Efficient Exact Algorithms on Planar Graphs: Exploiting Sphere Cut Decompositions

Frederic Dorn

Eelko Penninkx

Hans L. Bodlaender

Fedor V. Fomin

Department of Information and Computing Sciences,
Utrecht University

Technical Report UU-CS-2006-006

www.cs.uu.nl

ISSN: 0924-3275

Efficient Exact Algorithms on Planar Graphs: Exploiting Sphere Cut Decompositions *

Frederic Dorn[1], Eelko Penninkx[2], Hans L. Bodlaender[2], and Fedor V. Fomin[1]

[1] Department of Informatics, University of Bergen, N-5020 Bergen, Norway,
frederic.dorn@ii.uib.no, fedor.fomin@ii.uib.no
[2] Department of Information and Computing Sciences, Utrecht University,
P.O. Box 80.089, 3508 TB Utrecht, the Netherlands,
penninkx@cs.uu.nl, hansb@cs.uu.nl

Abstract. A divide-and-conquer strategy based on variations of the Lipton-Tarjan planar separator theorem has been one of the most common approaches for solving planar graph problems for more than 20 years. We present a new framework for designing fast subexponential exact and parameterized algorithms on planar graphs. Our approach is based on geometric properties of planar branch decompositions obtained by Seymour & Thomas, combined with refined techniques of dynamic programming on planar graphs based on properties of non-crossing partitions. Compared to divide-and-conquer algorithms, the main advantages of our method are a) it is a generic method which allows to attack broad classes of problems; b) the obtained algorithms provide a better worst case analysis. To exemplify our approach we show how to obtain an $O(2^{6.903\sqrt{n}})$ time algorithm solving weighted Hamiltonian Cycle. We observe how our technique can be used to solve Planar Graph TSP in time $O(2^{9.8594\sqrt{n}})$. Our approach can be used to design parameterized algorithms as well. For example we introduce the first $2^{O(\sqrt{k})}n^{O(1)}$ time algorithm for parameterized Planar $k-$cycle by showing that for a given k we can decide if a planar graph on n vertices has a cycle of length at least k in time $O(2^{13.6\sqrt{k}}n + n^3)$.

1 Introduction

The celebrated Lipton-Tarjan planar separator theorem [20] is one of the most common approaches to obtain algorithms with subexponential running time for many problems on planar graphs [21]. The usual running time of such algorithms is $2^{O(\sqrt{n})}$ or $2^{O(\sqrt{n}\log n)}$, however the constants hidden in big-Oh of the exponent are a serious obstacle for practical implementation. During the last few years a lot of work has been done to improve the running time of divide-and-conquer type algorithms [3, 4].

One of the possible alternatives to divide-and-conquer algorithms on planar graphs was suggested by Fomin & Thilikos [13]. The idea of this approach is very simple: compute the treewidth (or branchwidth) of a planar graph and then use the well developed machinery of dynamic programming on graphs of bounded treewidth (or branchwidth) [6]. For example, it can be shown that, when given a branch decomposition of width ℓ of a graph G on n vertices, the maximum independent set of G can be found in time $O(2^{\frac{3\ell}{2}}n)$. The branchwidth of a planar graph G is at most $2.122\sqrt{n}$ and it can be found in time $O(n^3)$ [24] (see also [14]). Putting all together, we obtain an $O(2^{3.182\sqrt{n}})$ time algorithm solving Independent Set on planar graphs. Note that planarity comes into play twice in this approach: First

* This work was partially supported by Norges forskningsråd project 160778/V30, and partially by the Netherlands Organisation for Scientific Research NWO (project *Treewidth and Combinatorial Optimisation*).

in the upper bound on the branchwidth of a graph and second in the polynomial time algorithm for constructing an optimal branch decomposition. A similar approach combined with the results from graph minor theory [22] works for many parameterized problems on planar graphs [9]. Using such an approach to solve, for example, HAMILTONIAN CYCLE we end up with an $2^{O(\sqrt{n}\log n)}$ algorithm on planar graphs, as all known algorithms for this problem on graphs of treewidth ℓ require $2^{O(\ell \log \ell)}n^{O(1)}$ steps [7]. In this paper we show how to get rid of the logarithmic factor in the exponent for a number of problems. The main idea to speed-up algorithms obtained by the branch decomposition approach is to exploit planarity for the third time: for the first time planarity is used in dynamic programming on graphs of bounded branchwidth.

Our results are based on deep results of Seymour & Thomas [24] on geometric properties of planar branch decompositions. Loosely speaking, their results imply that for a graph G embedded on a sphere Σ, some branch decompositions can be seen as decompositions of Σ into discs (or sphere cuts). We are the first describing such geometric properties of so-called sphere cut decompositions. Sphere cut decompositions seem to be an appropriate tool for solving a variety of planar graph problems. We use a refined dynamic programming technique whose running time can be calculated by the number of combinations of non-crossing partitions.

To demonstrate the power of this combinatorial method we apply it to the following problems.

Planar Hamiltonian Cycle. In the PLANAR HAMILTONIAN CYCLE problem one is given a weighted planar graph, and is asked to compute a cycle over all vertices with minimum weight with respect to the edges. Until very recently there was no known $2^{O(\sqrt{n})}$-time algorithm for this problem. Deĭneko et al. [8] obtained the first result of this form: a divide-and-conquer type algorithm of running time $2^{O(\sqrt{n})}$. Their goal was to get rid of the logarithmic factor in the exponent, accepting a large constant hidden in the big-Oh notation. But even with careful analysis, it is difficult to obtain small constants in the exponent of the divide-and-conquer algorithm due to its recursive nature.

In this paper we use sphere cut decompositions to obtain an $O(2^{6.903\sqrt{n}})$ time algorithm for PLANAR HAMILTONIAN CYCLE.

Traveling Salesman Problem. The TRAVELING SALESMAN PROBLEM (TSP) problem is one of the most attractive problems in Computer Science and Operations Research. For several decades, almost every new algorithmic paradigm was tried on TSP including approximation algorithms, linear programming, local search, polyhedral combinatorics, and probabilistic algorithms [19]. One of the first known exact exponential time algorithms is the algorithm of Held and Karp [15] solving TSP on n cites in time $2^{O(n)}$ by making use of dynamic programming. For some special cases like EUCLIDEAN TSP (where the cites are points in the Euclidean plane and the distances between the cites are Euclidean distances), several researchers independently obtained subexponential algorithms of running time $2^{O(\sqrt{n}\cdot logn)}$ by exploiting planar separator structures (see e.g. [16]). Smith & Wormald

[25] succeed to generalize these results to d-dimensional space and the running time of their algorithm is $2^{d^{O(d)}} \cdot 2^{O(dn^{1-1/d}\log n)} + 2^{O(d)}$. Another variant is Planar Graph TSP, which for a given weighted planar graph G is the TSP with distance metric the shortest path metric of G. Arora et al. [5] use non-crossing partitions to achieve faster approximation schemes. In this paper we give the first $2^{O(\sqrt{n})}$ time exact algorithm for solving Planar Graph TSP.

Parameterized Planar *k*-cycle. The last ten years showed a rapid development of a new branch of computational complexity: Parameterized Complexity (see the book of Downey & Fellows [11]). Roughly speaking, a parameterized problem with parameter k is *fixed parameter tractable* if it admits an algorithm with running time $f(k)|I|^{\beta}$. Here f is a function depending only on k, $|I|$ is the length of the non-parameterized part of the input and β is a constant. Typically, f is an exponential function, e.g. $f(k) = 2^{O(k)}$. During the last five years much effort was put in the construction of algorithms with running time $2^{O(\sqrt{k})}n^{O(1)}$ for different problems on planar graphs. The first paper on the subject was by Alber et al. [1] describing an algorithm with running time $O(2^{70\sqrt{k}}n)$ for the Planar Dominating Set problem. Different fixed parameter algorithms for solving problems on planar and related graphs are discussed in [3, 4, 9]. In the Planar k-Cycle problem a parameter k is given and the question is if there exists a cycle of length at least k in a planar graph. There are several ways to obtain algorithms solving different generalizations of Planar k-Cycle in time $2^{O(\sqrt{k}\log k)}n^{O(1)}$, one of the most general results is Eppstein's algorithm [12] solving the Planar Subgraph Isomorphism problem with pattern of size k in time $2^{O(\sqrt{k}\log k)}n$. Using non-crossing partitions, Demaine et al. [10] remove the logarithmic factor for some connected problems on graphs of outerplanarity k.

By making use of sphere cut decompositions we succeed to find an $O(2^{13.6\sqrt{k}}k\,n + n^3)$ time algorithm solving Planar k-Cycle.

