RESEARCH-ARTICLE

Packing Short Cycles

MATTHIAS BENTERT, University of Bergen, Bergen, Hordaland, Norway

FEDOR V. FOMIN, University of Bergen, Bergen, Hordaland, Norway

PETR A GOLOVACH, University of Bergen, Bergen, Hordaland, Norway

TUUKKA KORHONEN, University of Copenhagen, Copenhagen, Hovedstaden, Denmark

WILLIAM LOCHET, University of Montpellier, Montpellier, Occitanie, France

FAHAD PANOLAN, University of Leeds, Leeds, West Yorkshire, U.K.

View all

Published: 07 October 2025
Online AM: 01 September 2025
Accepted: 23 August 2025
Revised: 11 August 2025
Received: 28 October 2024

Packing Short Cycles

MATTHIAS BENTERT, FEDOR V. FOMIN, and PETR A. GOLOVACH, Universitetet i Bergen, Bergen, Norway
TUUKKA KORHONEN, University of Copenhagen, Kobenhavn, Denmark
WILLIAM LOCHET, Université de Montpellier, Montpellier, France
FAHAD PANOLAN, University of Leeds, Leeds, United Kingdom
M. S. RAMANUJAN, University of Warwick, Coventry, United Kingdom
SAKET SAURABH, The Institute of Mathematical Sciences, Chennai, India and Universitetet i Bergen, Bergen, Norway
KIRILL SIMONOV, University of Bergen, Bergen, Norway

Cycle packing is a fundamental problem in optimization, graph theory, and algorithms. Motivated by recent advancements in finding vertex-disjoint paths between a specified set of vertices that either minimize the total length of the paths [Björklund and Husfeldt, ICALP 2014; Mari et al., SODA 2024] or request the paths to be shortest [Lochet, SODA 2021], we consider the following cycle packing problems: Min-Sum Cycle Packing and Shortest Cycle Packing.

In Min-Sum Cycle Packing, we try to find, in a weighted undirected graph, k vertex-disjoint cycles of minimum total weight. Our first main result is an algorithm that, for any fixed k, solves the problem in polynomial time. We complement this result by establishing the W[1]-hardness of Min-Sum Cycle Packing parameterized by k. The same results hold for the version of the problem where the task is to find k edge-disjoint cycles.

Our second main result concerns Shortest Cycle Packing, which is a special case of Min-Sum Cycle Packing that asks to find a packing of k shortest cycles in a graph. We prove this problem to be Fixed-Parameter Tractable (FPT) when parameterized by k on weighted planar graphs. We also obtain a polynomial kernel for the edge-disjoint variant of the problem on planar graphs. Whether Min-Sum Cycle Packing is FPT on planar graphs, or Shortest Cycle Packing on general graphs, remains open.

A preliminary version of these results appeared in the proceedings of SODA 2025.
Tuukka Korhonen—Work was performed while at University of Bergen, Norway.
Kirill Simonov—Work was performed while at Hasso Plattner Institute, University of Potsdam, Germany.
This work is supported by the Research Council of Norway under BWCA project (grant no. 314528), the Franco-Norwegian AURORA project (grant no. 349476), the UKRI EPSRC (grant EP/V044621/1), the Swarnajayanti Fellowship (grant DST/SJF/MSA-01/2017-18), and the ERC Horizon 2020 research and innovation programme (grant no. 819416).
Authors' Contact Information: Matthias Bentert, Universitetet i Bergen, Bergen, Norway; e-mail: matthias.bentert@uib.no; Fedor V. Fomin, Department of Informatics, Universitetet i Bergen, Bergen, Norway; e-mail: fedor.fomin@uib.no; Petr A. Golovach (corresponding author), Universitetet i Bergen, Bergen, Norway; e-mail: Petr.Golovach@uib.no; Tuukka Korhonen, University of Copenhagen, Kobenhavn, Denmark; e-mail: tuko@di.ku.dk; William Lochet, Université de Montpellier, Montpellier, France; e-mail: william.lochet@gmail.com; Fahad Panolan, University of Leeds, Leeds, United Kingdom; e-mail: f.panolan@leeds.ac.uk; M. S. Ramanujan, University of Warwick, Coventry, United Kingdom; e-mail: R.Maadapuzhi-Sridharan@warwick.ac.uk; Saket Saurabh, The Institute of Mathematical Sciences, Chennai, India and Universitetet i Bergen, Bergen, Norway; e-mail: saket@imsc.res.in; Kirill Simonov, University of Bergen, Bergen, Norway; e-mail: Kirill.Simonov@uib.no.

