REPORTS
IN
INFORMATICS

ISSN 0333-3590

Fast subexponential algorithm for non-local problems on graphs of bounded genus

Frederic Dorn and Fedor V. Fomin and Dimitrios M. Thilikos

REPORT NO 320 **April 2006**

Department of Informatics

UNIVERSITY OF BERGEN

Bergen, Norway

This report has URL http://www.ii.uib.no/publikasjoner/texrap/pdf/2006-320.pdf

Reports in Informatics from Department of Informatics, University of Bergen, Norway, is available at
http://www.ii.uib.no/publikasjoner/texrap/.

Requests for paper copies of this report can be sent to:

Department of Informatics, University of Bergen, Høyteknologisenteret,
P.O. Box 7800, N-5020 Bergen, Norway

Fast subexponential algorithm for non-local problems on graphs of bounded genus

Frederic Dorn and Fedor V. Fomin*
Department of Informatics
University of Bergen
PO Box 7800, 5020 Bergen, Norway
{*dorn,fomin*}*@ii.uib.no*

Dimitrios M. Thilikos†
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Barcelona, Spain
sedthilk@lsi.upc.edu

Abstract

We give a general technique for designing fast subexponential algorithms for several graph problems whose instances are restricted to graphs of bounded genus. We use it to obtain time $2^{O(\sqrt{n})}$ algorithms for a wide family of problems such as Hamiltonian Cycle, Σ-embedded Graph Travelling Salesman Problem, Longest Cycle, and Max Leaf Tree. For our results, we combine planarizing techniques with dynamic programming on special type branch decompositions. Our techniques can also be used to solve parameterized problems. Thus, for example, we show how to find a cycle of length p (or to conclude that there is no such a cycle) on graphs of bounded genus in time $2^{O(\sqrt{p})} \cdot n^{O(1)}$.

Keywords: Exact and parameterized algorithms, bounded genus, treewidth, branchwidth, travelling salesman problem, Hamiltonian cycle.

*Additional support by the Research Council of Norway.
†Supported by the Spanish CICYT project TIN-2004-07925 (GRAMMARS).

1 Introduction

Many common computational problems are NP-hard and therefore do not seem to be solvable by efficient (polynomial time) algorithms. However, while NP-hardness is a good evidence for the intractability of a problem, in many cases, there is a real need for exact solutions. Consequently, an interesting and emerging question is to develop techniques for designing fast exponential or, when possible, sub-exponential algorithms for hard problems (see [17]).

The algorithmic study of graphs that can be embedded on a surface of small genus, and planar graphs in particular, has a long history. The first powerful tool for the design of sub-exponential algorithms on such graphs was the celebrated Lipton-Tarjan planar separator theorem [10, 11] and its generalization on graphs of bounded genus [8]. According to these theorems, an n-vertex graph of fixed genus can be "separated" into two roughly equal parts by a separator of size $O(\sqrt{n})$. This approach permits the use of a "divide and conquer" technique that provides subexponential algorithms of running time $2^{O(\sqrt{n})}$ for a wide range of combinatorial problems.

A similar approach is based on graph decompositions [7]. Here instead of separators one uses decompositions of small width, and instead of "divide and conquer" techniques, dynamic programming (here we refer to tree or branch decompositions – see Section 2 for details). The main idea behind this approach is very simple: Suppose that for a problem $\mathcal{P}$ we are able to prove that for every n-vertex graph G of branchwidth at most ℓ, the problem $\mathcal{P}$ can be solved in time $2^{O(\ell(G))} \cdot n^{O(1)}$. Since the branchwidth of an n-vertex graph of a fixed genus is $O(\sqrt{n})$, we have that $\mathcal{P}$ is solvable on G in time $2^{O(\sqrt{n})} \cdot n^{O(1)}$.

For some problems like Minimum Vertex Cover or Minimum Dominating Set, such an approach yields directly algorithms of running time $2^{O(\sqrt{n})} \cdot n^{O(1)}$ on graphs of bounded genus. However, for some problems, like Hamiltonian Cycle, Σ-embedded Graph TSP, Max Leaf Tree, and Steiner Tree, branchwidth arguments do not provide us with time $2^{O(\sqrt{n})} \cdot n^{O(1)}$ algorithms. The reason is that all these problems are "non-local" and despite many attempts, no time $2^{o(\ell(G) \log \ell)} \cdot n^{O(1)}$ algorithm solving these problems on graphs of branchwidth at most ℓ is known.

Recently, it was observed by several authors that if a graph G is not only of branchwidth at most ℓ but is also planar, then for a number of "non-local" problems the $\log \ell$ overhead can be removed [4, 5], resulting in time $2^{O(\sqrt{n})} \cdot n^{O(1)}$ algorithms on planar graphs. Similar result can be obtained by making use of separators [2].

It is a common belief that almost every technique working on planar graphs can be extended on graphs embedded on a surface of bounded genus. However, this is not always a straightforward task. The main difficulty in generalizing planar graph techniques [2, 4, 5] to graphs of bounded genus is that all these techniques are based on partitioning a graph embedded on a plane by a closed curve into smaller pieces. Deineko et al. use cyclic separators of triangulations [2], Demaine and Hajiaghayi use layers of k-outerplanar graphs [4], and Dorn et al. sphere cut decompositions [5]. But the essence of all these techniques is that, roughly speaking, the situation occurring in the "inner" part of the graph bounded by the closed curve can be represented in a compact way by Catalan structures. None of these tools works for graphs of bounded genus—separators are not cyclic anymore, nor are there sphere cut decompositions and k-outerplanarity in non-planar graphs.

In this paper we provide a method to design fast subexponential algorithms for graphs of bounded genus for a wide class of combinatorial problems. Our algorithms are "fast" in the sense that they avoid the $\log n$ overhead and also because the constants hidden in the big-Oh of the exponents are reasonable. The technique we use is based on reduction of the bounded genus instances of the problem to *planar* instances of a more general graph problem on planar graphs where Catalan structure arguments are still possible. Such a reduction employs several results from topological graph theory concerning graph structure and noncontractible cycles of non-planar embeddings.

Our techniques, combined with the excluded grid theorem for graphs of bounded genus and bidimensionality arguments [3] provide also faster *parameterized* algorithms. For example we introduce the first time $2^{O(\sqrt{p})} \cdot n^{O(1)}$ algorithm for parameterized p-CYCLE which asks, given a positive integer p and a n-vertex graph G, whether G has a cycle of length at least p. Similar results can be obtained for other parameterized versions of non-local problems.

This paper is organized as follows. Towards simplifying the presentation of our results we decided to demonstrate how our approach works for the HAMILTONIAN CYCLE problem. Later, at the end of Section 4, we will explain how it can be applied to other combinatorial problems. We start with some basic definitions in Section 2 and some results from topological graph theory. Section 3 is devoted to the solution of HAMILTONIAN CYCLE problem (which asks if a given graph G has a cycle containing all its vertices) on torus-embedded graphs. These graphs already inherit all "nasty" properties of non-planar graphs and all difficulties arising on surfaces of higher genus appear for torus-embedded graphs. However, the case of torus-embedded graphs is still sufficiently simple to exemplify the minimization technique used to obtained reasonable constants in the exponent. In Section 4, we explain how the results on torus-embedded graphs can be extended for any graphs embedded in a surface of fixed genus. Also in this section we discuss briefly applications of our results to parameterized algorithms on graphs of bounded genus.

2 Definitions and preliminary results

In this section we will give a series of definitions and results that will be useful for the presentation of the algorithms in Sections 3 and 4.

Surface embeddible graphs. We use the notation $V(G)$ and $E(G)$, for the set of the vertices and edges of G. A *surface* Σ is a compact 2-manifold without boundary (we always consider connected surfaces). We denote by $\mathbb{S}_0$ the sphere $(x, y, z \mid x^2 + y^2 + z^2 = 1)$ and by $\mathbb{S}_1$ the torus $(x, y, z \mid z^2 = 1/4 - (\sqrt{x^2 + y^2} - 1)^2)$. A *line* in Σ is subset homeomorphic to $[0, 1]$. An O-arc is a subset of Σ homeomorphic to a circle. Whenever we refer to a Σ-*embedded graph* G we consider a 2-cell embedding of G in Σ. To simplify notations we do not distinguish between a vertex of G and the point of Σ used in the drawing to represent the vertex or between an edge and the line representing it. We also consider G as the union of the points corresponding to its vertices and edges. That way, a subgraph H of G can be seen as a graph H where $H \subseteq G$. We call by *region* of G any connected component of $(\Sigma \setminus E(G)) \setminus V(G)$. (Every region is an open set.) A subset of Σ meeting the drawing only in vertices of G is called G-*normal*. If an O-arc is G-normal then we call it *noose*. The length of a noose N is the number of its vertices and we denote it by $|N|$. Representativity [13] is the measure how dense a graph is embedded on a surface. The *representativity* (or *face-width*) $\mathbf{rep}(G)$ of a graph G embedded in surface $\Sigma \neq \mathbb{S}_0$ is the smallest length of a noncontractible noose in Σ. In other words, $\mathbf{rep}(G)$ is the smallest number k such that Σ contains a noncontractible (non null-homotopic in Σ) closed curve that intersects G in k points. Given a Σ-embedded graph G, its *radial graph* (also known as vertex-face graph) is defined as the the graph R_G that has as vertex set the vertices and the faces of G and where an edge exists iff it connects a face and a vertex incident to it in G (R_G is also a σ-embedded graph). If the intersection of a noose with any region results into a connected subset, then we call such a noose *tight*. Notice that each tight noose N in a Σ-embedded graph G, corresponds to some cycle C of its radial graph R_G (notice that the length of such a cycle is $2 \cdot |N|$). Also any cycle C of R_G is a tight noose in G. As it was shown by Thomassen in [16] (see also Theorem 4.3.2 of [12]) a shortest noncontractible cycle in a graph embedded on a surface can be found in polynomial time. By Proposition 5.5.4 of [12]) a noncontractible noose of minimum size is always a tight noose, i.e. corresponds to a cycle of the radial graph. Thus we have the following proposition.

Proposition 1. *There exists a polynomial time algorithm that for a given Σ-embedded graph G, where $\Sigma \neq \mathbb{S}_0$, finds a noncontractible tight noose of minimum size.*

The Euler genus of a surface Σ is $\mathbf{eg}(\Sigma) = \min\{2\mathbf{g}(\Sigma), \tilde{\mathbf{g}}(\Sigma)\}$ where $\mathbf{g}$ is the orientable genus and $\tilde{\mathbf{g}}$ the nonorientable genus. We need to define the graph obtained by *cutting along* a noncontractible tight noose N. We suppose that for any $v \in N \cap V(G)$, there exists an open disk Δ containing v and such that for every edge e adjacent to v, $e \cap \Delta$ is connected. We also assume that $\Delta \setminus N$ has two connected components Δ_1 and Δ_2. Thus we can define partition of $N(v) = N_1(v) \cup N_2(v)$, where $N_1(v) = \{u \in N(v) \colon \{u,v\} \cap \Delta_1 \neq \emptyset\}$ and $N_2(v) = \{u \in N(v) \colon \{u,v\} \cap \Delta_2 \neq \emptyset\}$. Now for each $v \in N \cap V(G)$ we *duplicate* v: (a) remove v and its incident edges (b) introduce two new vertices v^1, v^2 and (c) connect v^i with the vertices in $N_i, i = 1, 2$. v^1 and v^2 are vertices of the new G-normal O-arcs N_X and N_Y that border Δ_1 and Δ_2, respectively. We call N_X and N_Y *cut-nooses.* Note that cut-nooses are not necessarily tight (In other words, a cut-noose can enter and leave a region of G several times.) The following lemma is very useful in proofs by induction on the genus. The first part of the lemma follows from Proposition 4.2.1 (corresponding to surface separating cycle) and the second part follows from Lemma 4.2.4 (corresponding to non-separating cycle) in [12].