This paper is organized as follows: in Section 2 we start with some basic definitions and introduce sphere cut decompositions. The main part of the presentation of our techniques is spent on Section 3 where we solve Planar Hamiltonian Cycle. We extend our techniques in Section 4 to Planar Graph TSP and in Section 5 to several variants of connected problems. Section 6 is devoted to conclusions and open problems.

2 Geometric Branch Decompositions of Σ-plane Graphs

In this section we introduce our main technical tool, sphere cut decompositions, but first we give some definitions.

Let Σ be a sphere $(x, y, z\colon x^2 + y^2 + z^2 = 1)$. By a *$\Sigma$-plane* graph G we mean a planar graph G with the vertex set $V(G)$ and the edge set $E(G)$ drawn (without crossing) in Σ. Throughout the paper we denote by n the number of vertices of G. To simplify notations, we usually do not distinguish between a vertex of the graph and the point of Σ used in the drawing to represent the vertex or between an edge and the open line segment representing it. 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 face at most once. The length of a noose O is $|O \cap V(G)|$, the number of vertices it meets. Every noose O bounds two open discs Δ_1, Δ_2 in Σ, i.e., $\Delta_1 \cap \Delta_2 = \emptyset$ and $\Delta_1 \cup \Delta_2 \cup O = \Sigma$.

Branch Decompositions and Carving Decompositions. A *branch decomposition* $\langle T, \mu \rangle$ of a graph G consists of an unrooted ternary tree T (i.e. all internal vertices 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* $\operatorname{mid}(e) \subseteq V(G)$ 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 intersection of the vertex sets of G_1 and G_2, i.e., $\operatorname{mid}(e) := V(G_1) \cap V(G_2)$. The *width* bw of $\langle T, \mu \rangle$ is the maximum order of the middle sets over all edges of T, i.e., $\operatorname{bw}(\langle T, \mu \rangle) := \max\{|\operatorname{mid}(e)| : e \in T\}$. An optimal branch decomposition of G is defined by the tree T and the bijection μ which together provide the minimum width, the *branchwidth* $\operatorname{bw}(G)$.

A *carving decomposition* $\langle T, \mu \rangle$ is similar to a branch decomposition, only with the difference that μ is the bijection between the leaves of the tree and the *vertex set* of the graph. For an edge e of T, the counterpart of the middle set, called the *cut set* $\operatorname{cut}(e)$, contains the edges of the graph with end vertices in the leaves of both subtrees. The counterpart of branchwidth is carvingwidth.

We will need the following result:

Proposition 1 ([13]). *For any planar graph G, $\operatorname{bw}(G) \leq \sqrt{4.5n} \leq 2.122\sqrt{n}$.*

Sphere cut Decompositions. For a Σ-plane graph G, we define a *sphere cut decomposition* or *sc-decomposition* $\langle T, \mu, \pi \rangle$ 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 $G_i \subseteq \Delta_i \cup O_e$, $1 \leq i \leq 2$. Thus O_e meets G only in $\operatorname{mid}(e)$ and its length is $|\operatorname{mid}(e)|$. A clockwise traversal of O_e in the drawing of G defines the cyclic ordering π of $\operatorname{mid}(e)$. We always assume that the vertices of every middle set $\operatorname{mid}(e) = V(G_1) \cap V(G_2)$ are enumerated according to π.

The following theorem provides us with the main technical tool. It follows almost directly from the results of Seymour & Thomas [24] and Gu & Tamaki [14]. Since this result is not explicitly mentioned in [24], we provide some explanations below.

Theorem 1. *Let G be a connected Σ-plane graph of branchwidth at most ℓ without vertices of degree one. There exists an sc-decomposition of G of width at most ℓ and such a branch decomposition can be constructed in time $O(n^3)$.*

Proof. Let G be a Σ-plane graph of branchwidth at most ℓ and with minimal vertex degree at least two. Then, $I(G)$ is the simple bipartite graph with vertices $V(G) \cup E(G)$, in which $v \in V(G)$ is adjacent to $e \in E(G)$ if and only if v is an endpoint of e in G. The *medial graph* M_G of G has vertex set $E(G)$, and for every vertex $v \in V(G)$ there is a cycle C_v in M_G with the following properties:

- The cycles C_v of M_G are mutually edge-disjoint and have as union M_G;

- For each $v \in V(G)$, let the neighbors $w_1, \ldots, w_t$ of v in $I(G)$ be enumerated according to the cyclic order of the edges $\{v, w_1\}$, $\ldots$, $\{v, w_t\}$ in the drawing of $I(G)$; then C_v has vertex set $\{w_1, \ldots, w_t\}$ and w_{i-1} is adjacent to w_i $(1 \leq i \leq t)$, where w_0 means w_t.

In a *bond* carving decomposition of a graph, every cut set is a bond of the graph, i.e., every cut set is a minimal cut. Seymour and Thomas [24, Theorems (5.1) and (7.2)] show that a Σ-plane graph G without vertices of degree one is of branchwidth at most ℓ if and only if M_G has a bond carving decomposition of width at most 2ℓ. They also show [24, Algorithm (9.1)] how to construct an optimal bond carving decompositions of the medial graph M_G in time $O(n^4)$. A refinement of the algorithm in [14] gives running time $O(n^3)$. A bond carving decomposition $\langle T, \mu \rangle$ of M_G is also a branch decomposition of G (vertices of M_G are the edges of G) and it can be shown (see the proof of (7.2) in [24]) that for every edge e of T if the cut set $\mathrm{cut}(e)$ in M_G is of size at most 2ℓ, then the middle set $\mathrm{mid}(e)$ in G is of size at most ℓ. It is well known that the edge set of a minimal cut forms a cycle in the dual graph. The dual graph of a medial graph M_G is the *radial graph* R_G. In other words, R_G is a bipartite graph with the bipartition $F(G) \cup V(G)$. A vertex $v \in V(G)$ is adjacent in R_G to a vertex $f \in F(G)$ if and only if the vertex v is incident to the face f in the drawing of G. Therefore, a cycle in R_G forms a noose in G.

To summarize, for every edge e of T, $\mathrm{cut}(e)$ is a minimal cut in M_G, thus $\mathrm{cut}(e)$ forms a cycle in R_G (and a noose O_e in G). Every vertex of M_G is in one of the open discs Δ_1 and Δ_2 bounded by O_e. Since O_e meets G only in vertices, we have that $O_e \cap V(G) = \mathrm{mid}(e)$. Thus for every edge e of T and the two subgraphs G_1 and G_2 of G formed by the leaves of the subtrees of $T \setminus \{e\}$, O_e bounds the two open discs Δ_1 and Δ_2 such that $G_i \subseteq \Delta_i \cup O_e$, $1 \leq i \leq 2$.

Finally, with a given bond carving decomposition $\langle T, \mu \rangle$ of the medial graph M_G, it is straightforward to construct a cycle in R_G corresponding to $\mathrm{cut}(e)$, $e \in E(T)$, and afterwards to compute the ordering π of $\mathrm{mid}(e)$ in time linear in ℓ. □

Non-Crossing Partitions. Together with sphere cut decompositions, non-crossing partitions give us the key to our later dynamic programming approach. A *non-crossing partition (ncp)* is a partition $P(n) = \{P_1, \ldots, P_m\}$ of the set $S = \{1, \ldots, n\}$ such that there are no numbers $a < b < c < d$ where $a, c \in P_i$, and $b, d \in P_j$ with $i \neq j$. A partition can be visualized by a circle with n equidistant vertices on its border, where every set of the partition is represented by the convex polygon with its elements as endpoints. A partition is non-crossing if these polygons do not overlap. Non-crossing partitions were introduced by Kreweras [18], who showed that the number of non-crossing partitions over n vertices is equal to the n-th Catalan number:

$$\mathrm{CN}(n) = \frac{1}{n+1}\binom{2n}{n} \sim \frac{4^n}{\sqrt{\pi} n^{\frac{3}{2}}} \approx 4^n \tag{1}$$

Non-Crossing Matchings. A *non-crossing matching (ncm)* is a special case of a ncp, where $|P_i| = 2$ for every element of the partition. A ncm can be visualized by placing n

vertices on a cycle, and connecting matching vertices with arcs at one fixed side of the cycle. A matching is non-crossing if these arcs do not cross. The number of non-crossing matchings over n vertices is given by:

$$\mathrm{M}(n) = \mathrm{CN}(\frac{n}{2}) \sim \frac{2^n}{\sqrt{\pi}(\frac{n}{2})^{\frac{3}{2}}} \approx 2^n \tag{2}$$