ACM 1549-6333/2025/10-ART8
https://doi.org/10.1145/3765285

CCS Concepts: • **Mathematics of computing → Graph algorithms**; • **Theory of computation → Parameterized complexity and exact algorithms**; *Problems, reductions and completeness*;

Additional Key Words and Phrases: vertex-disjoint cycles, planar graphs, parameterized complexity

ACM Reference format:
Matthias Bentert, Fedor V. Fomin, Petr A. Golovach, Tuukka Korhonen, William Lochet, Fahad Panolan, M. S. Ramanujan, Saket Saurabh, and Kirill Simonov. 2025. Packing Short Cycles. *ACM Trans. Algor.* 22, 1, Article 8 (October 2025), 35 pages.
https://doi.org/10.1145/3765285

1 Introduction

We consider the following problem.

Min-Sum Cycle Packing

Input:	A graph G with a weight function $w\colon E(G) \to \mathbb{Z}_{>0}$, an integer $k \geq 1$, and $\ell \in \mathbb{Z}_{\geq 0}$.
Task:	Decide whether there is a family (packing) of k (vertex-)disjoint cycles whose total length is at most ℓ.

We also consider the variant of the problem, called Min-Sum Edge-Disjoint Cycle Packing, where the task is to find a packing of edge-disjoint cycles of total length at most ℓ.

Min-Sum Cycle Packing could be considered a "relaxation" of the notoriously difficult Min-Sum Disjoint Paths. Recall that in the Min-Sum Disjoint Paths problem, we are given a graph with a set of terminal pairs $(s_1, t_1), \ldots, (s_k, t_k)$. The task is either to connect all terminal pairs (s_i, t_i) by pairwise vertex-disjoint paths of minimum total length or to decide that there is no set of pairwise disjoint paths. Of course, the existence of a polynomial-time algorithm solving Min-Sum Disjoint Paths for fixed k would imply a polynomial-time algorithm solving Min-Sum Cycle Packing. Unfortunately, no such algorithm is known. Björklund and Husfeldt [4] give an algorithm with running time $O(n^{11})$ for finding two disjoint s_i-t_i-paths of minimal total length in an n-vertex graph. For $k > 2$, the complexity of the problem is a long-standing open problem. Whether the problem is polynomial-time solvable for $k = 3$ remains open even on planar graphs. The main reason for our interest in Min-Sum Cycle Packing was the lack of any progress on the complexity of Min-Sum Disjoint Paths.

Our first theorem establishes the membership of Min-Sum Cycle Packing and Min-Sum Edge-Disjoint Cycle Packing in complexity class XP when parameterized by k. More precisely, we show the following.

Theorem 1. *Min-Sum Cycle Packing and Min-Sum Edge-Disjoint Cycle Packing can be solved in $n^{O(k^6)}$ time, where n is the number of vertices in the graph.*

Further, we show that Theorem 1 can be generalized for the variant of the problem with individual constraints on the length of the cycles in a packing. Formally, in Min-Vector Cycle Packing, we are given a graph G, a positive integer k, and k positive integers $\ell_1, \ldots, \ell_k$. Then the task is to find k (vertex) disjoint cycles $C_1, \ldots, C_k$ so that the length of C_i is upper-bounded by ℓ_i for each $i \in \{1, \ldots, k\}$. We prove that Min-Vector Cycle Packing and the variant where we want the cycles to be edge-disjoint, called Edge-Disjoint Min-Vector Cycle Packing, can also be solved in $n^{O(k^6)}$ time.