Proposition 2. *Let G be a Σ-embedded graph where $\Sigma \neq \mathbb{S}_0$ and let G' be a graph obtained from G by cutting along a noncontractible tight noose N on G. One of the following holds*
• *G' can be embedded in a surface with Euler genus strictly smaller than $\mathbf{eg}(\Sigma)$.*
• *G' is the disjoint union of graphs G_1 and G_2 that can be embedded in surfaces Σ_1 and Σ_2 such that $\mathbf{eg}(\Sigma) = \mathbf{eg}(\Sigma_1) + \mathbf{eg}(\Sigma_2)$ and $\mathbf{eg}(\Sigma_i) > 0$, $i = 1, 2$.*

Branchwidth. A *branch decomposition* of a graph G is a pair $\langle T, \mu \rangle$, where T is a tree with vertices of degree one or three and μ is a bijection from the set of leaves of T to $E(G)$. For a subset of edges $X \subseteq E(G)$ let $\delta_G(X)$ be the set of all vertices incident to edges in X and $E(G) \setminus X$. For each edge e of T, let $T_1(e)$ and $T_2(e)$ be the sets of leaves in two components of $T \setminus e$. For any edge $e \in E(T)$ we define the *middle set* as $\mathbf{mid}(e) = \bigcup_{v \in T_1(e)} \delta_G(\mu(v))$. The *width* of $\langle T, \mu \rangle$ is the maximum size of a middle set over all edges of T, and the *branch-width* of G, $\mathbf{bw}(G)$, is the minimum width over all branch decompositions of G. For a $\mathbb{S}_0$-embedded 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 and the two subgraphs G_1 and G_2 induced by the edges in $\mu(T_1(e))$ and $\mu(T_2(e))$, there exists a tight noose O_e bounding 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 $\mathbf{mid}(e)$ and its length is $|\mathbf{mid}(e)|$. Clockwise traversing of O_e in the drawing G defines the cyclic ordering π of $\mathbf{mid}(e)$. We always assume that in an sc-decomposition the vertices of every middle set $\mathbf{mid}(e) = V(G_1) \cap V(G_2)$ are enumerated according to π. The following result follows from the celebrated ratcatcher algorithm due to Seymour and Thomas [15] (the running time of the algorithm was recently improved in [9]; see also [5]).

Proposition 3. *Let G be a connected $\mathbb{S}_0$-embedded graph without vertices of degree one. There exists an sc-decomposition of G of width $\mathbf{bw}(G)$. Moreover, such a branch decomposition can be constructed in time $O(n^3)$.*

3 Hamiltonicity on torus-embedded graphs

The idea behind solving the Hamiltonian cycle problem on $\mathbb{S}_1$-embedded graphs is to suitably modify the graph G in such a way that the new graph G' is $\mathbb{S}_0$-embedded (i.e. planar) and restate the problem to an equivalent problem on G' that can be solved by dynamic programming on a sc-decomposition of G'. As we will see in Section 4, this procedure is extendable to graphs embedded on surfaces of higher genus.

Let G be an $\mathbb{S}_1$-embedded graph (i.e. a graph embedded in the torus). By Proposition 1, it is possible to find in polynomial time a shortest noncontractible (tight) noose N of G. Let G' be the graph obtained by cutting along N on G. By Proposition 2, G' is $\mathbb{S}_0$-embeddible.

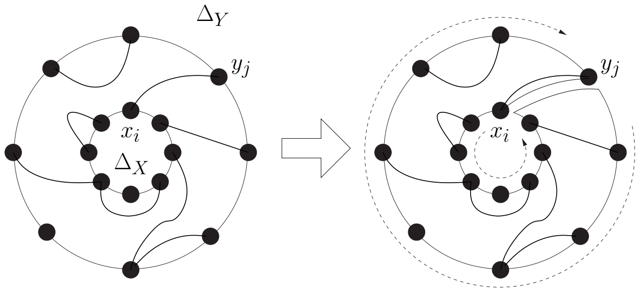

Figure 1: **Cut-nooses.** In the left diagram, one equivalence class of relaxed Hamiltonian sets is illustrated. All paths have endpoints in N_X and N_Y. Fix one path with endpoints x_i and y_i. In the right diagram we create a tunnel along this path. The empty disks Δ_X and Δ_Y are united to a single empty disk. Thus, we can order the vertices bordering the disk to π_{XY}.

Definition 4. A *cut* of a Hamiltonian cycle C in G along a tight noose N is the set of disjoint paths in G' resulting by cutting G along N.

Each cut-noose N_X and N_Y borders an open disk Δ_X and Δ_Y, respectively, with $\Delta_X \cup \Delta_Y = \emptyset$. Let $x_i \in N_X$ and $y_i \in N_Y$ be duplicated vertices of the same vertex in N.

Definition 5. A set of disjoint paths $\mathbf{P}$ in G' is *relaxed Hamiltonian* if:
(P1) Every path has its endpoints in N_X and N_Y.
(P2) Vertex x_i is an endpoint of some path P if and only if y_i is an endpoint of a path $P' \neq P$.
(P3) For x_i and y_i: one is an inner vertex of a path if and only if the other is not in any path.
(P4) Every vertex of $G' \setminus (N_X \cup N_Y)$ is in some path.

A *cut* of a Hamiltonian cycle in G is a relaxed Hamiltonian set in G', but not every relaxed Hamiltonian set in G' forms a Hamiltonian cycle in G. However, given a relaxed Hamiltonian set $\mathbf{P}$ one can check in linear time (by identifying the corresponding vertices of N_X and N_Y) if $\mathbf{P}$ is a cut of Hamiltonian path in G. Two sets of disjoint paths $\mathbf{P} = (P_1, P_2, \ldots, P_k)$ and $\mathbf{P'} = (P'_1, P'_2, \ldots, P'_k)$ are *equivalent* if for every $i \in \{1, 2, \ldots, k\}$, the paths P_i and P'_i have the same endpoints and an inner vertex in one set is also an inner vertex in the other set.

Lemma 6. *Let G' be a $\mathbb{S}_0$-embedded graph obtained from a $\mathbb{S}_1$-embedded graph G by cutting along a tight noose N. The number of different equivalence classes of relaxed Hamiltonian sets in G' is $O(\frac{k^2}{2}\, 2^{3k-2} + 2^{3k})$, where k is the length of N.*

Proof. In [5] it is argued that the number of non-crossing paths with its endpoints in one noose corresponds to a number of algebraic terms, namely the Catalan numbers. Here we deal with two cut-nooses and our intention is to transform them into one cut-noose. For this, assume two vertices $x_i \in N_X$ and $y_j \in N_Y$ being two fixed endpoints of a path $P_{i,j}$ in a relaxed Hamiltonian set $\mathbf{P}$. We look at all possible residual paths in $\mathbf{P} \setminus P_{i,j}$ and we observe that no path crosses $P_{i,j}$ in the $\mathbb{S}_0$-embedded graph G' . So we are able to 'cut' the sphere $\mathbb{S}_0$ along $P_{i,j}$ and, that way, create a "tunnel" between Δ_X and Δ_Y unifying them to a single disk Δ_{XY}. Take the counter-clockwise order of the vertices of N_X beginning with x_i and concatenate N_Y in clockwise order with y_j the last vertex. We denote the new cyclic ordering by π_{XY} (see Figure 1 for an example) . In π_{XY}, let a, b, c, d be four vertices where $x_i < a < b < c < d < y_j$. Notice that if there is a path $P_{a,c}$ between a and c, then there is no path between b and d since such a path either crosses $P_{a,c}$ or $P_{i,j}$. This means that we can encode the endpoints of each path with two symbols, one for the beginning and one for the ending of a path. The encoding corresponds to the brackets of an algebraic term. The number of algebraic terms is defined by the Catalan numbers. We say that $\mathbf{P}$ has a *Catalan structure*. With $k = |N_X| = |N_Y|$ and $x_i, y_j \in P_{i,j}$ fixed, there are $O(2^{2k-2})$ sets of paths having different endpoints and non-crossing $P_{i,j}$. An upper bound for the overall number of sets of paths satisfying (P1) is then $O(\frac{k^2}{2}\, 2^{2k-2} + 2^{2k})$ with the first summand counting all sets of paths for each fixed pair of endpoints x_i, y_j. The second summand counts the number of sets of paths when N_X and N_Y are not connected by any path. That is, each path has both endpoints in either only N_X or only N_Y. We now count the number of equivalent relaxed Hamiltonian sets $\mathbf{P}$. Apparently, in a feasible solution, if a vertex $x_h \in N_X$ is an inner vertex of a path, then $y_h \in N_Y$ does not belong to any path and vice versa. With (P3), there are two more possibilities for the pair of vertices x_h, y_h to correlate with a path. With $|N_X| = |N_Y| = k$, the overall upper bound of equivalent sets of paths is $O(\frac{k^2}{2}\, 2^{3k-2} + 2^{3k})$. □

We call a *candidate* $\mathbf{C}$ of an equivalence class of relaxed Hamiltonian sets to be a set of paths with vertices only in $N_X \cup N_Y$ satisfying conditions (P1)–(P3). Thus for each candidate we fix a path between N_X and N_Y and define the ordering π_{XY}. By making use of dynamic programming on sc-decompositions we check for each candidate $\mathbf{C}$ if there is a spanning subgraph of the planar graph G' isomorphic to a relaxed Hamiltonian set $\mathbf{P}$ such that $\mathbf{P}$ is equivalent to $\mathbf{C}$.

Instead of looking at the Hamiltonian cycle problem on G we solve the relaxed Hamiltonian set problem on the $\mathbb{S}_0$-embedded graph G' obtained from G: Given a candidate $\mathbf{C}$, i.e. a set of vertex tuples $\mathbf{T} = \{(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)\}$ with $s_i, t_i \in N_X \cup N_Y, i = 1, \ldots, k$ and a vertex set $\mathbf{I} \subset N_X \cup N_Y$. Does there exist a relaxed Hamiltonian set $\mathbf{P}$ such that every (s_i, t_i) marks the endpoints of a path and the vertices of $\mathbf{I}$ are inner vertices of some paths? Our algorithm works as follows: first encode the vertices of $N_X \cup N_Y$ according to $\mathbf{C}$ by making use of the Catalan structure of $\mathbf{C}$ as it follows from the proof of Lemma 6. We may encode the vertices s_i as the 'beginning' and t_i as the 'ending' of a path of $\mathbf{C}$. Using order π_{XY}, we ensure that the beginning is always connected to the next free ending. This allows us to design a dynamic programming algorithm using a small constant number of states. We call the encoding of the vertices of $N_X \cup N_Y$ *base encoding* to differ from the encoding of the sets of disjoint paths in the graph. We proceed with dynamic programming over middle sets of a rooted sc-decomposition $\langle T, \mu, \pi \rangle$ in order to check whether G' contains a relaxed Hamiltonian set $\mathbf{P}$ equivalent to candidate $\mathbf{C}$. As T is a rooted tree, this defines an orientation of its edges towards its root. Let e be an edge of T and let O_e be the corresponding tight noose in $\mathbb{S}_0$. Recall that the tight noose O_e partitions $\mathbb{S}_0$ into two discs which, in turn, induces a partition of the edges of G into two sets. We define as G_e the graph induced by the edge set that corresponds to the "lower side" of e it its orientation towards the root. All paths of $\mathbf{P} \cap G_e$ start and end in O_e and $G_e \cap (N_X \cup N_Y)$. For each G_e, we encode the equivalence classes of sets of disjoint paths with endpoints in O_e. From the leaves to the root for a parent edge and its two children, we update the encodings of the parent middle set with those of the children (for an example of dynamic programming on sc-decompositions, see also [5]). We obtain the algorithm in Figure 2.