3 Planar Hamiltonian Cycle

In this section we show how sc-decompositions in combination with ncm's can be used to design subexponential algorithms. In the Planar Hamiltonian Cycle problem we are given a weighted Σ-plane graph $G = (V, E)$ with weight function $w\colon E(G) \to N$ and we ask for a cycle of minimum weight through all vertices of V. We can formulate the problem in a different way: A labeling $\mathcal{H} : E(G) \to \{0, 1\}$ is *Hamiltonian* if the subgraph $G_{\mathcal{H}}$ of G formed by the edges with label '1' is a spanning cycle. Find a Hamiltonian labeling $\mathcal{H}$ minimizing $\sum_{e \in E(G)} \mathcal{H}(e) \cdot w(e)$. For an edge labeling $\mathcal{H}$ and a vertex $v \in V(G)$ we define the *$\mathcal{H}$-degree* $deg_{\mathcal{H}}(v)$ of v as the sum of labels assigned to the edges incident to v. Although the use of labelings makes the algorithm more complex, it is necessary for the understanding of the approach for Planar Graph TSP we use later. Let $\langle T, \mu, \pi \rangle$ be a sc-decomposition of G of width ℓ. We root T by arbitrarily choosing an edge e, and subdivide it by inserting a new node s. Let e', e'' be the new edges and set $\mathrm{mid}(e') = \mathrm{mid}(e'') = \mathrm{mid}(e)$. Create a new node *root* r, connect it to s and set $\mathrm{mid}(\{r, s\}) = \emptyset$. Each internal node v of T now has one adjacent edge on the path from v to r, called the *parent edge* e_P, and two adjacent edges towards the leaves, called the *left child* e_L and the *right child* e_R. For every edge e of T the subtree towards the leaves is called the *lower part* and the rest the *residual part* with regard to e. We call the subgraph G_e induced by the leaves of the lower part of e the *subgraph rooted at* e. Let e be an edge of T and let O_e be the corresponding noose in Σ. The noose O_e partitions Σ into two discs, one of which, Δ_e, contains G_e.

Given a labeling $\mathcal{P}[e]\colon E(G_e) \to \{0, 1\}$ we define for every vertex v in G_e the *$\mathcal{P}[e]$-degree* $deg_{\mathcal{P}[e]}(v)$ to be the sum of the labels on the edges incident to v. Let $G_{\mathcal{P}[e]}$ be the subgraph induced by the edges with label '1'. We call $\mathcal{P}[e]$ a *partial Hamiltonian labeling* if $G_{\mathcal{P}[e]}$ satisfies the following properties:

- For every vertex $v \in V(G_e) \setminus O_e$, $deg_{\mathcal{P}[e]}(v) = 2$.
- Every connected component of $G_{\mathcal{P}[e]}$ has exactly two vertices in O_e with $deg_{\mathcal{P}[e]}(v) = 1$, all other vertices of $G_{\mathcal{P}[e]}$ have $deg_{\mathcal{P}[e]}(v) = 2$.

Observe that $G_{\mathcal{P}[e]}$ forms a collection of disjoint paths with endpoints in O_e, and note that every partial Hamiltonian labeling of $G_{\{r,s\}}$ forms a Hamiltonian labeling.

Now the geometric properties of sc-decompositions in combination with non-crossing matchings come into play. For a partial Hamiltonian labeling $\mathcal{P}[e]$ let P be a path of $G_{\mathcal{P}[e]}$. As the graph is planar, no paths cross and we can reduce P to an arc in Δ_e with endpoints on the noose O_e. If we do so for all paths, the endpoints of these arcs form a non-crossing matching.

Because O_e bounds the disc Δ_e and the graph $G_{\mathcal{P}[e]}$ is in Δ_e, we are enabled to scan the vertices of $V(P) \cap O_e$ according to the ordering π and mark with '$1_[$' the first and with '$1_]$' the last vertex of P on O_e. Mark the endpoints of all paths of $G_{\mathcal{P}[e]}$ in such a way. Then the obtained sequence with marks '$1_[$' and '$1_]$', decodes the complete information on how the endpoints of $V(G_{\mathcal{P}[e]})$ hit O_e. With the given ordering π the '$1_[$' and '$1_]$' encode a ncm. The other vertices of $V(G_{\mathcal{P}[e]}) \cap O_e$ are the 'inner' vertices and we mark them by '2'. All vertices of O_e that are not in $G_{\mathcal{P}[e]}$ are marked by '0'.

For an edge e of T and the corresponding noose O_e, the state of dynamic programming is specified by an ordered ℓ-tuple $\boldsymbol{t}_e := (v_1, \ldots, v_\ell)$. Here, the variables $v_1, \ldots, v_\ell$ correspond to the vertices of $O_e \cap V(G)$ taken according to the cyclic order π with an arbitrary first vertex. This order is necessary for a well-defined encoding where the variables v_i take one of the four values: 0, $1_[$, $1_]$, 2. Hence, there are at most $O(4^\ell |V(G)|)$ states. For every state, we compute a value $W_e(v_1, \ldots, v_\ell)$ that is the minimum weight over all partial Hamiltonian labelings $\mathcal{P}[e]$ encoded by $v_1, \ldots, v_\ell$. If no such labeling exists we have $W_e(v_1, \ldots, v_\ell) = +\infty$. For an illustration of a partial Hamiltonian labeling see Figure 1.

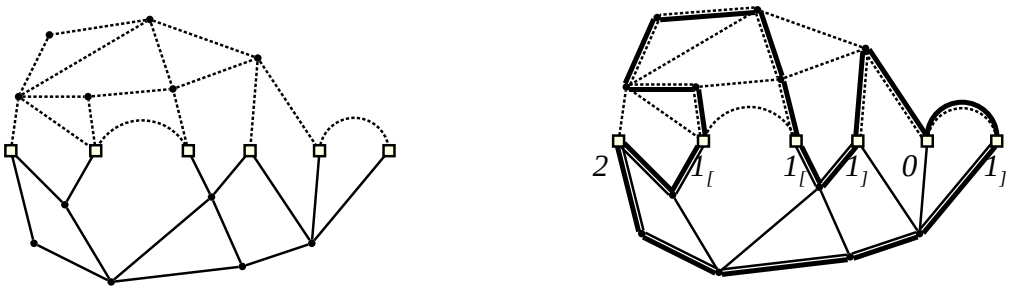

Fig. 1. On the left we see a graph G partitioned by the rectangle vertices of $O_e \cap V(G)$ into G_e in drawn-through edges and $\overline{G_e}$ in dashed edges. On the right subgraph $G_{\mathcal{H}}$ marks a Hamiltonian cycle. $G_{\mathcal{H}}$ is partitioned by the vertices of $O_e \cap V(G)$ which are labeled corresponding to two vertex-disjoint paths in G_e induced by the partial Hamiltonian labeling $\mathcal{P}[e]$.

To compute an optimal Hamiltonian labeling we perform dynamic programming over middle sets $\mathrm{mid}(e) = O(e) \cap V(G)$, starting at the leaves of T and working bottom-up towards the root edge. The first step in processing the middle sets is to initialize the leaves with values $W_e(0, 0) = 0$, $W_e(1_[, 1_]) = w(f)$, where f represents the edge of the graph corresponding to the leaf. Then, bottom-up, update every pair of states of two child edges e_L and e_R to a state of the parent edge e_P assigning a finite value W_P if the state corresponds to a feasible partial Hamiltonian labeling.

Let O_L, O_R, and O_P be the nooses corresponding to edges e_L, e_R and e_P, and let Δ_L, Δ_R and Δ_P be the discs bounded by these nooses. Due to the definition of branch

decompositions, every vertex must appear in at least two of the three middle sets. We partition the set $(O_L \cup O_R \cup O_P) \cap V(G)$ into three sets:

- *Portal vertices* $P := O_L \cap O_R \cap O_P \cap V(G)$,
- *Intersection vertices* $I := O_L \cap O_R \cap V(G) \setminus P$,
- *Symmetric difference vertices* $D := O_P \cap V(G) \setminus (P \cup I)$.

See Figure 2 for an illustration of these notions. Observe that $|P| \leq 2$, as the disc Δ_P contains the union of the discs Δ_L and Δ_R. This observation will prove to be crucial in the analysis of the algorithm.