We complement Theorem 1 by a computational lower bound, showing that the existence of an **Fixed-Parameter Tractable (FPT)** algorithm for Min-Sum Cycle Packing is unlikely.

Theorem 2. *Min-Sum Cycle Packing is* W[1]*-hard in subcubic graphs with unit edge weights when parameterized by* k.

Because the lower bound is obtained for subcubic graphs, the result also holds for Min-Sum Edge-Disjoint Cycle Packing.

A special but still fascinating case of Min-Sum Disjoint Paths is the Disjoint Shortest Paths problem. Here, we want to connect terminal pairs by vertex-disjoint *shortest* paths. Equivalently, this is the variant of Min-Sum Disjoint Paths, where we request to connect terminals by disjoint paths whose total length does not exceed $\sum_{1\leq i\leq k} \mathrm{dist}(s_i, t_i)$.

The "relaxation" of Disjoint Shortest Paths as a cycle packing is the variant of Min-Sum Cycle Packing where all the cycles in the packing should be shortest cycles, that is, $\ell = kg$ where g is the girth of G. (We remind that the girth of a weighted graph is the minimum length of its cycles.)

Shortest Cycle Packing

Input:	A graph G with a weight function $w\colon E(G) \to \mathbb{Z}_{>0}$ and an integer $k \geq 1$.
Task:	Decide whether there is a family (packing) of k vertex-disjoint cycles, each of minimum length.

For example, if all edges of G are of weight one and the girth of G is three, then Shortest Cycle Packing becomes the Triangle Packing problem.

By Theorem 1, Shortest Cycle Packing is solvable in polynomial time for a fixed number k of cycles. We do not know whether it is W[1]-hard or FPT parameterized by k. Our next theorem establishes the fixed-parameter tractability of Shortest Cycle Packing on planar graphs.

Theorem 3. *Shortest Cycle Packing is solvable in time* $k^{O(k)} \cdot n^{O(1)}$ *on planar graphs with* n *vertices.*

We do not know whether Shortest Cycle Packing admits a polynomial kernel on planar graphs. However, the methods developed for the proof of Theorem 3 allow establishing a polynomial kernel for the Edge-Disjoint Shortest Cycle Packing problem, the version of Shortest Cycle Packing where the cycles should be edge-disjoint.

Theorem 4. *Edge-Disjoint Shortest Cycle Packing on planar graphs admits a polynomial kernel such that the output graph has* $O(k^2)$ *vertices.*

The kernelization algorithm also implies that Edge-Disjoint Shortest Cycle Packing can be solved in $k^{O(k)} \cdot n^{O(1)}$ time on planar graphs.

Due to duality in planar graphs, Edge-Disjoint Shortest Cycle Packing in planar graphs is equivalent to finding k disjoint minimum edge cuts; here we assume that two edge cuts (X_1, Y_1) and (X_2, Y_2) are assumed to be disjoint if $E(X_1, Y_1) \cap E(X_2, Y_2) = \emptyset$. Here, for each $i \in [2]$, (X_i, Y_i) partitions the vertex set and $E(X_i, Y_i)$ denotes the set of edges with exactly one endpoint in X_i. Thus, Theorem 4 implies the existence of a polynomial kernel for the problem of packing edge-disjoint minimum cuts. We observe that this problem parameterized by k on general graphs is W[1]-hard even when restricted to graphs with unit edge weights.

Remarks on Impact of the Weight Function. Our algorithm in Theorem 1 is strongly polynomial for each k. This rules out standard modifications used for converting weighted problems to the unweighted case, for instance, by subdividing edges according to their weight which could be

exponential in the graph size. In fact, we do not use the fact that the weights are integers and this result can easily be generalized to positive rationals (or even positive reals assuming the real RAM model). The same holds for Theorem 3.

1.1 Overview of the Methods

Here, we give a high-level overview over how we achieve the different results.

