph Minors. XIII. The Disjoint Paths Problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995.
[31] N. Robertson, P. Seymour, and R. Thomas. Quickly excluding a planar graph. *J. Comb. Theory, Ser. B*, 62(2):323–348, 1994.
[32] N. Robertson and P. D. Seymour. Graph Minors. XX. Wagner's Conjecture. *J. Comb. Theory, Ser. B*, 92(2):325–357, 2004.
[33] J. Rué, I. Sau, and D. M. Thilikos. Dynamic Programming for Graphs on Surfaces. In *Proc. of the 37th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 6198 of *LNCS*, pages 372–383, 2010.
[34] D. P. Sanders. On Hamilton cycles in certain planar graphs. *J. Graph Theory*, 21(1):43–50, 1998.
[35] P. D. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.
[36] W. T. Tutte. A census of planar triangulations. *Canadian Journal of Mathematics*, 14:21–38, 1962.
[37] H. Whitney. A theorem on graphs. *Annals of Mathematics*, 32:378–390, 1931.