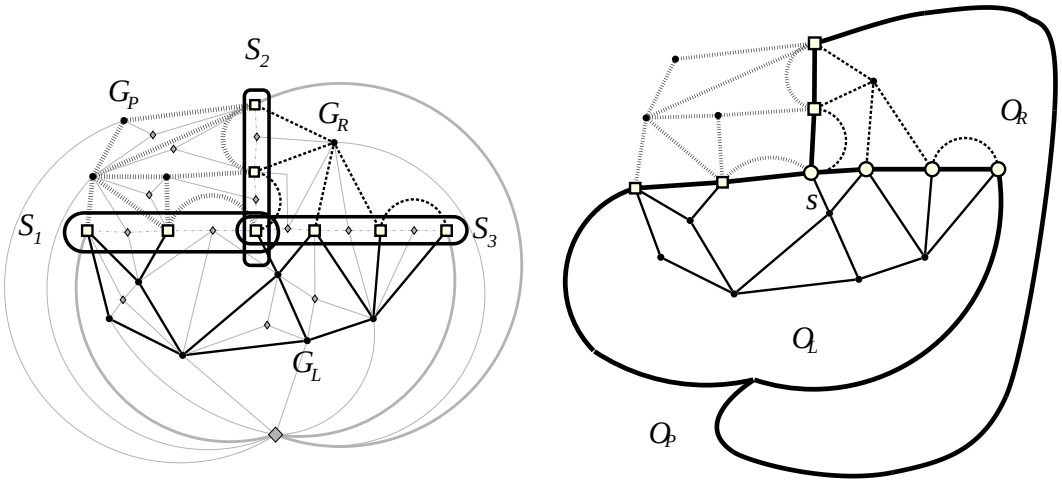

Fig. 2. On the left we see the same graph G as in the last figure. The grey rhombus and grey edges illustrate the radial graph R_G. G is partitioned by the rectangle vertices of S_1, S_2, S_3 into G_L in drawn-through edges, G_R in dashed edges, and G_P in pointed edges. On the right the three nooses O_L, O_R and O_P are marked. Note that the nooses are induced by S_1, S_2, S_3 and the highlighted grey edges on the left hand. All three nooses here intersect in one portal vertex s.

We compute all valid assignments to the variables $\boldsymbol{t}_P = (v_1, v_2, \ldots, v_p)$ corresponding to the vertices $\mathrm{mid}(e_P)$ from all possible valid assignments to the variables of $\boldsymbol{t}_L$ and $\boldsymbol{t}_R$. For a symbol $x \in \{0, 1_[, 1_], 2\}$, we denote by $|x|$ its 'numerical' part, e.g. $|1_[| = 1$. We say that an assignment c_P is *formed* by assignments c_L and c_R if for every vertex $v \in (O_L \cup O_R \cup O_P) \cap V(G)$:

- $\underline{v \in D}$: $c_P(v) = c_L(v)$ if $v \in O_L \cap V(G)$, and $c_P(v) = c_R(v)$ otherwise.
- $\underline{v \in I}$: $|c_L(v)| + |c_R(v)| = 2$.
- $\underline{v \in P}$: $|c_P(v)| = |c_L(v)| + |c_R(v)| \leq 2$.

We compute all ℓ-tuples for $\mathrm{mid}(e_P)$ that can be formed by tuples corresponding to $\mathrm{mid}(e_L)$ and $\mathrm{mid}(e_R)$ and check if the obtained assignment corresponds to a labeling without cycles. For every encoding of $\boldsymbol{t}_P$, we set $W_P = min\{W_P, W_L + W_R\}$.

For the root edge $\{r, s\}$ and its children e' and e'' note that $(O_{e'} \cup O_{e''}) \cap V(G) = I$ and $O_{\{r,s\}} = \emptyset$. Hence, for every $v \in V(G_{\mathcal{P}[\{r,s\}]})$ it must hold that $deg_{\mathcal{P}[\{r,s\}]}(v)$ is two, and that the labelings form a cycle. The optimal Hamiltonian labeling of G results from $min_{\boldsymbol{t}_{\{r,s\}}}\{W_r\}$.

Analyzing the algorithm, we obtain the following lemma.

Lemma 1. Planar Hamiltonian Cycle *on a graph G with branchwidth ℓ can be solved in time $O(2^{3.292\ell}\ell n + n^3)$.*

Proof. By Theorem 1, an sc-decomposition $\langle T, \mu, \pi\rangle$ of width at most ℓ of G can be found in $O(n^3)$.

For a worst-case scenario, assume we have three adjacent edges e_P, e_L, and e_R of T with $|O_L| = |O_R| = |O_P| = \ell$. Without loss of generality we limit our analysis to even values for ℓ, and assume there are no portal vertices. This can only occur if $|I| = |D \cap O_L| = |D \cap O_R| = \frac{\ell}{2}$.

By just checking every combination of ℓ-tuples from O_L and O_R we obtain a bound of $O(\ell 4^{2\ell})$ for our algorithm.

Some further improvement is apparent, as for the vertices $u \in I$ we want the sum of the $\{0, 1_[, 1_], 2\}$ assignments from both sides to be 2, i.e., $|c_L(u)| + |c_R(u)| = 2$.

We start by giving an expression for $Q(\ell, m)$: the number of ℓ-tuples over ℓ vertices where the $\{1_[, 1_]\}$ assignments for m vertices from I is fixed. The only freedom is thus in the $\ell/2$ vertices in $D \cap O_L$ and $D \cap O_R$, respectively:

$$\mathrm{Q}(\ell, m) = \sum_{i=0}^{\frac{\ell}{2}} \binom{\frac{\ell}{2}}{i} 2^{\frac{\ell}{2}-i} \mathrm{M}(i+m) \tag{3}$$

This expression is a summation over the number of $1_[$'s and $1_]$'s in $D \cap O_L$ and $D \cap O_R$, respectively. The term $\binom{\frac{\ell}{2}}{i}$ counts the possible locations for the $1_[$'s and $1_]$'s, the $2^{\frac{\ell}{2}-i}$ counts the assignment of $\{0, 2\}$ to the remaining $\ell/2 - i$ vertices, and the $\mathrm{M}(i+m)$ term counts the ncm's over the $1_[$'s and $1_]$'s. As we are interested in exponential behaviour for large values of ℓ we ignore if $i + m$ is odd, and use that $M(n) \approx 2^n$:

$$Q(\ell, m) = O(\sum_{i=0}^{\frac{\ell}{2}} \binom{\frac{\ell}{2}}{i} 2^{\frac{\ell}{2}-i} 2^{i+m}) = O(2^{\ell+m}) \tag{4}$$

We define $C(\ell)$ as the number of possibilities of forming an ℓ-tuple from O_P. We sum over i: the number of $1_[$'s and $1_]$'s in the assignment for I:

$$C(\ell) = \sum_{i=0}^{\frac{\ell}{2}} \binom{\frac{\ell}{2}}{i} 2^{\frac{\ell}{2}-i} Q(\ell, i)^2 = O(\sum_{i=0}^{\frac{\ell}{2}} \binom{\frac{\ell}{2}}{i} 2^{\frac{\ell}{2}-i} 2^{2\ell} 2^{2i}) \tag{5}$$

The term $2^{\frac{\ell}{2}-i}$ counts the number of ways how the vertices of I are assigned on one side with 0 and on the other side with 2. The term 2^{2i} counts for I the number of ways vertices

are assigned on both side by symbols with numerical value one. Straightforward calculation yields:

$$C(\ell) = O(2^{\frac{5\ell}{2}} \sum_{i=0}^{\frac{\ell}{2}} \binom{\frac{\ell}{2}}{i} 2^i) = O(2^{\frac{5\ell}{2}} 3^{\frac{\ell}{2}}) = O((4\sqrt{6})^\ell) \tag{6}$$

Since we can check in time linear in ℓ if an assignment forms no cycle and the number of edges in the tree of a branch decomposition is $O(n)$, we obtain an overall running time of $O((4\sqrt{6})^\ell \ell n + n^3) = O(2^{3.292\ell} \ell n + n^3)$.

□

By Proposition 1 and Lemma 1 we achieve the running time $O(2^{6.987\sqrt{n}} n^{3/2} + n^3)$ for Planar Hamiltonian Cycle.

Forbidding Cycles We can further improve upon the previous bound by only forming encodings that do not create a partial cycle. As cycles can only be formed at the vertices in I with numerical part 1 in both O_L and O_R, we only consider these vertices.

We would like to have an upper bound for the number of combinations from O_L and O_R that do not induce a cycle. This bound could then be applied to the previous analysis.