1.1.1 Min-Sum Cycle Packing. The algorithms for Min-Sum Cycle Packing, Min-Sum Edge-Disjoint Cycle Packing, Min-Vector Cycle Packing, and Edge-Disjoint Min-Vector Cycle Packing are based on a combinatorial result about the number of paths in graphs that are no-instances for the Min-Sum Cycle Packing problem. In particular, we prove that if an edge-weighted graph G does not contain a collection of k pairwise vertex-disjoint cycles of total length at most ℓ, then the number of paths of length at most ℓ in G is at most $n^{O(k^5)}$ (Lemma 3.6). Given this lemma, obtaining $n^{O(k^6)}$-time algorithms for these problems is quite simple as shown in the next paragraph.

In the case of Min-Sum Cycle Packing, we start enumerating paths of length at most ℓ, and if there are more than $n^{\Omega(k^5)}$ of them, then we know that it is a yes-instances. Otherwise, from the set of all paths of length at most ℓ, we obtain the set of all cycles of length at most ℓ and then enumerate all collections of k of them in time $n^{O(k^6)}$. To generalize this to Min-Vector Cycle Packing, we assume that ℓ_1 is the smallest of the length bounds, apply the same enumeration strategy with the parameters ℓ_1 and k, and then branch on the $n^{O(k^5)}$ enumerated cycles, to again obtain the $n^{O(k^6)}$ running time. These algorithms generalize to the edge-disjoint case simply by applying the same combinatorial result about vertex-disjoint cycles, but inserting the additional check when branching that the cycles we choose are edge-disjoint.

Thus, the main technical ingredient of the proof of Theorem 1 is the proof of this combinatorial result (Lemma 3.6). We sketch the ideas here. First, consider the case of $k = 1$. If G does not contain a cycle of length at most ℓ, then the number of paths of length at most $\ell/2$ in G is at most $\binom{n}{2}$, because having two distinct paths of length at most $\ell/2$ between the same pair of vertices would imply the existence of a cycle of length at most ℓ. Then, we observe that for any integer $c \geq 1$ and length-bound ℓ_0, if G contains at most N paths of length at most ℓ_0, then G contains at most N^c paths of length $c \cdot \ell_0$, and therefore conclude that G contains at most $O(n^4)$ paths of length at most ℓ.

Now, we extend the above argument to the general case of $k > 1$. First, let $\tilde{C}$ be a subgraph of G comprising a maximal collection of pairwise vertex-disjoint cycles each of length at most ℓ/k. By our assumption, $\tilde{C}$ consists of at most $k - 1$ cycles, and the graph $G - V(\tilde{C})$ does not contain a cycle of length at most ℓ/k. We then consider a path P of length at most ℓ in G. By considering how the path P intersects with $\tilde{C}$, we divide it into *segments* of two types: maximal subpaths of P that are internally vertex-disjoint with $V(\tilde{C})$, and maximal subpaths of P that are contained in $\tilde{C}$.

We next argue that if the number of such segments is more than $\Omega(k^4)$, then we can in fact construct a family of k vertex-disjoint cycles of total length at most ℓ, contradicting the premise of the lemma. In order to do so, we show that if there is a cycle C in $\tilde{C}$ such that P "enters and exits" C more than $\Omega(k^3)$ times, then we can find the claimed collection of k vertex-disjoint cycles in $P \cup C$. The proof of this proceeds by "cleaning up" the interactions of P and C into one of three cases with the help of the Erdös-Szekeres theorem, and then demonstrating a simple construction in each of the cases.

After proving that P decomposes into $O(k^4)$ segments that are either internally vertex-disjoint with $\tilde{C}$ or contained in $\tilde{C}$, it remains to show that there are only $n^{O(k)}$ possible choices for each

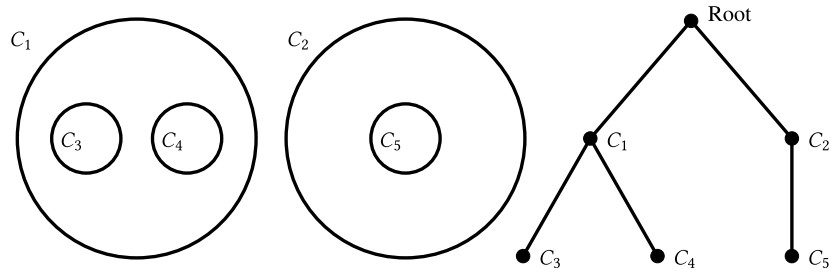

Fig. 1. A solution $C = \{C_1, \ldots, C_5\}$ and the tree representing C.