In the proof of the following lemma we show how to apply the dynamic programming step of **HamilTor**. The proof is technical, and has been moved to the appendix. But we sketch the main idea here: For a dynamic programming step we need the information on how a tight noose O_e and $N_X \cup N_Y$ intersect and which parts of $N_X \cup N_Y$ are a subset of the subgraph G_e. Define the vertex set $\mathcal{X} = (G_e \setminus O_e) \cap (N_X \cup N_Y)$. G_e is bordered by O_e and $\mathcal{X}$. G_e is partitioned into several edge-disjoint components that we call *partial components*. Each partial component is bordered by a noose that is the union of subsets of O_e and $\mathcal{X}$. Let us remark that this noose is not necessarily tight. The partial components intersect pairwise only in vertices of $\mathcal{X}$ that we shall define as *connectors*. In each partial component we encode a collection of paths with endpoints in the bordering noose using Catalan structures. The union of these collections over all partial components must form a collection of paths in G_e with endpoints in O_e and in $\mathcal{X}$. We ensure that the encoding of the connectors of each two components fit. During the dynamic programming we need to keep track of the base encoding of

Algorithm **HamilTor**
Input: $\mathbb{S}_1$-embedded graph G.
Output: Decision/Construction of the HAMILTONIAN CYCLE problem on G.

Preliminary Step: Cut G along a shortest noncontractible (tight) noose N and output the $\mathbb{S}_0$-embedded graph G' and the cut-nooses N_X,N_Y.
Main step: For all candidates $\mathbf{C}$ of relaxed Hamiltonian sets in G' {
  If $\mathbf{C}$ is equivalent to a Hamiltonian cycle when identifying the duplicated vertices in N_X,N_Y {
    Determine the pair of endpoints (s_i, t_i) that build the first and last vertex in π_{XY}.
    Make a base encoding of the vertices of N_X and N_Y, marking the intersection of $\mathbf{C}$ and $N_X \cup N_Y$.
    Compute a rooted sc-decomposition $\langle T, \mu, \pi \rangle$ of G'.
    From the leaves to the root on each middle set O_e of T bordering G_e {
      Do dynamic programming — find all equivalence classes of sets of disjoint paths in G_e with endpoints in O_e and in $G_e \cap (N_X \cup N_Y)$ with respect to the base encoding of N_X,N_Y.}
    If there exists a relaxed Hamiltonian set $\mathbf{P}$ in G' equivalent to $\mathbf{C}$, then {
    Reconstruct $\mathbf{P}$ from the root to the leaves of T and output corresponding Hamiltonian cycle.} } }
Output "No Hamiltonian Cycle exists".

Figure 2: Algorithm **HamilTor**.

$\mathcal{X}$. We do so by only encoding the vertices of O_e without explicitely memorizing with which vertices of $\mathcal{X}$ they form a path. With several technical tricks we can encode O_e such that two paths with an endpoint in O_e and the other in $\mathcal{X}$ can be connected to a path of $\mathbf{P}$ only if both endpoints in $\mathcal{X}$ are the endpoints of a common path in $\mathbf{C}$.

Lemma 7. *For a given a sc-decomposition $\langle T, \mu, \pi \rangle$ of G' of width ℓ and a candidate $\mathbf{C} = (\mathbf{T}, \mathbf{I})$ the running time of the main step of* **HamilTor** *on* $\mathbf{C}$ *is* $O(2^{5.433\ell} \cdot |V(G')|^{O(1)})$.

To finish the estimation of the running time we need some combinatorial results.

Lemma 8. *Let G be a $\mathbb{S}_1$-embedded graph on n vertices and G' the planar graph obtained by cutting along a noncontractible tight noose of G. Then* $\mathbf{bw}(G') \leq \sqrt{4.5} \cdot \sqrt{n} + 2$.

Proof. (sketch) Let N_X and N_Y be the cut-nooses in G' bordering the empty disks Δ_X and Δ_Y. We will prove the theorem assuming that after cutting along a noncontractible tight noose N of G, all edges with both ends in N are incident to N_X (the general case is a slightly more technical implementation of the same idea). We construct a new graph G^* by removing the vertices of N_Y from G'. Thus $|V(G^*)| = |V(G)| = n$. By [7] there is a sc-decomposition $\langle T, \mu, \pi \rangle$ of G^* of width at most $\sqrt{4.5} \cdot \sqrt{n}$. Δ_Y is part of a region R of G^* bordered by a closed walk C. The neighborhood of N_Y in G' is a subset of the vertices of C in G^*. Let E_Y be the set of edges in G' incident to N_Y. Note that $E(G^*) \cup E_Y = E(G')$ and that E_Y induces a graph that is a subgraph that can be seen as a union of stars whose centers lay on C (this is based on the assumption that no edge has both ends in N_Y). We construct a branch decomposition of G' from $\langle T, \mu, \pi \rangle$ by doing the following. For every edge $x \in E_Y$ we choose edge y of C having a common endpoint v with x and being the next edge of C in counter-clockwise ordering incident to v. Let e_y be the edge of T adjacent to the leaf of corresponding to y. We subdivide e_y by placing a new vertex on it and attach a new leaf corresponding to x. We claim that the width of the new branch decomposition $\langle T', \mu' \rangle$ is at most the width of $\langle T, \mu, \pi \rangle$ plus two. For an edge y of C we may subdivide e_y of T several times creating a subtree T_y. But all the middle sets of the edges T_y have only one vertex in common, namely the common endpoint v. The middle set connecting T_y to T may have up to two more vertices that are, in order of appearance in π_{XY}, the first and the last endpoints of the considered edges of E_Y. Let $E(C)$ be the edge set of C. Since $\langle T, \mu, \pi \rangle$ is a sc-decomposition, we have that for every e of $E(T)$, if the corresponding tight noose O_e bordering G_e^* intersects region R bordered by C, then $E(C) \cap G_e^*$ induces a connected subset of C. Note that in contrast O_e and C may intersect in single vertices only. Thus, O_e and that subset intersect in only two vertices v, w. v and w each have at most one adjacent vertex in N_Y

that is connected to $C \setminus G_e^*$. Hence each middle set of T' has at most two vertices more than the corresponding middle set of T. $\square$

Lemma 9. *Let G be a $\mathbb{S}_1$-embedded graph on n vertices. Then* $\mathbf{rep}(G) \leq \sqrt{4.5} \cdot \sqrt{n} + 2$.

Proof. (sketch) Let G' be the $\mathbb{S}_0$-embedded graph obtained by cutting along a noncontractible tight noose N of G. By Lemma 8, there is a sc-decomposition $\langle T, \mu, \pi \rangle$ of G' of width at most $\sqrt{4.5} \cdot \sqrt{n} + 2$. We subdivide an arbitrary edge e of T into the edges e_1,e_2 and root the tree at the new node r. Assume that for one of e_1, e_2, say e_1, both cut-nooses N_X and N_Y are properly contained in G_{e_1}. We traverse the tree from e_1 to the leaves. We always branch towards a child edge e with middle set O_e such that $N_X \cup N_Y \subset G_e$. At some point we reach an e with either *a*) G_e properly containing exactly one cut-noose or *b*) O_e intersecting both cut-nooses or *c*) G_e properly containing one cut-noose and O_e intersecting the other. In case *c*) we continue traversing from e towards the leafs always branching towards the edge with *c*) until we reach an edge with either *a*) or *b*). In case *a*), tight noose O_e forms a noncontractible tight noose in G, hence the length of O_e must be at least the representativity of G. In case *b*), O_e is the union of two lines with the shortest, say N_1, of length at most $\frac{|O_e|}{2}$. But both endpoints of N_1 are connected in the $\mathbb{S}_1$-embedded graph G by a line N_2 of N of length L with $0 \leq L \leq \frac{|N|}{2}$. N_1 and N_2 form a noncontractible tight noose in G of length at most $\frac{|O_e|}{2} + \frac{|N|}{2}$. Hence, $|O_e| \geq \mathbf{rep}(G)$. $\square$

Putting all together we obtain the following theorem.

Theorem 10. *Let G be a graph on n vertices embedded on a torus $\mathbb{S}_1$. The* Hamiltonian cycle *problem on G can be solved in time $O(2^{17.893\sqrt{n}} \cdot n^{O(1)})$.*

Proof. We run the algorithm **HamilTor** on G. The algorithm terminates positively when the dynamic programming is successful for some candidate of an equivalence class of relaxed Hamiltonian sets and this candidate is a cut of a Hamiltonian cycle. By Propositions 1, Step 0 can be performed in polynomial time. Let k be the minimum length of a noncontractible noose N, and let G' be the graph obtained from G by cutting along N. By Lemma 6, the number of all candidates of relaxed Hamiltonian sets in G' is $O(2^{3k}) \cdot n^{O(1)}$. So the main step of the algorithm is called $O(2^{3k}) \cdot n^{O(1)}$ times. By Proposition 3, an optimal branch decomposition of G' of width ℓ can be constructed in polynomial time. By Lemma 7, dynamic programming takes time $O(2^{5.433\ell}) \cdot n^{O(1)}$. Thus the total running time of **HamilTor** is $O(2^{5.433\ell} \cdot 2^{3k}) \cdot n^{O(1)}$. By Lemma 9, $k \leq \sqrt{4.5} \cdot \sqrt{n} + 2$ and by Lemma 8, $\ell \leq \sqrt{4.5} \cdot \sqrt{n} + 2$, and the theorem follows. $\square$

4 Hamiltonicity on graphs of bounded genus

Now we extend our algorithm to graphs of higher genus. For this, we use the following kind of planarization: We apply Proposition 2 and cut iteratively along shortest noncontractible nooses until we obtain a planar graph G'. If at some step G' is the disjoint union of two graphs G_1 and G_2, we apply Proposition 2 on G_1 and G_2 separately.