Let I have n vertices and be assigned by an ordered n-tuple of variables $(v_1, \ldots, v_n)$. Each variable v_i is a two-tuple $(c_L(v_i), c_R(v_i))$ of assignments $c_L, c_R \in \{1_[, 1_]\}$ of vertex v_i such that $|c_L(v_i)| + |c_R(v_i)| = 2$. For example, suppose I has only two vertices x and y. A cycle is formed if $c_L(x) = c_R(x) = 1_[$ and $c_L(y) = c_R(y) = 1_]$. That is, $((1_[, 1_[), (1_], 1_]))$ encodes a cycle.

Let $B(n)$ denote the set of all n-tuples over the first n vertices of I that form no cycles: $B(0) = \emptyset$, $B(1) = \{((1_[, 1_[))\}$, $B(2) = \{((1_[, 1_[), (1_[, 1_[)), ((1_[, 1_[), (1_], 1_[)), ((1_[, 1_[), (1_[, 1_]))\}$,etc. Exact counting of $B(n)$ for all vertices of I is complex, so we use a different approach. We have a natural upper bound $|B(n)| \leq z^n$ with $z = 4$ when we consider all possible n-tuples.

We divide each $B(i)$ into two classes: $C_1(i)$ contains all i-tuples of the form $(\ldots, (1_[, 1_[))$, and $C_2(i)$ contains all other i-tuples. We add every possible two-tuple to $C_1(i)$ and $C_2(i)$ to obtain two new classes $C_1(i+1)$ and $C_2(i+1)$ of $B(i+1)$. Adding two-tuple $(1_], 1_])$ to items from $C_1(i)$ is forbidden, as this directly gives us a cycle. Addition of $(1_[, 1_[)$ to i-tuples of both, $C_1(i)$ and $C_2(i)$ gives us $i+1$-tuples of class $C_1(i+1)$. Addition of $(1_[, 1_])$ or $(1_], 1_[)$ to either class leads to $i+1$-tuples of class $C_2(i+1)$, or might lead to infeasible encodings. Given these classes we create a 2×2 transition matrix A for the transposed vectors of the class cardinalities $(|C_1(i)|, |C_2(i)|)^T$ and $(|C_1(i+1)|, |C_2(i+1)|)^T$ such that $(|C_1(i+1)|, |C_2(i+1)|)^T \leq A\,(|C_1(i)|, |C_2(i)|)^T$. For large n we have that $(|C_1(n)|, |C_2(n)|)^T \leq A^n\,(|C_1(1)|, |C_2(1)|)^T \approx z^n x_1$ where z is largest real eigenvalue of A and x_1 is an eigenvector. Thus, z^n is a bound of $|B(n)|$. It follows that $A = \begin{pmatrix} 1 & 1 \\ 2 & 3 \end{pmatrix}$. As the largest real eigenvalue of A is $2 + \sqrt{3}$, we have $z \leq 3.73205$ and bound $|B(n)| \leq 3.73205^n$.

Using these two classes eliminates all cycles over two consecutive vertices. By using three classes we can also prevent larger cycles and obtain tighter bounds for z:

- $C_1(i)$ contains all i-tuples $(\ldots,(1_[,1_]),x)$, where x can consist of zero or more elements $(1_[,1_])$, $(1_],1_[)$ or $(1_],1_[),(1_[,1_])$ after each other.
- $C_2(i)$ contains all i-tuples $(\ldots,(1_[,1_]),x,y)$ where y represents $(1_[,1_])$ or $(1_],1_[)$.
- $C_3(i)$ contains all other i-tuples.

Because we use three classes here, we can also prevents some cycles over more than two consecutive vertices. We obtain a 3×3 transition matrix A such that $(|C_1(i+1)|,|C_2(i+1)|,|C_3(i+1)|)^T \leq A\,(|C_1(i)|,|C_2(i)|,|C_3(i)|)^T$ of the form:

$$A=\begin{pmatrix}1&2&1\\2&0&0\\0&2&3\end{pmatrix}$$

By calculating the largest real eigenvalue we obtain $z\leq 3.68133$. This bound is definitely not tight, it is possible to generalize this technique. We may take more classes into consideration, but already concerning two classes improves our results only incrementally. Computational research suggests that z is probably larger than 3.5. We replace 2^{2i} in Equation (5) by the last calculated value z^i to approximate the number of Planar Hamiltonian Cycles:

$$C(\ell)=O(\sum_{i=0}^{\frac{\ell}{2}}\binom{\frac{\ell}{2}}{i}2^{\frac{\ell}{2}-i}4^{\ell}z^i)=O(2^{3.253\ell}) \tag{7}$$

Thus, we get the following result:

Theorem 2. Planar Hamiltonian Cycle *is solvable in* $O(2^{6.903\sqrt{n}}n^{3/2}+n^3)=O(2^{6.903\sqrt{n}})$.

4 Planar Graph TSP

In the Planar Graph TSP we are given a weighted Σ-plane graph $G=(V,E)$ with weight function $w\colon E(G)\to N$ and we are asked for a shortest closed walk that visits all vertices of G at least once. Equivalently, this is TSP with distance metric the shortest path metric of G. We only sketch the algorithm for Planar Graph TSP since it is very similar to the algorithm for Planar Hamiltonian Cycle. Instead of collections of disjoint paths we now deal with connected components with even vertex degree for the vertices outside the nooses of the sc-decomposition.

It is easy to show that a shortest closed walk passes through each edge at most twice. Thus every shortest closed walk in G corresponds to the minimum Eulerian subgraph in the graph G' obtained from G by adding to each edge a parallel edge. Every vertex of an Eulerian graph is of even degree, which brings us to another equivalent formulation of the problem. A labeling $\mathcal{E}: E(G)\to\{0,1,2\}$ is *Eulerian* if the subgraph $G_{\mathcal{E}}$ of G formed by the edges with positive labels is a connected spanning subgraph and for every vertex $v\in V$ the sum of labels assigned to edges incident to v is even. Thus Planar Graph TSP is

equivalent to finding an Eulerian labeling $\mathcal{E}$ minimizing $\sum_{e\in E(G)} \mathcal{E}(e)\cdot w(e)$. For a labeling $\mathcal{E}$ and vertex $v \in V(G)$ we define the *$\mathcal{E}$-degree* $deg_{\mathcal{E}}(v)$ of v as the sum of labels assigned to the edges incident to v.

Let G be a Σ-plane graph and let $\langle T, \mu, \pi\rangle$ be a rooted sc-decomposition of G of width ℓ. We use the same definitions for O_e, G_e, and Δ_e. We call a labeling $\mathcal{P}[e]\colon E(G_e) \to \{0,1,2\}$ a *partial Eulerian labeling* if the subgraph $G_{\mathcal{P}[e]}$ induced by the edges with positive labels satisfies the following properties:

- Every connected component of $G_{\mathcal{P}[e]}$ has a vertex in O_e.
- For every vertex $v \in V(G_e) \setminus O_e$, the $\mathcal{P}[e]$-degree $deg_{\mathcal{P}[e]}(v)$ of v is even and positive.

The weight of a partial Eulerian labeling $\mathcal{P}[e]$ is $\sum_{f\in E(G_e)} \mathcal{P}[e](f)\cdot w(f)$. Note that every partial Eulerian labeling of $G_{\{r,s\}}$ is also a Eulerian labeling.

Again we encode the information on which vertices of the connected components of $G_{\mathcal{P}[e]}$ of all possible partial Eulerian labelings $\mathcal{P}[e]$ hit $O_e \cap V(G)$. Also for every vertex $v \in O_e \cap V(G)$ the information if $deg_{\mathcal{P}[e]}(v)$ is either 0, or odd, or even and positive.

For a partial Eulerian labeling $\mathcal{P}[e]$ let C be a component of $G_{\mathcal{P}[e]}$ with at least two vertices in noose O_e. We scan the vertices of $V(C) \cap O_e$ according to the ordering π and mark with index '[' the first and with ']' the last vertex of C on O_e. We also mark by '$\square$' the other 'inner' vertices of $V(C) \cap O_e$. Finally we assign a numerical value.

If C has only one vertex in O_e, we mark this vertex by '0'. This includes the case $|V(C)| = 1$. Note that the connected components of $G_{\mathcal{P}[e]}$ form a non-crossing partition ncp. Thus, we can again decode the complete information on which vertices of each connected component of $V(G_{\mathcal{P}[e]})$ hit O_e.

Thus every such state must be an *algebraic term* with the indices '[' being the opening and ']' the closing bracket (with '$\square$' and '0' representing a possible term inside the brackets).