segment and since P was chosen arbitrarily, this would in turn allow us to bound the number of possibilities for P. For subpaths internally vertex-disjoint with $\tilde{C}$ we use the property that $V(G) - V(\tilde{C})$ does not contain a cycle of length at most ℓ/k, implying that $G - V(\tilde{C})$ contains at most $n^{O(k)}$ paths of length at most ℓ. For subpaths contained in $\tilde{C}$, it is simple to observe that there are at most $O(n^2)$ possible choices for them, as each path in $\tilde{C}$ is defined by the choice of two vertices and the choice of a side to travel inside a cycle. This concludes the informal overview of the proof of Theorem 1.

1.1.2 Cycle Packing on Planar Graphs. The proof of Theorem 3 is based on constructing a laminar family of disjoint cycles representing all shortest cycles and decomposing this family into a tree. We use random separation on such trees to "filter out" the most "promising" parts and combine branching arguments with an FPT algorithm for computing Independent Set in map graphs. In more detail:

Let G be a weighted planar graph G of (weighted) girth g. Without loss of generality, we can assume that each vertex and edge of G is included in a shortest cycle. Otherwise, we can preprocess the graph in polynomial time and delete vertices and edges that do not appear in a shortest cycle. Consider a plane embedding of G. Packing *facial* cycles of G is equivalent to computing an Independent Set in a map graph. Recall that a map graph is the intersection graph of connected and internally disjoint regions of the Euclidean plane [10]. Indeed, facial cycles of G are vertex-disjoint if and only if the corresponding vertices in the map graph induced by G are not adjacent. For computing an Independent Set in a map graph, Chen [9] gave an EPTAS; the approximation algorithm of Chen immediately implies an FPT algorithm for Independent Set on map graphs, and hence for packing shortest facial cycles. The first natural approach to try would be to reduce Shortest Cycle Packing to Independent Set in map graphs. This approach fails because a solution may contain many non-facial cycles. The main challenge in obtaining the algorithm for Shortest Cycle Packing is identifying such cycles. However, we will use Chen's algorithm as a subroutine for an appropriate base case.

In order to handle non-facial cycles, we construct a rooted tree whose nodes are shortest cycles, called a **Laminar Shortest Cycle Tree (LSCT)**. Constructions with similar properties are used to approximate the maximum size of cycle packings for uncrossable families of cycles [24, 35, 36], as well as for finding the minimum cycle basis for a planar graph (see, e.g., [26]). However, our algorithm needs a new decomposition with specific properties tailored to the properties of shortest cycles.

Suppose that C is a family of vertex-disjoint shortest cycles of G. Then C is a laminar family of cycles. Some of these cycles may be nested. This means that a rooted tree can represent C naturally (see Figure 1). We assume that the leaves of the tree are facial cycles in the embedding.

It is convenient to assume that in the plane embedding of G the frontier of the external face is a shortest cycle. (It is not difficult to prove that such an embedding exists and it can be constructed in polynomial time.) We fix such an embedding and construct an LSCT tree $\mathcal{T}(G)$. The root of the tree is the facial cycle of the external face. We construct the tree by recursively processing shortest cycles that are current leaves of the already constructed tree.

If C is a facial cycle, it becomes a leaf of $\mathcal{T}(G)$. When C is not a facial cycle, we consider two possibilities:

(i) C contains two vertices s and t (called the *poles*) that are at a distance $g/2$ from each other in C and such that there is an s-t-path P of length $g/2$ whose edges and internal vertices are in the internal face of C. Then we say that C is a *splittable cycle* and we create an *S-node* of $\mathcal{T}(G)$ from C. We find an inclusion-maximal family $\mathcal{P} = \{P_0, P_1, \ldots, P_\ell\}$ of internally vertex-disjoint s-t-paths of length $g/2$ where P_0 and P_ℓ are the paths in C and the other paths have their edges and internal vertices in the internal face of C. We assume the paths are indexed in the clockwise order from s. We define the cycles formed by the consecutive paths in $\mathcal{P}$ (see Figure 3(a)) to be the children of C. We show that the poles (if they exist) must be unique, so this step is well defined.