Lemma 11. *There exists a polynomial time algorithm that given a Σ-embedded graph G where $\Sigma \neq \mathbb{S}_0$, returns a minimum size noncontractible noose. Moreover, the length of such a noose, $\mathbf{rep}(G)$, is at most $\mathbf{bw}(G) \leq (\sqrt{4.5} + 2 \cdot \sqrt{2 \cdot \mathbf{eg}(\Sigma)})\sqrt{n}$.*

Proof. In order to find a tight noose in G of minimum size we use Proposition 1, and we are looking instead for a shortest noncontractible cycle in R_G. This can be done by the algorithm of Thomassen in [16] (See also Theorem 4.3.2 of [12]). From [6], the branchwidth of an 2-cell-embedded graph on the surface Σ is bounded by $(\sqrt{4.5} + 2 \cdot \sqrt{2\mathbf{eg}(\Sigma)})\sqrt{n}$. By Theorem 4.1 of [14], $\mathbf{rep}(G) \leq \mathbf{bw}(G)$. $\square$

We examine how a shortest noncontractible noose affects the cut-nooses of previous cuts:

Definition 12. Let $\mathcal{K}$ be a family of cycles in G. We say that $\mathcal{K}$ satisfies the *3-path-condition* if it has the following property. If x, y are vertices of G and P_1, P_2, P_3 are internally disjoint paths joining x and y, and if two of the three cycles $C_{i,j} = P_i \cup P_j$, $(1 \leq i < j \leq 3)$ are not in $\mathcal{K}$, then also the third cycle is not in $\mathcal{K}$.

Proposition 13. *(Mohar and Thomassen [12]) The family of Σ-noncontractible cycles of a Σ-embedded graph G satisfies the 3-path-condition.*

Proposition 13 is useful to restrict the number of ways not only on how a shortest noncontractible tight noose may intersect a face but as well on how it may intersect the vertices incident to a face. The proof of the following lemma is moved to the appendix.

Lemma 14. *Let G be Σ-embedded and F a face of G bordered by $V_1 \subseteq V(G)$. Let $\overline{F} := V_1 \cup F$. Let N_s be a shortest noncontractible (tight) noose of G. Then one of the following holds*

1) $N_s \cap \overline{F} = \emptyset$.
2.1) $N_s \cap F = \emptyset$ *and* $|N_s \cap V_1| = 1$.
2.2) $N_s \cap F = \emptyset, N_s \cap V_1 = \{x, y\}$, *and x and y are both incident to one more face different than F which is intersected by N_s.*
3) $N_s \cap F \neq \emptyset$ *and* $|N_s \cap V_1| = 2$.

We use Lemma 14 to extend the process of cutting along noncontractible tight nooses such that we obtain a planar graph with a small number of disjoint cut-nooses of small lengths. Let $g \leq \mathbf{eg}(\Sigma)$ be the number of iterations needed to cut along shortest noncontractible nooses such that they turn a Σ-embedded graph G into a planar graph G'. However, these cut-nooses may not be disjoint. In our dynamic programming approach we need pairwise disjoint cut-nooses. Thus, whenever we cut along a noose, we manipulate the cut-nooses found so far. After g iterations, we obtain the *set of cut-nooses* $\mathfrak{N}$ that is a set of disjoint cut-nooses bounding empty open disks in the embedding of G'. Let $L(\mathfrak{N})$ be the *length* of $\mathfrak{N}$ as the sum over the lengths of all cut-nooses in $\mathfrak{N}$. The proof of the following proposition is moved to the appendix.

Proposition 15. *It is possible to find, in polynomial time, a set of cut-nooses $\mathfrak{N}$ that contains at most $2g$ disjoint cut-nooses. Furthermore $L(\mathfrak{N})$ is at most $2g\,\mathbf{rep}(G)$.*

We extend the definition of relaxed Hamiltonian sets from graphs embedded on a torus to graphs embedded on higher genus, i.e. from two cut-nooses N_X and N_Y to the set of cut-nooses $\mathfrak{N}$. For each vertex v in the vertex set $V(G)$ of graph G we define the vertex set D_v that contains all duplicated vertices $v_1, \ldots, v_f$ of v in $\mathfrak{N}$ along with v. Set $\mathfrak{D} = \bigcup_{v \in V(G)} D_v$.

Definition 16. A set of disjoint paths $\mathbf{P}$ in G' is *relaxed Hamiltonian* if:
(P1) Every path has its endpoints in $\mathfrak{N}$.
(P2) If a vertex $v_i \in D_v \in \mathfrak{D}$ is an endpoint of path P, then there is one $v_j \in D_v$ that is also an endpoint of a path $P' \neq P$. All $v_h \in D_v \setminus \{v_i, v_j\}$ do not belong to any path.
(P3) $v_i \in D_v$ is an inner path vertex if and only if all $v_h \in D_v \setminus \{v_i\}$ are not in any path.
(P4) Every vertex of the residual part of G' is in some path.

Similar to torus-embedded graphs, we order the vertices of $\mathfrak{N}$ for later encoding in a counterclockwise order $\pi_{\mathbf{L}}$ depending on the fixed paths between the cut-nooses of $\mathfrak{N}$:

Lemma 17. *Let G' be the planar graph after cutting along $g \leq \mathbf{eg}(\Sigma)$ tight nooses of G along with its set of disjoint cut-nooses $\mathfrak{N}$. The number of different equivalence classes of relaxed Hamiltonian sets in G' is $2^{O(g \cdot (\log g + \mathbf{rep}(G)))}$.*

Proof. We create one cut-noose out of all the cut-nooses of $\mathfrak{N}$ by using "tunnels" as in the proof of lemma 2. But the difficulty here is that the cut-nooses are connected by a relaxed Hamiltonian set in an arbitrary way. We use a tree structure in order to cut the sphere along that structure. Given such a tree structure, we create tunnels in order to connect open disks and to merge them to one disk. Let $\mathbf{C}$ be a candidate of the relaxed Hamiltonian set. Define graph H such that each cut-noose $N_i \in \mathfrak{N}$ in G' corresponds to a vertex i in $V(H)$. Two vertices i, j of H are adjacent if there is a path between vertices of N_i and N_j in $\mathbf{C}$. Let F be a spanning forest of H. For every pair of adjacent vertices i, j in F fix a path in G' between two arbitrary vertices $v^i_x \in N_i$ and $v^j_y \in N_j$. Walk along a tree by starting and ending in a node r and visiting all nodes by always visiting the next adjacent neighbor in counterclockwise order. A node is visited as many times as many neighbors it has. In this way we create tunnels in G' by ordering the vertices of the cut-nooses: Starting with an ordered list $\mathbf{L} = \{\emptyset\}$ and one cut-noose N_i and one endpoint $v^i_x \in N_i$ of a fixed path. Take in counterclockwise order the vertices of N_i into $\mathbf{L}$ that are between v^i_x and the last vertex before the next endpoint $v^i_y \in N_i$ connected to $v^j_z \in N_j$ in the fixed path $P_{i,j}$. Concatenate to $\mathbf{L}$ in counterclockwise order the vertices of N_j after v^j_z until the last vertex before the next endpoint of a fixed path. Repeat the concatenation until one reaches again v^i_x. Whenever an endpoint is visited for the second time concatenate it to $\mathbf{L}$, too. Create an ordered list $\mathbf{L}_C$ for every component C in $\mathbf{C}$ and concatenate $\mathbf{L}_C$ to $\mathbf{L}$. The order of $\mathbf{L}$ is then $\pi_{\mathbf{L}}$. (See Figure 7 in the Appendix for an example.) Consider all $\leq n^{n-2}$ possible spanning trees on n vertices ([1]), and so $\leq 2^n n^{n-2}$ possible spanning forrests. There is a spanning forrest over the $2k$ cut-nooses for each candidate $\mathcal{P}$. With $2g$ cut-nooses of $\mathfrak{N}$ each of length at most $2g\,\mathbf{rep}(G)$ there are $O((2g\,\mathbf{rep}(G))^2)$ possible fixed path between each two cut-nooses. Then we obtain a rough upper bound of $O(2g\,\mathbf{rep}(G))^{4g})$ on the number possible fixed path between the cut-nooses in a given tree-structure. We obtain $O(2^{2g}(2g)^{2g-2}(2g\,\mathbf{rep}(G))^{4g})$ possibilities for above concatenation and tunneling of $\mathfrak{N}$. Again we argue $\mathbf{C}$ has a Catalan structure when tunneling the cut-nooses in this way. Due to (P2) in definition 16, there are $O(2^{2g\,\mathbf{rep}(G)})$ many sets of paths with endpoints in the cut-nooses of $\mathfrak{N}$ non-crossing the fixed paths.The number of relaxed Hamiltonian sets is due to (P3) $O(2^{3g\,\mathbf{rep}(G)})$. □

Given the order $\pi_{\mathbf{L}}$ of the vertices $\mathfrak{N}$ in the encoding of candidate $\mathbf{C}$. As in the previous section, we preprocess the graph G' and encode the vertices of $\mathfrak{N}$ with the base values. We extend the dynamic porgramming approach by analysing how the tight noose O_e can intersect several cut-nooses. The proofs of the next two statements are moved to the Appendix.

Lemma 18. *Let G' be the planar graph after cutting along $g \leq \mathbf{eg}(\Sigma)$ shortest noncontractible nooses of G. For a given sc-decomposition $\langle T, \mu, \pi \rangle$ of G' of width ℓ and a candidate $\mathbf{C}$ the* Relaxed Hamiltonian Set *problem on G' can be solved in time $2^{O(g^2 \log \ell)} \cdot 2^{O(\ell)} \cdot n^{O(1)}$.*

Lemma 19. *Let G be a Σ-embedded graph with n vertices and G' the planar graph obtained after cutting along $g \leq \mathbf{eg}(\Sigma)$ tight nooses. Then, $\mathbf{bw}(G') \leq \sqrt{4.5} \cdot \sqrt{n} + 2g$.*

Lemmata 11, 17, 18 and 19 imply the following:

Theorem 20. *Given a Σ-embedded graph G on n vertices and $g \leq \mathbf{eg}(\Sigma)$. The* Hamiltonian cycle *problem on G can be solved in time $n^{O(g^2)} \cdot 2^{O(g\sqrt{g \cdot n})}$.*

Our dynamic programming technique can be used to design faster parameterized algorithms as well. For example, the parameterized p-Cycle on Σ-embedded Graphs problem asks for a given Σ-embedded graph G, to check for the existence of a cycle of length at least a parameter p. First, our technique can be used to find the longest cycle of G with $g \leq \mathbf{eg}(\Sigma)$ in time $n^{O(g^2)} \cdot 2^{O(g\sqrt{g \cdot n})}$. (On torus -embedded graphs this can be done in time $O(2^{17.957\sqrt{n}} n^3)$.) By combining this running time with bidimensionality arguments from [3] we arrive at a time $2^{O(g^2 \log p)} \cdot 2^{O(g\sqrt{g \cdot p})} \cdot n^{O(1)}$ algorithm solving the parameterized p-Cycle on Σ-embedded Graphs.

5 Conclusive Remarks

In this paper we have introduced a new approach for solving non-local problems on graphs of bounded genus. With some sophisticated modifications, this generic approach can be used to design time $2^{O(\sqrt{n})}$ algorithms for many other problems including Σ-EMBEDDED GRAPH TSP (TSP with the shortest path metric of a Σ-embedded graph as the distance metric for TSP), MAX LEAF TREE, and STEINER TREE, among others. Clearly, the ultimate step in this line of research is to prove the existence of time $2^{O(\sqrt{n})}$ algorithms for non-local problems on any graph class that is closed under taking of minors. Recently, we were able to complete a proof of such a general result, using results from the Graph Minor series. One of the steps of our proof is strongly based on the results and the ideas of this paper.