When encoding the parity of the vertex degrees, the following observation is useful: In every graph the number of vertices with odd degree is even. Consider a component C of $G_{\mathcal{P}[e]}$. There is an even number of vertices in $C \cap O_e$ with odd $\mathcal{P}[e]$-degree. Thus, we do not encode the parity of the degree of a vertex assigned by ']'. The parity is determined by the other vertices of the same component. The state of dynamic programming is $\boldsymbol{t}_e := (v_1, \ldots, v_\ell)$ with variables $v_1, \ldots, v_\ell$ having one of the six values: 0, $1_[$, $1_\square$, $2_[$, $2_\square$,]. Hence, there are at most $O(6^\ell |V(G)|)$ states. For every state, we compute a value $W_e(v_1, \ldots, v_\ell)$ that is the minimum weight over all partial Eulerian labelings $\mathcal{P}[e]$ encoded by $(v_1, \ldots, v_\ell)$:

- For every connected component C of $G_{\mathcal{P}[e]}$ with $|C \cap O_e| \geq 2$ the first vertex of $C \cap O_e$ in π is represented by $1_[$ or $2_[$ and the last vertex is represented by]. All other vertices of $C \cap O_e$ are represented by $1_\square$ or $2_\square$. For every vertex v marked by by $1_[$ or $1_\square$ the parity of $deg_{\mathcal{P}[e]}(v)$ is even and for every vertex v marked by by $2_[$ or $2_\square$, $deg_{\mathcal{P}[e]}(v)$ is positive and even.
- For every connected component C of $G_{\mathcal{P}[e]}$ with $v = C \cap O_e$, v is represented by 0. (Note that since for every $w \in V(G_e) \setminus O_e$ it holds that $deg_{\mathcal{P}[e]}(w)$ is even so must $deg_{\mathcal{P}[e]}(v)$.)
- Every vertex $v \in (V(G_e) \cap O_e) \setminus G_{\mathcal{P}[e]}$ is marked by 0. (Note that the vertices of the last two items can be treated in the same way in the dynamic programming.)

We put $W_e = +\infty$ if no such labeling exists. For an illustration of a partial Eulerian labeling see Figure 4. To compute an optimal Eulerian labeling we perform dynamic programming

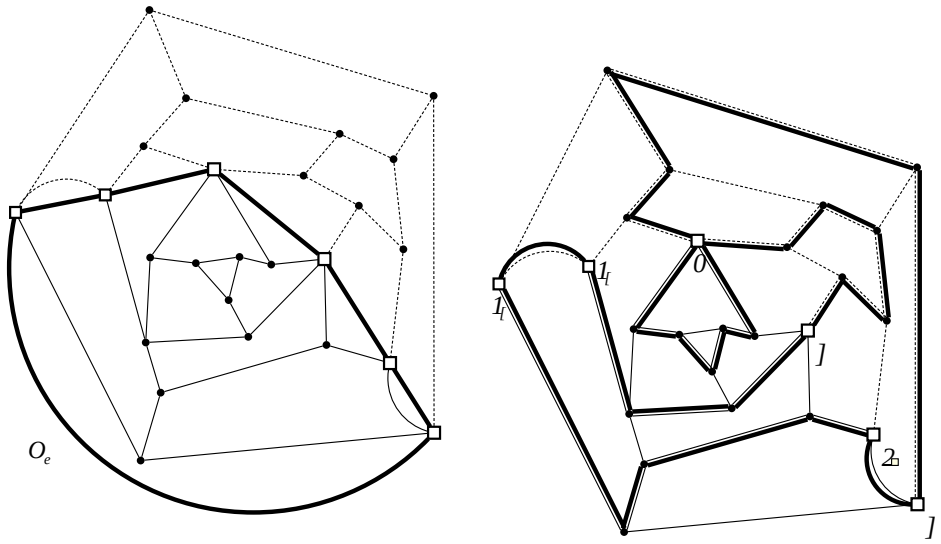

Fig. 4. On the left we see a plane graph G— 3-connected and non-Hamiltonian—partitioned by the rectangle vertices hit by the marked noose O_e into G_e in drawn-through edges and $\overline{G_e}$ in dashed edges. To the right a subgraph $G_{\mathcal{E}}$ with Eulerian labeling $\mathcal{E}$ is marked. $G_{\mathcal{E}}$ is partitioned by the vertices of $O_e \cap V(G)$ which are labeled corresponding to partial Eulerian labeling $\mathcal{P}[e]$ of G_e. Encoding the vertices touched by O_e from the left to the right with $1_[, 1_[, 0,], 2_\square,]$, $G_{\mathcal{P}[e]}$ consists of three components C_1, C_2 and C_3 with $C_1 \cap O_e = \{1_[, 2_\square,]\}, C_2 \cap O_e = \{0\}, C_3 \cap O_e = \{1_[,]\}$. Here $G_{\mathcal{P}[e]}$ has edges only labeled with 1.

over middle sets as in the previous section. The first step of processing the middle sets is to initialize the leaves corresponding to edges $f \in E$ of the graph G with values $W_e(0,0) = 0$, $W_e(1_[,]) = w(f)$, and $W_e(2_[,]) = 2w(f)$. Then, bottom-up, update every pair of states of two child edges e_L and e_R to a state of the parent edge e_P assigning a finite value W_P if the state corresponds to a feasible partial Eulerian labeling.

We compute all valid assignments to the variables $\boldsymbol{t}_P = (v_1, v_2, \ldots, v_p)$ from all possible valid assignments to the variables of $\boldsymbol{t}_L$ and $\boldsymbol{t}_R$. We define the numerical value $|\cdot|$ of ']' to be one if the sum of $deg_{\mathcal{P}[e]}$ over all vertices in the same component is odd, and to be two if the sum is even.

For every vertex $v \in (O_L \cup O_R \cup O_P) \cap V(G)$ we consider the three cases:

- $\underline{v \in D}$: $c_P(v) = c_L(v)$ if $v \in O_L \cap V(G)$, or $c_P(v) = c_R(v)$ otherwise.
- $\underline{v \in I}$: $(|c_L(v)| + |c_R(v)|)\, mod\, 2 = 0$ and $|c_L(v)| + |c_R(v)| > 0$.
- $\underline{v \in P}$: $|c_P(v)| = 0$ if $|c_L(v)| + |c_R(v)| = 0$, $|c_P(v)| = 1$ if $(|c_L(v)| + |c_R(v)|)\, mod\, 2 = 1$, else $|c_P(v)| = 2$.

Note that for a vertex $v \in O_P \cap V(G)$ it is possible that $|c_P(v)| = 0$ even if $|c_L(v)| + |c_R(v)|$ is even and positive since v might be the only intersection of a component with O_P. In order to verify that the encoding formed from two states of e_L and e_R corresponds to

a labeling with each component touching O_P, we use an auxiliary graph A with $V(A) = (O_L \cup O_R) \cap V(G)$ and $\{v, w\} \in E(A)$ if v and w both are in one component of $G_{\mathcal{P}[e_L]}$ and $G_{\mathcal{P}[e_R]}$, respectively. Every component of A must have a vertex in $O_P \cap V(G)$. For every encoding of $\boldsymbol{t}_P$, we set $W_P = min\{W_P, W_L + W_R\}$.

For the root edge $\{r, s\}$ and its children e' and e'' note that $(O_{e'} \cup O_{e''}) \cap V(G) = I$ and $O_{\{r,s\}} = \emptyset$. Hence, for every $v \in V(G_{\mathcal{P}[\{r,s\}]})$ it must hold that $deg_{\mathcal{P}[\{r,s\}]}(v)$ is positive and even, and that the auxiliary graph A is connected. The optimal Eulerian labeling of G results from $min_{\boldsymbol{t}_{\{r,s\}}}\{W_r\}$.

Analyzing the algorithm, we obtain the following lemma.

Lemma 2. Planar Graph TSP *on a graph G with branchwidth at most ℓ can be solved in time $O(2^{4.6496\ell}\ell n + n^3)$.*

Proof. Assume three adjacent edges e_P, e_L, and e_R of T with $|O_L| = |O_R| = |O_P| = \ell$ and that there are no portal vertices. Thus we have $|I| = |D \cap O_L| = |D \cap O_R| = \frac{\ell}{2}$.

By just checking every combination of ℓ-tuples from O_L and O_R we obtain a bound of $O(\ell 6^{2\ell})$ for our algorithm.

This bound can be improved by using the fact that for all vertices $u \in I$ we want the sum of the assignments to be even, i.e., $(|c_L(u)| + |c_R(u)|)\, mod\, 2 = 0$.