(ii) If C has no poles, we call such cycles *unsplittable* and we make a *U-node* of $\mathcal{T}(G)$ from C. We find an inclusion-maximal laminar family $\mathcal{C} = \{C_1 \ldots, C_\ell\}$ of shortest cycles distinct from C such that (i) the cycles of $\mathcal{C}$ are inside C in the sense that they have no elements in the external face of C, (ii) for every $i, j \in [\ell]$, cycles C_i and C_j are outside each other, that is, C_j has no elements in the internal face of C_i and vice versa, and (iii) the cycles of $\mathcal{C}$ are maximal in the sense that for each $i \in [\ell]$, there is no shortest cycle C' distinct from C and C_i such that C_i is inside C'. We set the cycles of $\mathcal{C}$ to be the children of C in $\mathcal{T}(G)$ (see Figure 3(b)).

We prove that an LSCT can be constructed in polynomial time and it contains all shortest cycles in the following sense: every shortest cycle C is either a node of $\mathcal{T}(G)$ or the tree has an S-node S with poles s and t such that C is formed by two distinct paths of $\mathcal{P}$ constructed for S.

The bottleneck in directly using LSCT at this point to find a solution, is dealing with S-nodes and cycles formed by paths of $\mathcal{P}$ as these cycles are not nodes of the tree. To avoid having to consider such cycles in the solution, we argue that the random separation technique [8] can be used to highlight pairs of paths in the families $\mathcal{P}$ that could be cycles in a potential solution. We then modify $\mathcal{T}(G)$ in such a way that for the obtained tree $\mathcal{T}^*$, every cycle of some potential solution is a node of $\mathcal{T}^*$.

Finally, we use a recursive branching algorithm to find a solution. First, we check in FPT time whether there is a solution formed by the facial cycles using the algorithm for Independent Set on map graphs. Otherwise, if we fail to find a solution formed by facial cycles, we conclude that for a yes-instance, there should be a cycle C in $\mathcal{T}^*$ and a solution containing C such that (i) the only cycles in this solution that are in the internal face of C are facial cycles and (ii) the solution contains $k' \geq 1$ facial cycles that are inside C. The crucial observation is that we can choose C to be one of the lowest nodes in $\mathcal{T}^*$ having the property that there are k' vertex-disjoint facial cycles inside C, and we have at most k possibilities to choose C for a given k'. Here, again we can use the algorithm for Independent Set on map graphs as a black box. We branch on $O(k^2)$ possible choices of k' and C. For each branch, we delete the vertices and edges of G that are in the inner face of C and modify $\mathcal{T}^*$ respectively, decrease the parameter by k', and recurse. This results in the algorithm solving Shortest Cycle Packing in time $k^{O(k)} \cdot n^{O(1)}$, which proves Theorem 3.

In the proof of Theorem 4 which provides a polynomial kernel for Edge-Disjoint Shortest Cycle Packing on planar graphs, we use a similar approach. In this case the solution is easier because when the LSCT has at least $4k$ leaves, that is, shortest facial cycles of G, the subgraph of the dual graph induced by the shortest facial cycles of G has an independent set of size at least k

(by the Four-Color Theorem). This means that (G, k) is a yes-instance of Edge-Disjoint Shortest Cycle Packing. In the case when the number of leaves is less than $4k$, we can mark $O(k^2)$ nodes of $\mathcal{T}(G)$ in such a way that a yes-instance has a solution containing only marked shortest cycles or cycles formed by paths from $\mathcal{P}$ constructed for S-nodes. The idea behind the marking is to choose the lowest "useful" cycles. This leads to a kernel for Edge-Disjoint Shortest Cycle Packing with $O(k^2)$ vertices and edges and proves Theorem 4.