References

[1] A. CAYLEY, *A theorem on trees*, Quart J. Pure Appl. Math., 23 (1889), pp. 26–28.

[2] V. G. DEĬNEKO, B. KLINZ, AND G. J. WOEGINGER, *Exact algorithms for the Hamiltonian cycle problem in planar graphs*, Operations Research Letters, (2006), p. to appear.

[3] 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.

[4] E. D. DEMAINE AND M. HAJIAGHAYI, *Bidimensionality: new connections between fpt algorithms and ptass*, in Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2005), New York, 2005, ACM-SIAM, pp. 590–601.

[5] F. DORN, E. PENNINKX, H. BODLAENDER, AND F. V. FOMIN, *Efficient exact algorithms on planar graphs: Exploiting sphere cut branch decompositions*, in Proceedings of the 13th Annual European Symposium on Algorithms (ESA 2005), vol. 3669 of LNCS, Springer, Berlin, 2005, pp. 95–106.

[6] F. V. FOMIN AND D. M. THILIKOS, *Fast parameterized algorithms for graphs on surfaces: Linear kernel and exponential speed-up*, in Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP 2004), vol. 3142 of LNCS, Berlin, 2004, Springer, pp. 581–592.

[7] ———, *New upper bounds on the decomposability of planar graphs*, Journal of Graph Theory, 51 (2006), pp. 53–81.

[8] J. R. GILBERT, J. P. HUTCHINSON, AND R. E. TARJAN, *A separator theorem for graphs of bounded genus*, Journal of Algorithms, 5 (1984), pp. 391–407.

[9] Q.-P. GU AND H. TAMAKI, *Optimal branch-decomposition of planar graphs in $O(n^3)$ time*, in Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP 2005), vol. 3580 of LNCS, Springer, Berlin, 2005, pp. 373–384.

[10] R. J. LIPTON AND R. E. TARJAN, *A separator theorem for planar graphs*, SIAM J. Appl. Math., 36 (1979), pp. 177–189.

[11] ———, *Applications of a planar separator theorem*, SIAM J. Comput., 9 (1980), pp. 615–627.

[12] B. MOHAR AND C. THOMASSEN, *Graphs on surfaces*, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, 2001.

[13] N. ROBERTSON AND P. D. SEYMOUR, *Graph minors. VII. Disjoint paths on a surface*, J. Combin. Theory Ser. B, 45 (1988), pp. 212–254.

[14] ———, *Graph minors. XI. Circuits on a surface*, J. Combin. Theory Ser. B, 60 (1994), pp. 72–106.

[15] P. D. SEYMOUR AND R. THOMAS, *Call routing and the ratcatcher*, Combinatorica, 14 (1994), pp. 217–241.

[16] C. THOMASSEN, *Embeddings of graphs with no short noncontractible cycles*, J. Combin. Theory Ser. B, 48 (1990), pp. 155–177.

[17] G. WOEGINGER, *Exact algorithms for NP-hard problems: A survey*, in Combinatorial Optimization - Eureka, you shrink!, vol. 2570 of LNCS, Springer-Verlag, Berlin, 2003, pp. 185–207.

A Appendix: Proof of Lemma 7

Preprocess G'. In a preprocessing step we delete all vertices of $N_X \cup N_Y$ from G' which do not belong to any path of $\mathbf{C}$. The other vertices in $N_X \cup N_Y$ are encoded by *base values* $\{[,], S, \square\}$. This *base encoding* depends on the order π_{XY} and is fixed throughout the phase of dynamic programming. Say in the tuple (s,t) of $\mathbf{T}$, s is marking the first vertex in π_{XY} and t the last vertex. We encode both s and t by 'S'. For every other tuple (s_i, t_i) of $\mathbf{T}$ we encode s_i by '[' and t_i by ']' where $s_i < t_i$ in π_{XY}. The additional value 'S' is important for a consistent dynamic programming. It determines the "tunnel" created by the path with endpoints s and t. As described in the proof of Lemma 6 the cut-nooses N_X, N_Y and the tunnel border the outer face that enables the encoding. The vertices of $\mathbf{I}$ simply are encoded by base value '$\square$'.

Constructing branch decomposition. We use Proposition 3 to obtain a sc-decomposition $\langle T, \mu, \pi\rangle$ of G' of optimum width ℓ. For dynamic programming it is convenient to root T by choosing arbitrarily an edge e and subdividing e by inserting a new node s. Let e', e'' be the new edges then we set $\mathbf{mid}(e') = \mathbf{mid}(e)$ and $\mathbf{mid}(e'') = \mathbf{mid}(e)$. Create a new node *root* r and connect it to s and set $\mathbf{mid}(\{r,s\}) = \emptyset$. Each node v of T now has one adjacent edge on the path from v to r, called *parent edge* e_P, and two adjacent edges towards the leaves, called *left child* e_L and *right child* e_R. For every edge e of T, we call the subtree towards the leaves the *lower part* and the rest the *residual part* concerning 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 tight noose in $\mathbb{S}_0$. Recall that tight noose O_e partitions $\mathbb{S}_0$ into two discs, one of which, Δ_e, contains G_e.

In the following, we often do not distinguish between $\mathbf{mid}(e)$ and $O_e \cap V(G)$. We start at the leaves of T and work 'bottom-up' processing the subgraphs rooted at the edges up to the root edge. All paths of $\mathbf{P} \cap G_e$ start and end in O_e and $G_e \cap (N_X \cup N_Y)$. For a dynamic programming step we need the information on how a tight noose O_e and $N_X \cup N_Y$ intersect and which parts of $N_X \cup N_Y$ are a subset of the subgraph G_e. Define the vertex set $\mathcal{X} = (G_e \setminus O_e) \cap (N_X \cup N_Y)$.

In the dynamic programming approach we differentiate three phases. In Phase 1 no vertex of $N_X \cup N_Y$ is contained in disk Δ_e bounded by O_e, thus $\mathcal{X} = \emptyset$. Note that in this phase $(N_X \cup N_Y) \cap O_e$ is not necessary empty because O_e may touch N_X,N_Y in common vertices. Hence all paths must start and end in O_e. At some step of dynamic programming we arrive at Phase 2: $\mathcal{X} \neq \emptyset$ but neither $V(N_X) \subseteq \mathcal{X}$, nor $V(N_Y) \subseteq \mathcal{X}$. This is when we connect the loose paths—i.e. paths with endpoints in $O_e \setminus (N_X \cup N_Y)$—to their predestinated endpoints in N_X and N_Y. Finally, we arrive at the situation, Phase 3, when either $V(N_X) \subseteq \mathcal{X} \vee V(N_Y) \subseteq \mathcal{X}$, or $V(N_X) \subseteq \mathcal{X} \wedge V(N_Y) \subseteq \mathcal{X}$. Here we take care that the residual paths with one determined endpoint in N_X and N_Y are connected in the corresponding way.

Proposition 21. *Phase 1: Given a sc-decomposition $\langle T, \mu, \pi\rangle$ of G' of width ℓ and a candidate $\mathbf{C} = (\mathbf{T}, \mathbf{I})$. The phase of dynamic programming with $\mathcal{X} = \emptyset$ takes time $O(2^{3.292\ell})$.*

Every vertex of the subgraph G_e below O_e is part of one of the vertex-disjoint paths $P_1, \ldots, P_q$ with endpoints in O_e. The state of dynamic programming is specified by an ordered ℓ-tuple $\vec{t_e} := (v_1, \ldots, v_\ell)$ with the variables $v_1, \ldots, v_\ell$ corresponding to the vertices of $O_e \cap V(G)$. The variables have one of the four values: 0, $1_[$, $1_]$, 2. For every state, we compute a Boolean value $B_e(v_1, \ldots, v_\ell)$ that is TRUE if and only if $P_1, \ldots, P_q$ in G_e have the following properties: *(A1) Every vertex of $V(G_e) \setminus O_e$ is contained in one of the paths P_i, $1 \leq i \leq q$.*
(A2) Every P_i has both its endpoints in $O_e \cap V(G)$;

Let P be a path in G_e. Since none of the paths in G_e cross, we argue again by making use of the *Catalan structure*. We scan the vertices of $O_e \cap V(G)$ according to the ordering π and mark with '$1_[$' the first and with '$1_]$' the last vertex of P. If a vertex of $O_e \cap V(G)$ is adjacent to two edges P we mark it with '2'. If a vertex is not contained in any path we mark it with '0'.

Processing middle sets. The first step in processing the middle sets is to initialize the leaves with values $(0,0)$, $(1_[, 1_])$. 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. Let O_L, O_R, and O_P be the tight nooses corresponding to edges e_L, e_R and e_P. Due to the definition of branch decompositions, every vertex must appear in at least two of the three middle sets and we can define the following partition of the set $(O_L \cup O_R \cup O_P) \cap V(G)$ into sets $I := O_L \cap O_R \cap V(G)$ and $D := O_P \cap V(G) \setminus I$ (I stands for 'Intersection' and D for 'symmetric Difference'). The disc Δ_P bounded by O_P and including the subgraph rooted at e_P contains the union of the discs Δ_L and Δ_R bounded by O_L

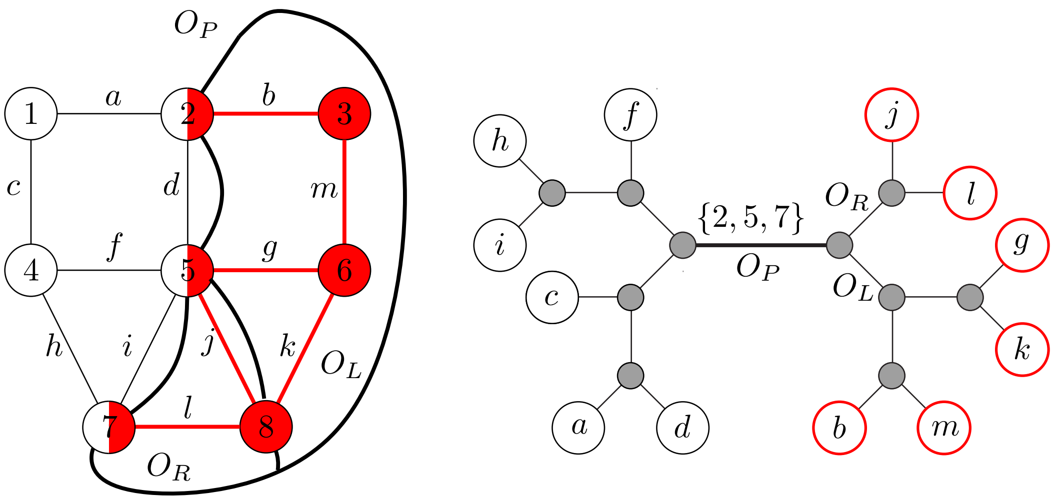

Figure 3: **Processing middle sets.** The diagrams show a graph together with its sc-decomposition. The tight nooses O_P, O_L and O_R are emphasized. O_P touches the graph in vertices $2, 5, 7$, O_L in $2, 5, 8$ and O_R in $5, 7, 8$. Vertex 5 is the only portal vertex of $O_L \cap O_R \cap O_P \cap V(G)$.

and O_R and including the subgraphs rooted at e_L and e_R. Thus $|O_L \cap O_R \cap O_P \cap V(G)| \leq 2$. The vertices of $O_L \cap O_R \cap O_P \cap V(G)$ are called *portal vertices*. See Figure 3 for an illustration.