We define $Q(\ell, m_1, m_2)$ as the number of ℓ-tuples over ℓ vertices of O_L and O_R, respectively, where the $\{0, 1_[, 1_\square, 2_[, 2_\square,]\}$ assignments for vertices from I is fixed and contains m_1 vertices of odd $\mathcal{P}[e]$-degree and m_2 vertices of even $\mathcal{P}[e]$-degree. The only freedom is thus in the $\ell/2$ vertices in $D \cap O_L$ and $D \cap O_R$, respectively:

$$\begin{aligned} Q(\ell, m_1, m_2) &= O(\sum_{i=0}^{\frac{\ell}{2}} \sum_{j=0}^{\frac{\ell}{2}-i} \binom{\frac{\ell}{2}}{i} \binom{\frac{\ell}{2} - i}{j} 1^{\frac{\ell}{2}-i-j} (\frac{5}{2})^{i+m_1} (\frac{5}{2})^{j+m_2}) \\ &= O(6^{\frac{\ell}{2}} (\frac{5}{2})^{m_1} (\frac{5}{2})^{m_2}) \end{aligned} \tag{8}$$

This expression is a summation over the number of vertices of odd and even $\mathcal{P}[e]$-degree in $D \cap O_L$ and $D \cap O_R$, respectively. The terms $\binom{\frac{\ell}{2}}{i}$ and $\binom{\frac{\ell}{2}-i}{j}$ count the possible locations for the vertices of odd and even $\mathcal{P}[e]$-degree, respectively, whereas $(\frac{5}{2})^{i+m_1}$ and $(\frac{5}{2})^{i+m_2}$ count the number of those assignments. The $1^{\frac{\ell}{2}-i-j}$ is left in the formula to represent the assignment of $\mathcal{P}[e]$-degree zero to the remaining $\ell/2 - i - j$ vertices.

We define $C(\ell)$ as the number of possibilities of forming an ℓ-tuple from O_P. We sum over i and j: the number of vertices of odd and even $\mathcal{P}[e]$-degree in the assignment for I:

$$C(\ell) = \sum_{i=0}^{\frac{\ell}{2}} \sum_{j=0}^{\frac{\ell}{2}-i} \binom{\frac{\ell}{2}}{i} \binom{\frac{\ell}{2} - i}{j} 5^{\frac{\ell}{2}-i-j} Q(\ell, i, j)^2 \tag{9}$$

The term $5^{\frac{\ell}{2}-i}$ denotes the number of ways the vertices of I can be assigned from one side with $\mathcal{P}[e]$-degree zero and from the other side with even $\mathcal{P}[e]$-degree. Straightforward

calculation yields:

$$C(\ell) = O(\sum_{i=0}^{\frac{\ell}{2}} \sum_{j=0}^{i} \binom{\frac{\ell}{2}}{i} \binom{\frac{\ell}{2}-i}{j} 5^{\frac{\ell}{2}-i-j} 6^{\ell} (\frac{5}{2})^{2i} (\frac{5}{2})^{2j})$$
$$= O((6\sqrt{17.5})^{\ell}) \quad (10)$$

We obtain an overall running time of $O(6^{\ell}(\frac{35}{2})^{\frac{\ell}{2}}\ell n)$.

□

Forbidding several components Again we can further improve upon the previous bound by only forming encodings that do not create several components. In contrast to cycles, the components can be formed at the vertices in I with numerical part 1 and 2 in both O_L and O_R. But we only consider vertices with even sum of the numerical part of the assignment. Thus, we look separately at the classes of even $\mathcal{P}[e]$-degree and odd $\mathcal{P}[e]$-degree. Without loss of generality consider odd $\mathcal{P}[e]$-degree: as in the previous section we want to exclude the case $((1_[, 1_[), (],]))$. Again suppose the two classes: $C_1(i)$ contains all i-tuples $(\ldots, (1_[, 1_[))$, and $C_2(i)$ contains all other i-tuples. Adding $(],])$ to i-tuples from $C_1(i)$ is forbidden, as this will lead to a single component. Addition of $(1_[, 1_[)$ to i-tuples of both, $C_1(i)$ and $C_2(i)$ gives us the $i+1$-tuples of class $C_1(i+1)$. We obtain the matrix $A = \begin{pmatrix} 1 & 1 \\ 5 & \frac{21}{4} \end{pmatrix}$ with largest eigenvalue $z = \frac{\sqrt{77}+9}{2} \leq 6.2097$. We can insert z for both, the odd and the even valued vertices separately in Equation (10):

$$C(\ell) = O(\sum_{i=0}^{\frac{\ell}{2}} \sum_{j=0}^{i} \binom{\frac{\ell}{2}}{i} \binom{\frac{\ell}{2}-i}{j} 5^{\frac{\ell}{2}-i-j} 6^{\ell} z^i z^j)$$
$$= O((6\sqrt{17.4195})^{\ell}) \quad (11)$$

We obtain the following result:

Theorem 3. Planar Graph TSP *is solvable in time* $O(2^{9.8594\sqrt{n}}n^{3/2} + n^3) = O(2^{9.8594\sqrt{n}})$.

5 Variants

In this section we will discuss results on other non-local problems on planar graphs.

Hamiltonian Path. The problem of finding the minimum weight Planar Hamiltonian path is closely related to Planar Hamiltonian Cycle. The main difference is that we now have some more freedom in the allowed partial labelings, as there can be at most two vertices not on a noose having degree 1. It is clear that this only contributes a constant factor to the total running time, yielding the following theorem.

Theorem 4. Planar Hamiltonian Path *is solvable in time* $O(2^{6.903\sqrt{n}})$.

Longest Cycle and Longest Path. The problem of Planar Longest Cycle (Path) is, given a weighted planar graph, find the cycle (path) with the largest sum of edge weights. Let C be a cycle in G. For an edge e of an sc-decomposition tree T, the noose O_e can affect C in two ways: Either cycle C is partitioned by O_e such that in G_e the remains of C are disjoint paths, or C is not touched by O_e and thus is completely in G_e or $G \setminus E(G_e)$.

With the same encoding as for Planar Hamiltonian Cycle, we add a counter for all states $\boldsymbol{t}_e$ which is initialized by 0 and counts the maximum number of edges over all possible vertex-disjoint paths represented by one $\boldsymbol{t}_e$. In contrast to Planar Hamiltonian Cycle, we allow for every vertex $v \in I$ that $|c_L(v)| + |c_R(v)| = 0$ in order to represent the isolated vertices. A cycle as a connected component is allowed if all other components are isolated vertices. Then all other vertices in $V(G) \setminus V(G_P)$ of the residual part of T must be of value 0. Implementing a counter Z for the actual longest cycle, a state in $\boldsymbol{t}_P$ consisting of only 0's represents a collection of isolated vertices with Z storing the longest path in G_P without vertices in $\mathrm{mid}(e)$. At the root edge, Z gives the size of the longest cycle. Analysis is similar to that of Planar Hamiltonian Cycle, we get a slightly worse running time since we have to account for isolated vertices.

By the same argument as for Planar Hamiltonian Path we see that Planar Longest Path has the same running time as Planar Longest Cycle. Thus we have the following theorem.

Theorem 5. Planar Longest Cycle *and* Planar Longest Path *are solvable in time* $O(2^{7.223\sqrt{n}})$.

Planar k-Cycle The Planar k-Cycle problem asks for a given planar graph G to find a cycle of length at least a parameter k. The algorithm on Planar Longest Cycle can be used for obtaining parameterized algorithms by adopting the techniques from [9, 13].

Before we proceed, let us remind the notion of a minor. A graph H obtained by a sequence of edge-contractions from a graph G is said to be a *contraction* of G. H is a *minor* of G if H is the subgraph of some contraction of G. Let us note that if a graph H is a minor of G and G contains a cycle of length at least k, then so does G.

We need the following combination of statements (4.3) in [23] and (6.3) in [22].

Theorem 6 ([22]). *Let $k \geq 1$ be an integer. Every planar graph with no $(k \times k)$-grid as a minor has branchwidth at most $4k - 3$.*

It easy to check that every $(\sqrt{k} \times \sqrt{k})$-grid, $k \geq 2$, contains a cycle of length at least $k - 1$. This observation combined with Theorem 6 suggests the following parameterized algorithm. Given a planar graph G and integer k, first compute the branchwidth of G. If the branchwidth of G is at least $4\sqrt{k+1} - 3$ then by Theorem 6, G contains a $(\sqrt{k+1} \times \sqrt{k+1})$-grid as a minor and thus contains a cycle of length at least k. If the branchwidth of G is less than $4\sqrt{k+1} - 3$ we can find the longest cycle in G in time $O(2^{13.6\sqrt{k}}\sqrt{k}\, n + n^3)$. We conclude with the following theorem.