1.2 Related Work

The complexity of Min-Sum Disjoint Paths is widely open on general and planar graphs. Björklund and Husfeldt [4] give a randomized algorithm with running time $O(n^{11})$ for finding two disjoint s_i-t_i-paths of minimum total length. It is not known whether the problem is polynomial-time solvable for $k = 3$. On planar graphs, several polynomial-time algorithms are known for the special cases when the terminals belong to two faces [7, 11, 13, 30]. Recently, Mari et al. [32] designed an algorithm for this problem with running time $k^{2^{O(k)}}$ when the input graph is a grid.

Disjoint Shortest Paths is NP-hard when k is part of the input [17] and W[1]-hard parameterized by k [2]. Lochet [31] gives a polynomial-time algorithm for any fixed k in undirected graphs without weights, see also Bentert et al. [2] for an improvement of the running time of Lochet's algorithm. Whether Disjoint Shortest Paths is FPT parameterized by k on planar graphs is an interesting open question.

Very little was known about the complexity of Min-Sum Cycle Packing and Shortest Cycle Packing. Both problems are NP-complete when restricted to planar graphs [23, 25, 27] because they generalize the well-known triangle packing problem. Rautenbach and Regen [33] show that for graphs of girth $g \in \{4, 5\}$, Shortest Cycle Packing and Edge-Disjoint Shortest Cycle Packing allow polynomial-time algorithms for instances with maximum degree 3 but are APX-hard for instances with maximum degree 4. For each $g \geq 6$, both problems are APX-hard already for graphs with maximum degree 3.

Approximation algorithms of ratio $(3 + \varepsilon)$ for Shortest Cycle Packing and Edge-Disjoint Shortest Cycle Packing on planar graphs follow from the framework of Schlomberg et al. [36] for uncrossing families of cycles.

When the sizes of the cycles are bounded, cycle packing is a special case of the subgraph packing problem. For such problems a plethora of parameterized and kernelization algorithms are known [12]. However, all these algorithms work only for cycles of constant sizes. For vertex-disjoint (or edge-disjoint) cycle packing (without conditions on the lengths of cycles), a polynomial kernel on planar graphs, as well as more general classes of sparse graphs, are known [5, 21].

2 Preliminaries

For a positive integer p, we use $[p]$ to denote the set $\{1, \ldots, p\}$, $[p]_0$ for the set $\{0, 1, \ldots, p\}$, and we define $[p, q] = \{p, \ldots, q\}$ for integers $p < q$.

Graphs. In this article, we consider finite undirected graphs and refer to the textbook by Diestel [14] for undefined notion. By default, the considered graphs are simple but we may allow multiple edges and loops in some occasions. Let $G = (V, E)$ be a graph. We use $V(G)$ and $E(G)$ to denote the set of vertices and the set of edges of G, respectively. We use n and m to denote the number of vertices and edges in G, respectively, unless this creates confusion, in which case we clarify the notation explicitly. For a vertex subset $U \subseteq V$, we use $G[U]$ to denote the subgraph of G induced by the vertices in U and $G - U$ to denote $G[V \setminus U]$. For two graphs G_1 and G_2, the *intersection* $G = G_1 \cap G_2$ is the graph with $V(G) = V(G_1) \cap V(G_2)$ and $E(G) = E(G_1) \cap E(G_2)$, and the *union* $G = G_1 \cup G_2$ is the graph with $V(G) = V(G_1) \cup V(G_2)$ and $E(G) = E(G_1) \cup E(G_2)$. A *path* $P = v_0 \cdots v_\ell$