We compute all valid assignments to the variables $\vec{t}_P = (v_1, v_2, \ldots, v_p)$ corresponding to the vertices $\mathrm{mid}(e_P)$ from all possible valid assignments to the variables of $\vec{t}_L$ and $\vec{t}_R$. For a symbol $x \in \{0, 1_{[}, 1_{]}, 2\}$ we denote by $|x|$ its "numerical" part. Thus, for example $|1_{[}| = 1$. We say that an assignment c_P is *formed* by assignments c_L and c_R if for every vertex $v \in (O_L \cup O_R \cup O_P) \cap V(G)$:

1. $\underline{v \in D}$: $c_P(v) = c_L(v)$ if $v \in O_L \cap V(G)$, or $c_P(v) = c_R(v)$ otherwise.
2. $\underline{v \in I \setminus O_P}$: $(|c_L(v)| + |c_R(v)|) = 2$.
3. $\underline{v \text{ portal vertex}}$: $|c_P(v)| = |c_L(v)| + |c_R(v)| \leq 2$.

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 do not form cycles.

Running time. 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 for simplicity assume there are no portal vertices. This can only occur if $|I| = |D \cap O_L| = |D \cap O_R| = \frac{\ell}{2}$. We give an expression for $Q(\ell, m)$: the number of ℓ-tuples over ℓ vertices where the $\{0, 1_{[}, 1_{]}, 2\}$ assignments for vertices from I is fixed and contains m $1_{[}$'s and $1_{]}$'s. The only freedom we have is thus in the $\ell/2$ vertices in $D \cap O_L$ and $D \cap O_R$, respectively:

$$Q(\ell, m) \approx \sum_{i=0}^{\frac{\ell}{2}} \binom{\frac{\ell}{2}}{i} 2^{\frac{\ell}{2}-i} 2^{i+m} = 2^{\ell+m} \tag{1}$$

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. As we are interested in exponential behaviour for large values of ℓ we ignore that $i + m$ is even.

We can count the total cost of forming an ℓ-tuple from O_P by summing over i: the number of $1_{[}$'s and $1_{]}$'s in the assignment for I:

$$C(\ell) = \sum_{i=0}^{\frac{\ell}{2}} \binom{\frac{\ell}{2}}{i} 2^{\frac{\ell}{2}-i} Q(\ell, i)^2 \approx (4\sqrt{6})^{\ell} \approx 2^{3.292\ell} \tag{2}$$

There is one restriction to the encoding of the vertices in $O_e \cap (N_X \cup N_Y)$: a vertex with base value '[','] ' or 'S' cannot be assigned with '2' at any stage.

Proposition 22. <u>*Phase 2:*</u> *The phase of dynamic programming with $\mathcal{X} \neq \emptyset$ and $V(N_X) \not\subseteq \mathcal{X}$ and $V(N_Y) \not\subseteq \mathcal{X}$ takes time $O(2^{6.360\ell})$.*

Let us remind that $\mathcal{X} = (G_e \setminus O_e) \cap (N_X \cup N_Y)$. The difficulty of the second phase lies in keeping track of the base encoding of $\mathcal{X}$. Thus, we do not want to memorize explicitly with which endpoint in $\mathcal{X}$ a vertex of O_e forms a path. We apply again the Catalan structures. The key to it is first that the vertices of O_e inherit the base values of the sets $\mathbf{T}$ and $\mathbf{I}$—the sets of the definition of the relaxed Hamiltonian set problem. That is, if one vertex of O_e is paired with a vertex assigned by '[' it must be paired in $\overline{G_e}$ with a vertex with value ']'. Second, we observe that the cut-nooses $N_X \cup N_Y$ and the tight noose O_e intersect in a characteristic way: we show that we obtain a structure that allows us to encode paths in an easy way. In other words we make use of the structure of the subgraph G_e bordered by $N_X \cup N_Y$ and O_e for synchronizing the encoding of $\mathbf{T}$ and $\mathbf{I}$ with the encoding of O_e. Thus, we need some more definitions. A *partial noose* is a proper connected subset of a tight noose and cut-noose, respectively. A *partial component* of a graph is embedded on an open disk bounded by partial nooses. The vertices in the intersection of two partial components are called *connectors*. In fact, G_e can be partitioned into several partial components with no connector in three components. Each component is bordered by partial nooses of $N_X \cup N_Y$ and O_e.

Proposition 23. *The subgraph G_e is the union of partial components $C_1, \dots, C_q$ $(q \geq 1)$ such that for every i*

$C_i \cap (\bigcup_{r=1, r\neq i}^{q} C_r) \subseteq O_e \cap (N_X \cup N_Y)$. *Furthermore, for every i, j, h, $C_i \cap C_j \cap C_h = \emptyset$.*

Proof. Recall that by definition a tight noose intersects a region exactly once. Hence O_e intersects at most once the empty disks Δ_X and Δ_Y that are bounded by N_X and N_Y. In this case, O_e and $(N_X \cup N_Y)$ can intersect in vertices or as well twice the arc between to successive vertices in N_X and N_Y, respectively. That is due to the fact that N_X and N_Y are cut-nooses and hence may have several arcs in one face. In contrast, O_e and $(N_X \cup N_Y)$ can touch arbitrarily often, but only in vertices. In Phase 2, $\Delta_e \cap (\Delta_X \cup \Delta_Y) \neq \emptyset$. Hence, if one removes $\Delta_X \cup \Delta_Y$ then Δ_e is partitioned into several disks $\Delta_1, \dots, \Delta_q$ each bordered by the union of some partial nooses bounded by vertices v of $V(O_e) \cap V(N_X \cup N_Y)$ or some points μ of the crossing of the arcs between two successive vertices of O_e and $N_X \cup N_Y$. Since N_X and N_Y do not intersect, we have that v and μ are the endpoints of at most four partial nooses. Hence, v is neighboring at most two partial components and μ one. □

See the left diagram of Figure 4 for an example.

Proposition 24. *Each C_i is bordered by sets of partial nooses A_i and B_i with $A_i \subset (N_X \cup N_Y) \setminus O_e$ and $B_i \subset O_e$ with $\bigcup_{i=1}^{k} A_i \cup B_i = G_e \cap ((N_X \cup N_Y) \cup O_e)$ such that one of the following hold:*

1. *$|A_i| = |B_i| = 1$ with $A_i \subset N_X$ or $A_i \subset N_Y$,*
2. *$|A_i| = |B_i| = 2$ with $A_i \subset N_X$ and $A_i \subset N_Y$.*

There is at most one partial component C_i with property 2.

Proof. Assume, there is a component C_i with two partial noose $P_i^1, P_i^2 \in A_i \cap N_X$. Two of the points bordering P_i^1 and P_i^2 bound the partial noose of O_e that intersects Δ_X. Then both other points that border P_i^1 and P_i^2 are each bordering two partial nooses of O_e intersecting G_e. Thus, there is no possible configuration in which P_i^1 and P_i^2 bound the same component. Assume two components C_i and C_j with $|A_i| = |A_j| = 2$. Then there are four partial nooses of $B_i \subset O_e$ that must be connectable to a tight noose O_e. That is not possible without crossing Δ_X and Δ_Y more than once. □

See the right diagram of Figure 4 for an illustration.

In contrast to the first phase we encode the vertices for each component C_i of Proposition 23 separately. The connectors, the vertices that are in two components are encoded twice. A restriction to the encoding of the vertices in $O_e \cap (N_X \cup N_Y)$ is the consideration of the base encoding, for example a vertex with base value '[' or ']' cannot be assigned with '2' at any stage.

We introduce new values for indicating a connection to vertices of $\mathcal{X} = (G_e \setminus O_e) \cap (N_X \cup N_Y)$. Proposition 24 guarantees that we can differentiate between three types of partial components C_i. The ones without any vertex in $\mathcal{X}$ and the two that have properties 1 and 2. For all three cases every vertex of $V(C_i) \setminus O_e$ is contained in one path.

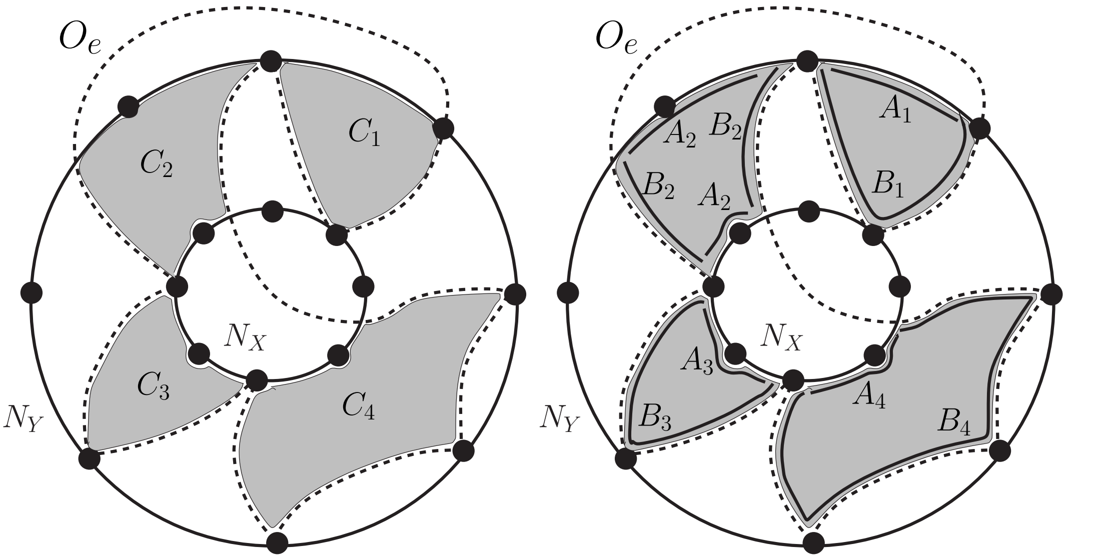

Figure 4: **Partial components and partial nooses.** The left diagram illustrates how O_e and N_X,N_Y form the partial components $C_1, \ldots, C_4$. Observe that $C_1, \ldots, C_4$ only intersect in vertices whereas O_e and N_X,N_Y do not have to. In the right diagram, each partial component is bounded by partial nooses. Only component C_2 has $|A_2| = |B_2| = 2$.