Theorem 7. Planar k-Cycle *is solvable in time* $O(2^{13.6\sqrt{k}}n + n^3)$.

By standard techniques (see for example [11]) the recognition algorithm for Planar k-Cycle can easily be turned into a constructive one.

Minimum Number Cycle Cover. Minimum Number (Cost) Cycle Cover asks for a minimum number (cost) of vertex disjoint cycles that cover the vertex set of the input graph. The algorithm can be implemented as a variant of Planar Hamiltonian Cycle algorithm, with the additional freedom of allowing cycles in the merging step. Thus the result from Equation (6) can be used directly, leading to a running time of $O(n^{\frac{3}{2}}2^{6.987\sqrt{n}})$.

Problems with tree-like solutions. The problem Connected Dominating Set asks for a minimum Dominating Set that induces a connected subgraph. See [10] for a subexponential algorithm on graphs of bounded outerplanarity. Connected Dominating Set can be formulated as Max Leaf Problem where one asks for a spanning tree with the maximum number of leaves.

For the state of the vertices on the nooses we can use an encoding with symbols $0_0, 0_1, 1_0, 1_[, 1_\square, 1_]$. The numerical part indicates whether (1) or not (0) a vertex is an inner node of the solution spanning tree. The indices for the vertices labeled with a 1 encode to which connected component they belong, 1_0 is an isolated vertex that becomes an inner node. The indices for the leaves 0 indicate if a vertex is connected (1) or not (0) to any vertex marked as an inner node. Using our technique, we obtain for Connected Dominating Set a running time of $O(2^{9.822\sqrt{n}})$.

The Steiner Tree of some subset X of the vertices of a planar graph G is a minimum-weight connected subgraph of G that includes X. It is always a tree; thus, we only encode connected subgraphs by using four symbols $0, [,], \square$. Here, $[,], \square$ mark the first, the last and all other vertices of a component and 0 marks isolated vertices and vertices that are the only intersection of a component and the noose. Note that every vertex of X must be part of a component, whereas the vertices of $V \setminus X$ must not. We obtain for Steiner Tree a running time of $O(2^{8.488\sqrt{n}})$.

In Feedback Vertex Set on an undirected planar graph G, one is asked to find a set Y of vertices of minimum cardinality such that every cycle of G passes through at least one vertex of Y. Feedback Vertex Set is equivalent to the problem: find an *induced* forest F in G with vertex set $V(F)$ of maximum cardinality. It holds that $V(G) \setminus Y = V(F)$. We are able to encode induced connected subgraphs with our technique. We mark if a vertex is in $V(F)$ or not. Every edge of G is an edge in the forest if its incident vertices are in $V(F)$. We can solve Feedback Vertex Set in time $O(2^{9.264\sqrt{n}})$.

In the parameterized version of the problem, k-Feedback Vertex Set, we ask if Y is of size at most parameter k. We improve the $2^{O(\sqrt{k}\log k)}n^{O(1)}$ algorithm in [17] to $2^{O(\sqrt{k})}n^{O(1)}$ by using the *bidimensionality* of k-Feedback Vertex Set (see [9] for more information). If a problem on graphs of bounded treewidth tw is solvable in time $2^{O(tw)}n^{O(1)}$ and its parameterized version with parameter k is bidimensional then it is solvable in time $2^{O(\sqrt{k})}n^{O(1)}$.

6 Concluding Remarks

In this paper we introduced a new algorithmic design technique based on geometric properties of branch decompositions. Our technique can be also applied to construct $2^{O(\sqrt{n})} \cdot n^{O(1)}$-time algorithms for a variety of cycle, path, or tree subgraph problems in planar graphs like Hamiltonian Path, Longest Path, and Connected Dominating Set, and Steiner Tree amongst others. An interesting question here is if the technique can be extended to more general problems, like Subgraph Isomorphism. For example, Eppstein [12] showed that Planar Subgraph Isomorphism problem with pattern of size k can be solved in time $2^{O(\sqrt{k}\log k)}n$. Can we get rid of the logarithmic factor in the exponent (maybe in exchange to a higher polynomial degree)?

The results of Cook & Seymour [7] on using branch decompositions to obtain high-quality tours for (general) TSP show that branch decomposition based algorithms run much faster than their worst case time analysis would indicate.

Together with our preliminary experience on the implementation of a similar algorithm technique for solving Planar Vertex Cover in [2], we conjecture that sc-decomposition based algorithms perform much faster in practice.

References

[1] 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 (2002), pp. 461–493.

[2] J. Alber, F. Dorn, and R. Niedermeier, *Experimental evaluation of a tree decomposition-based algorithm for vertex cover on planar graphs*, Discrete Applied Mathematics, 145 (2005), pp. 219–231.

[3] J. Alber, H. Fernau, and R. Niedermeier, *Graph separators: a parameterized view*, Journal of Computer and System Sciences, 67 (2003), pp. 808–832.

[4] ———, *Parameterized complexity: exponential speed-up for planar graph problems*, Journal of Algorithms, 52 (2004), pp. 26–56.

[5] S. Arora, M. Grigni, D. Karger, P. N. Klein, and A. Woloszyn, *A polynomial-time approximation scheme for weighted planar graph TSP*, in 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), New York, 1998, ACM, pp. 33–41.

[6] H. L. Bodlaender, *A tourist guide through treewidth*, Acta Cybernet., 11 (1993), pp. 1–21.

[7] W. Cook and P. Seymour, *Tour merging via branch-decomposition*, INFORMS Journal on Computing, 15 (2003), pp. 233–248.

[8] V. G. Deĭneko, B. Klinz, and G. J. Woeginger, *Exact algorithms for the Hamiltonian cycle problem in planar graphs*, Oper. Res. Lett., (2006), p. to appear.

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

[10] E. D. Demaine and M. T. Hajiaghayi, *Bidimensionality: new connections between FPT algorithms and PTASs*, in 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), New York, 2005, ACM, pp. 590–601.

[11] R. G. Downey and M. R. Fellows, *Parameterized complexity*, Springer-Verlag, New York, 1999.

[12] D. Eppstein, *Subgraph isomorphism in planar graphs and related problems*, Journal of Graph Algorithms and Applications, 3 (1999), pp. 1–27.

[13] F. V. Fomin and D. M. Thilikos, *New upper bounds on the decomposability of planar graphs*, Journal of Graph Theory, 51 (2006), pp. 53–81.

[14] Q.-P. Gu and H. Tamaki, *Optimal branch-decomposition of planar graphs in $O(n^3)$*, in 32nd International Colloquium on Automata, Languages and Programming (ICALP), 2005, pp. 373–384.

[15] M. Held and R. M. Karp, *A dynamic programming approach to sequencing problems*, Journal of SIAM, 10 (1962), pp. 196–210.

[16] R. Z. Hwang, R. C. Chang, and R. C. T. Lee, *The searching over separators strategy to solve some NP-hard problems in subexponential time*, Algorithmica, 9 (1993), pp. 398–423.
[17] T. Kloks, C. M. Lee, and J. Liu, *New algorithms for k-face cover, k-feedback vertex set, and k-disjoint cycles on plane and planar graphs*, in 28th International Workshop on Graph-theoretic Concepts in Computer Science, WG 2002, vol. 2573 of Lecture Notes in Comput. Sci., Berlin, 2002, Springer, pp. 282–295.
[18] G. Kreweras, *Sur les partition non croisées d'un circle*, Discrete Mathematics, 1 (1972), pp. 333–350.
[19] E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, eds., *The traveling salesman problem*, John Wiley & Sons Ltd., Chichester, 1985.
[20] R. J. Lipton and R. E. Tarjan, *A separator theorem for planar graphs*, SIAM Journal on Applied Mathematics, 36 (1979), pp. 177–189.
[21] ———, *Applications of a planar separator theorem*, SIAM Journal on Computing, 9 (1980), pp. 615–627.
[22] N. Robertson, P. Seymour, and R. Thomas, *Quickly excluding a planar graph*, Journal of Combinatorial Theory Series B, 62 (1994), pp. 323–348.
[23] N. Robertson and P. D. Seymour, *Graph minors. X. Obstructions to tree-decomposition*, Journal of Combinatorial Theory Series B, 52 (1991), pp. 153–190.
[24] P. Seymour and R. Thomas, *Call routing and the ratcatcher*, Combinatorica, 15 (1994), pp. 217–241.
[25] W. D. Smith and N. C. Wormald, *Geometric separator theorems and applications*, in The 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1998), IEEE Computer Society, 1998, pp. 232–243.