1. $C_i \cap \mathcal{X} = \emptyset$.
   - Every path has both endpoints in $V(C_i) \cap O_e$.
   - Every vertex of $V(C_i) \cap (N_X \cup N_Y)$ with base value '[' or ']' is not to be an inner vertex of a path.
   - We use the same encoding as in phase 1.
2. $C_i \cap \mathcal{X} \neq \emptyset$ and $|A_i| = 1$.
   - Every path has both endpoints in $V(C_i) \cap (O_e \cup \mathcal{X})$.
   - A vertex of $V(C_i) \cap O_e$ with other endpoint w in $\mathcal{X}$ is encoded with the base value of w, '[' or ']'. Since $|A_i| = 1$ and the Catalan structure is retained for the border vertices of C_i, it is possible to reconstruct the order π_{XY} in which the other endpoints in $\mathcal{X}$ are. The base value '$\Box$' does not appear. We introduce 'S_X' and 'S_Y' for marking the connection to vertices in N_X and N_Y that have the base values 'S'. 'S_X' and 'S_Y' appear at most once.
   - For paths with both endpoints in $V(C_i) \cap O_e$ we use the same encoding as in phase 1.
3. $C_i \cap \mathcal{X} \neq \emptyset$ and $|A_i| = 2$.
   - Every path has both endpoints in $V(C_i) \cap (O_e \cup \mathcal{X})$.
   - As in the latter case we use the base encoding to encode the vertices of $V(C_i) \cap O_e$, too. Additionally, we introduce values '$]_L$','$[_L$' to mark each of the last two vertices in order π that are endpoint of a path with other endpoint in N_X and N_Y, respectively. These values are used only once in an unique C_i and hence do not play any role for the running time. Note that if there is no vertex encoded with '$]_L$','$[_L$', this means that vertices encoded by ']','[' are only connected to N_X. In contrast to, if there is only one vertex encoded with '$]_L$','$[_L$', and it is after N_Y then all vertices encoded by ']','[' are only connected to N_Y.

See Figure 5 for an example on the usage of encoding '$]_L$','$[_L$'.

Additional special cases.

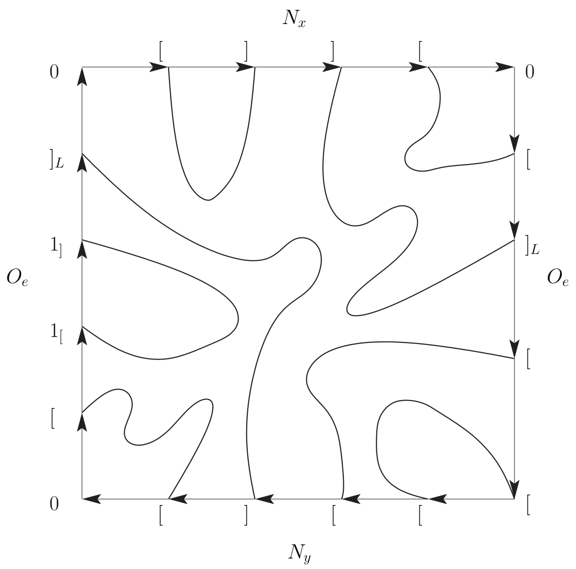

Figure 5: **Usage of '$]_L$'and '$[_L$'.** The diagram shows the partial component that is bounded by four partial nooses. The vertices are clockwise ordered beginning in the upper left corner. '$]_L$' on the right partial noose of O_e marks the last vertex of O_e connected to N_X. '$]_L$' on the left partial noose of O_e marks the last vertex connected to N_Y.

- **Base value 'S'.**

  We leave it to the reader to consider the special cases that occur with base value 'S'. Recall that 'S' marks the fixed path $P_{i,j}$ and the beginning and the end of order π_{XY}. It is easy to see that the encoding determines wether a vertex of $V(C_i) \cap O_e$ is connected to a vertex before or after an endpoint of $P_{i,j}$. For example, suppose both endpoints of $P_{i,j}$ are in $\mathcal{X} \cup O_e$ and $P_{i,j} \subset C_i$. Then $P_{i,j}$ separates C_i into two parts which cannot be connected by a path, since neither $N_X \subset C_i$ nor $N_Y \subset C_i$.
- **Right encoding of connector.** Let c be a connector between two partial components C_i,C_j. The two values of c must combine to the correct base value. If base value of c is not '$\square$', at least one of the two values in C_i or C_j must be '0' and none is '2'.
- **Path through several components.** For every component C_i, a vertex v of the tight noose $O_e \cap C_i$ can be paired to a connector with base value '$\square$'. Hence, v can be an endpoint of a path with other endpoint in $\mathcal{X}$ in another component C_j. In this case assign v with the corresponding base value.

Processing middle sets. The middle sets are processed exactly as described in the first phase. In the first step every pair of states of two child edges e_L and e_R are updated to a state of the parent edge e_P. For a symbol of $\{[,], S_X, S_Y,]_L, [_L\}$ the numerical value is 1 and we form the vertex assignments as above. I.e., base values are treated exactly as '$1_[$' or '$1_]$'. With the only restriction if the assignments c_L and c_R of a vertex v both are base values. The base value in O_L must fit to the base value in O_R, i.e., if wlog $c_L(v)$ has value '[' then $c_R(v)$ must have ']', if wlog $c_L(v) = S_X$ then $c_R(v) = S_Y$. In the second step, we not only check forbidden cycles, but consistency of the encoding regarding to the base values. With the help of an auxiliary graph consisting of G_L and G_R together with the partial components, we check the following:

1. The base values of c_L and c_R are connected respecting π_{XY}. For a vertex v of I encoded by '[' and ']' in O_L and O_R, respectively, it must hold that the endpoint with base value '[' must be in order π_{XY} before the endpoint with ']'.
2. The vertices of a partial component C_i of O_P that are paired to a vertex of $\mathcal{X} \cap C_i$ are assigned with

the correct value of $\{[,], S_X, S_Y,]_L, [_L\}$. Note that new connector vertices are generated, which must be encoded component-wise.

Running time. When counting the number of states we omit values $\{S_X, S_Y,]_L, [_L\}$ since they are assigned to at most two vertices of O_L and O_R. Each connector is assigned with two values. The number of connectors can be in order size of a O_L and O_R, respectively. The values of the vertices in the D-set are transferred in time depending on the number of values $\{0, 1_{[}, 1_{]}, 2, [,]\}$ and the number of valid encoding of the connectors. There are 25 ways of encoding a connector correctly. Apparently, if a vertex is a connector in O_L then it is not in O_R. To simplify matters, assume that $V(O_L)$ are connectors and $V(O_R)$ are not. Then the update time for D is $O(25^{|D\cap O_L|}6^{|D\cap O_R|})$. There are 45 possible assignments for vertices in I to sum up to two. Thus, updating time for the I-set is $O(45^{|I|})$. With analogous calcultions as before we get an overall running time $6^{0.5\ell}25^{0.5\ell}45^{0.5\ell} \approx 2^{6.36\ell}$.

Proposition 25. *<u>Phase 3:</u> The phase of dynamic programming with $\mathcal{X} \neq \emptyset$ and either $V(N_X) \subseteq \mathcal{X}$ or $V(N_Y) \subseteq \mathcal{X}$ or both takes time $O(2^{6.360\ell})$.*

In the last phase at least one of both cut-nooses N_X,N_Y are subsets of $V(G_e) \setminus O_e$. The difficulty is apparently to encode the endpoint of a path with one endpoint in such N_X, N_Y. We consider two cases:

1. The fixed path $P_{i,j}$ is crossing O_e.
   If both N_X and N_Y are in $V(G_e) \setminus O_e$ we assume the first vertex v assigned by 'S_X' in π and the other w by 'S_Y'. Use encoding with '$[_L$','$]_L$' to mark the last vertex in the partial noose (v, w) connected to N_X and the last vertex in the partial noose (w, v) connected to N_Y. If wlog N_X crosses O_e we find a partial component C_i including N_Y. We mark two vertices with '$[_L$','$]_L$ in the same way , no matter if there is one or two of 'S_X','S_Y' in $V(C_i) \cap O_e$.
2. $P_{i,j} \subset G_e \setminus O_e$.
   One vertex in O_e is marked '$[_{X,L}$' or '$]_{X,L}$' to be the last vertex in π connected to N_X and one vertex by '$[_{Y,L}$' or '$]_{Y,L}$' to be the last connected to N_X. If one of N_X,N_Y cross O_e we again find a partial component C_i can be encoded in that way.

Since the new values '$[_*$','$]_*$' appear only twice per middle set, they do not affect the running time. The algorithm works the same as in phase two, considering the two latter cases.

With more complicated encodings and analysis we are able to improve the running time of phase 2 and 3 :

Proposition 26. *<u>Phase 2 and 3:</u> takes time $O(2^{5.433\ell})$.*

We omit the details here.

B Appendix: Proof of Lemma 14

Recall that N_s is tight i.e. it can be seen as a cycle in the radial graph R_G. This directly implies that if $|N_s \cap V_1| = 1$, then $N_s \cap F = \emptyset$.

Suppose now that $N_s \cap V_1 = \{v_i, v_j\}$ and $N_s \cap F = \emptyset$. Suppose also that there is no face as the one required in 2.2. Then the cycle C_s of R_G corresponding to N_s is partitioned into two paths P_2 and P_3, each with ends v_i and v_j and of length > 2. We use the notation v_F for the vertex of R_G corresponding to the face F. Let also $P_1 = (v_i, v_F, v_j)$ and notice that the two cycles of R_G defined by $P_1 \cup P_3$ and $P_1 \cup P_2$ have length smaller than $P_2 \cup P_3 = C_s$ and therefore they are contractibe. By Proposition 13, N_s is contractible—a contradiction.

For the sake of contradiction, we assume that $|N_s \cap V_1| \geq 3$. Assume N_s intersects V_1 in vertices $I = v_1, \ldots, v_k$, $k \geq 3$, and with at most two vertices connected by the part of the noose of N_s that intersects F. In the radial graph R_G of G, N_s corresponds to the shortest noncontractible cycle C_s. In R_G each vertex of V_1 is a neighbour of the vertex v_F.

We consider the two cases: $N_s \cap F \neq \emptyset$. That is, there exists a path $\{v_i, v_F, v_j\} \subset C_s$ in R_G with $v_i, v_j \in I$. Let v_h be another vertex in $I = V_1 \cap C_s$. Consider the three paths in R_G connecting v_F and v_h, namely $P_1 = (v_F, v_i, \ldots, v_h)$, $P_2 = (v_F, v_j, \ldots, v_h)$, and $P_3 = (v_F, v_h)$. Notice also that the two cycles of R_G defined by $P_1 \cup P_3$ and $P_2 \cup P_3$ have length smaller than $P_1 \cup P_2 = C_s$ and therefore they are contractibe which is a contradiction to Proposition 13.

In case $N_s \cap F = \emptyset$, we choose $v_i, v_j, v_h \in I$ arbitrarily and the arguments of the previous case imply that the path $P_1 \cup P_2$ is contractible. We define now the paths $Q_1 = (v_i, \ldots, v_h, \ldots, v_j)$, $Q_2 = (v_i, v_F, v_j)$, and $Q_3 = (v_i, \ldots, v_j)$ between the vertices v_i and v_j. As $Q_1 \cup Q_2 = P_1 \cup P_2$, the cycle $Q_1 \cup Q_2$ of R_G is contractible. The same holds for the cycle $Q_2 \cup Q_3$ as its length is less than the length of $Q_1 \cup Q_3 = C_s$. Then again Proposition 13 implies that $Q_1 \cup Q_3 = C_s$ is contractible, a contradiction.

C Appendix: Proof of Proposition 15

Let $\mathfrak{N}_i$ be the set of disjoint cut-nooses after i cuts. Consider the cases of Lemma 14 of how a shortest noncontractible (tight) noose N_s intersects a cut-noose of $\mathfrak{N}_i$.

• Suppose N_s intersects with the empty disk Δ_j bounded by $N_j \in \mathfrak{N}_i$. Let $P_1 \cup P_2 = N_j$ be the two partial nooses of N_j determined by the intersection of N_j and N_s. When we cut along N_s, we replace N_s by the contractible cut-nooses N_X and N_Y. We replace $N_X \cap \Delta_j$ by P_1 and $N_Y \cap \Delta_j$ by P_2. In $\mathfrak{N}_i$ we substitute N_j by N_X and N_Y. Note that N_s can intersect with several disjoint cut-nooses of $\mathfrak{N}_i$ in this way. See upper diagrams Figure 6 for an example.

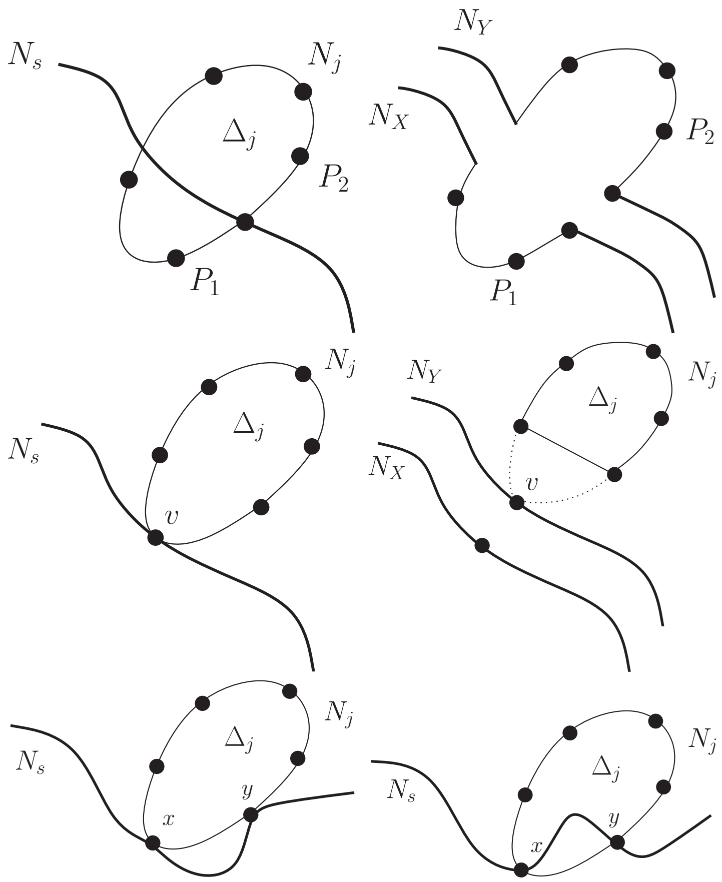

Figure 6: **Making cut-nooses disjoint.** The upper diagrams show how a noncontractible tight noose N_s partitions N_j into two partial nooses P_1 and P_2. $N_X \cup P_1$ and $N_Y \cup P_2$ form new cut-nooses. The middle diagrams show how N_s touches N_j in only one vertex v. Since N_j and N_Y intersect in v, we set $N_j == N_j \setminus v$. The lower diagrams show how to shift the part of N_s between vertices x and y from the outside of Δ_j into the inside.

• Suppose N_s intersects with $N_j \in \mathfrak{N}_i$ in one vertex. One of the cut-nooses N_X, N_Y intersects with N_j in vertex v. Delete v from N_j and add N_X, N_Y. to $\mathfrak{N}_i$. Also here N_s can intersect with several disjoint cut-nooses of $\mathfrak{N}_i$ in this way. See the middle diagrams in Figure 6 for an example.
• Suppose N_s intersects with $N_j \in \mathfrak{N}_i$ in two vertices x, y and $N_s \cap \Delta_j = \emptyset$ (corresponding to special case 2.2) in Lemma 14). Since there is no vertex in the part of N_s between x and y we are allowed to shift that part entirely inside of Δ_j. See the lower diagrams in Figure 6 for an example. Thus, we obtain the first case above that N_s intersects with the empty disk Δ_j bounded by $N_j \in \mathfrak{N}_i$.

D Appendix: Concerning proof of Lemma 17

See figure 7.

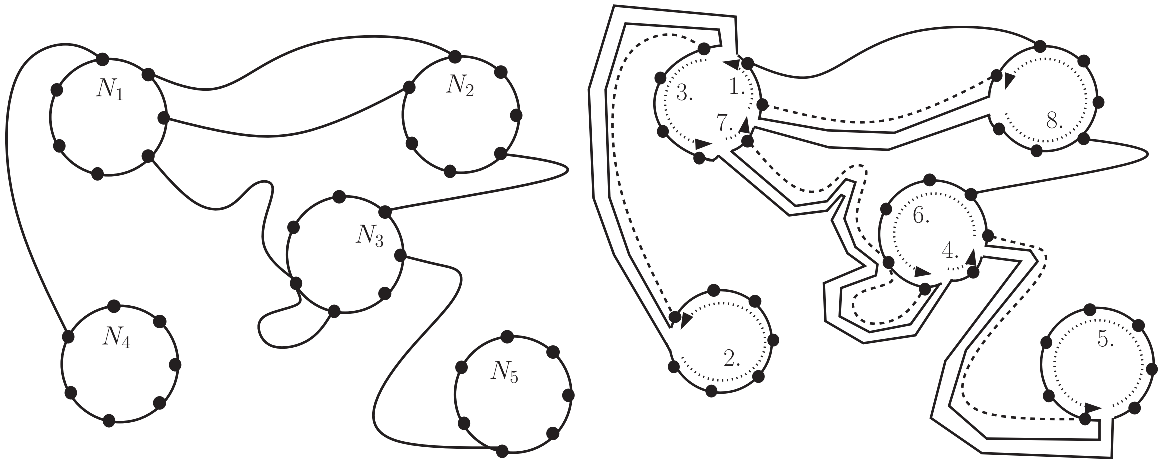

Figure 7: **Tree structure for fixing paths.** The left diagram shows a candidate connecting five cut-nooses $N_1, \ldots, N_5$ by paths. In the right diagram, the fixed paths are emphasized dashed. The nooses are connected by tunnels along these fixed paths. The order $\pi_{\mathbf{L}}$ of the vertices is illustrated by the labeled dotted and directed lines.

E Appendix: Proof of Lemma 18

As in the previous section, we preprocess the graph G' by deleting all vertices in $\mathfrak{N}$ that do not belong to any path in candidate $\mathbf{C}$. We also encode the vertices of $\mathfrak{N}$ with the same base values, except for 'S': we replace 'S' by the values 'S_1' to 'S_{2g}' since the number of cut-nooses is bounded by $2g$. The endpoints x, y of a fixed path with $x < y$ in $\pi_{\mathbf{L}}$ are encoded with S_i if $x \in N_i$.

Dynamic programming is done as described in the previous section with slight changes caused by the extension of Propositions 23 and 24. Due to Proposition 2 we can have the case that G' consists of several components $G'_1, G'_2, \ldots$. We simply do dynamic programming for each component separately.

Consider subgraph G_e bordered by tight noose O_e and $\mathfrak{N}_e \subset \mathfrak{N}$ as the cut-nooses intersecting G_e:

Proposition 27. *The subgraph G_e is the union of partial components $C_1, \ldots, C_q$ $(q \geq 1)$ such that for every i*

$C_i \cap (\bigcup_{r=1, r\neq i}^{q} C_r) \subseteq O_e \cap \mathfrak{N}_e$. Furthermore, for every i, j, h, $C_i \cap C_j \cap C_h = \emptyset$.

Proposition 28. *Each C_i is bordered by partial nooses of A_i of tight nooses of $\mathfrak{N}_e \setminus O_e$ and partial nooses of $B_i \subset O_e$ with $\bigcup_{i=1}^{q} A_i \cup B_i = G_e \cap (\mathfrak{N}_e \cup O_e)$ such that one of the following hold:*

1. $|A_i| = |B_i| = 1$ *with* $A_i \subset N_X$ *for a tight noose* $N_X \in \mathfrak{N}_e$,
2. $|A_i| = |B_i| \leq 2g$ *with each partial noose of* A_i *part of a different tight noose of* $\mathfrak{N}_e$.

For all partial components C_i, C_j *with property 2:* A_i *contains at least one partial noose that is part of a cut-noose of* $\mathfrak{N}_e$ *that has no partial noose in* A_j. *There are at most* $2g$ *components with property 2 and* $|\bigcup_{i=1}^{2g} A_i| \leq 2g$.

See Figure 8 for an illustration.

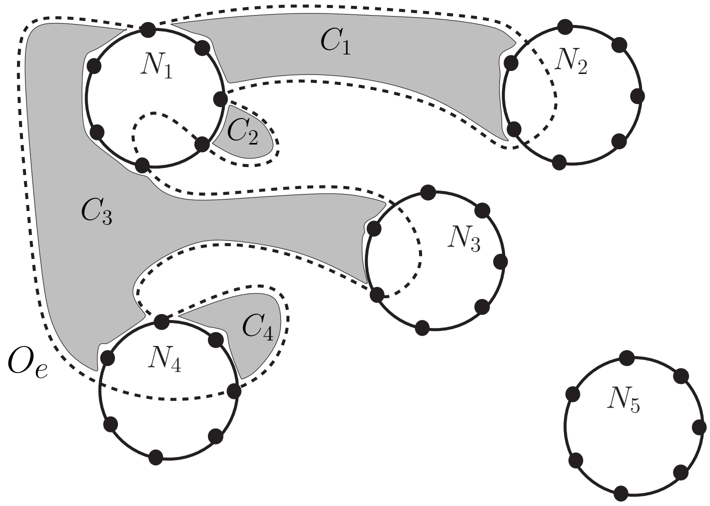

Figure 8: **Partial components with several cut-nooses.** The diagram shows how tight noose O_e intersects $\mathfrak{N}_e = \{N_1, \ldots, N_4\}$ and form the partial components $C_1, \ldots, C_4$. Observe that every partial noose of A_i $(1 \leq i \leq 4)$ is of a different cut-noose.

Component C_i with property 2 is encoded similarly as before only by replacing '$[_L$', '$]_L$' by '$[_1$','$]_1$' to '$[_{2g}$','$]_{2g}$': the last vertex in π of a vertex in a set of B_i connected to cut-noose $N_j \in \mathfrak{N}$ is encoded by '$[_j$' or '$]_j$'. In one set of B_i there are at most $2g$ vertices encoded by the new values. Because of the last statement of Proposition 28 the size of the union over all B_i is also bounded by $2g$. Hence, there are at most $4g^2$ vertices in O_e encoded with '$[_1$','$]_1$' to '$[_{2g}$','$]_{2g}$' and $O((2g\mathbf{bw}(G'))^{4g^2}$ possibilities for assigning these values to $V(O_e)$.

F Appendix: Proof of Lemma 19

We only give an idea of the proof, that is extending the proof of Lemma 8. Here we delete temporarily all cut-nooses of $\mathfrak{N}$ and construct the sc-decomposition $\langle T, \mu, \pi \rangle$ of G' of width at most $\sqrt{4.5} \cdot \sqrt{n}$. Now we simply make use of the argument that a middle set O_e intersects a cut-noose in at most two vertices for each cut-noose separately. Thus, we obtain at most two vertices more for O_e per cut-noose that O_e intersects.