is a graph with vertex set $\{v_0, v_1, \ldots, v_\ell\}$ and edge set $\{\{v_{i-1}, v_i\} \mid i \in [\ell]\}$; the vertices v_0 and v_ℓ are the *endpoints* of P and the other vertices are *internal*. For a path with endpoints s and t, we say that P is an s-t-path. For a path P, a subgraph R of P with a single connected component is called a *subpath* of P and we use the notation $R \subseteq P$ to denote this. In the case of unweighted graphs, the *length* of P is defined as the number of edges, and if we are given edge *weights* $w\colon E(G) \to \mathbb{Z}_{>0}$ then the *length* of $P = v_0 \cdots v_\ell$ is $w(P) = \sum_{i=1}^{\ell} w(\{v_{i-1}, v_i\})$. A *cycle* C is a path along with an additional edge between the two endpoints. The length of a cycle is defined in the same way as the length of a path. The *girth* of G is the minimum length of a cycle in G; $g(G) = +\infty$ if G is a forest. An *(edge) cut* of G is a partition (X, Y) of $V(G)$; the set of edges $E(X, Y) = \{\{x, y\} \mid x \in X,\ y \in Y\}$ is the *cut-set*. If G is a weighted graph then the weight of a cut is the weight of its cut-set.

A graph is *planar* if it can be drawn on the plane such that its edges do not cross each other. Such a drawing is called a *planar embedding* of the graph and a planar graph with a planar embedding is called a *plane* graph. The planarity of a graph can be tested and a planar embedding can be found (if it exists) in linear time by the results of Hopcroft and Tarjan [28]. The *faces* of a plane graph are the open regions of the plane bounded by a set of edges and that do not contain any other vertices or edges. The outer region is the *external* face and the other faces are *internal*. The set of vertices and edges appearing on the closed walk bounding the face forms its *frontier*. Given a plane graph G, we use $F(G)$ to denote its set of faces. If a cycle C of G is the frontier of some face, then C is a *facial cycle*. For a plane graph G, its *dual* graph $G^* = (F(G), E^*)$ has the set of faces of G as the vertex set, and for each $e \in E(G)$, G^* has the *dual* edge e^* whose endpoints are either two faces having e on their frontiers or e^* is a self-loop at f if e is in the frontier of exactly one face f (i.e., e is a bridge of G). Observe that G^* is not necessarily simple even if G is a simple graph. If G is a weighted graph then it is standard to define the weights of the edges of the dual graph by setting the weight of e^* to be equal to the weight of e for each edge $e \in E(G)$. It is well known that finding a shortest cycle for a plane graph is equivalent to computing a minimum cut for the dual graph. More precisely, C is a shortest cycle in a weighted plane graph G if and only if the set $\{e^* \mid e \in E(C)\}$ of dual edges is a cut-set of G^* of minimum weight. In a weighted graph, the distance between two vertices is the weight of a minimum-weight path between them.

The celebrated four-color theorem [1, 34] implies the following observation about independent sets in planar graphs. Observation 2.1. *An n-vertex planar graph has an independent set of size at least $\frac{n}{4}$.*

A *map* graph is the intersection graph of a finite family of simply connected and internally disjoint regions of the plane. In this article, we assume that each map graph G is given by its embedding, that is, by a plane graph H such that $V(G) \subseteq F(H)$ and two vertices $f_1, f_2 \in V(G)$ are adjacent if and only if the frontiers of the faces f_1 and f_2 of H share at least one point (a vertex or an edge of H). It was shown by Chen [9] that Independent Set is FPT on map graphs.[1]

Proposition 2.2 ([9]). *It can be decided in $2^{O(k)} \cdot |V(H)|^2$ time whether a map graph G given by its embedding H has an independent set of size at least k.*

Parameterized Complexity. We refer to the textbooks by Cygan et al. [12] and by Downey and Fellows [16] for a detailed introduction. Here, we just briefly remind the main notions. A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$ where Σ^* is a set of strings over a finite alphabet Σ. An input of a parameterized problem is a pair (x, k) where x is a string over Σ and $k \in \mathbb{N}$ is a *parameter*. A parameterized problem is FPT if it can be solved in $f(k) \cdot |x|^{O(1)}$ time for

[1]The result of Chen [9] is stated as an EPTAS but the approximation algorithm immediately implies an FPT algorithm.

some computable function f. The complexity class FPT contains all fixed-parameter tractable parameterized problems. A parameterized problem is in the class XP if it can be solved in $|x|^{f(k)}$ time for a computable function f. A *kernelization algorithm* or *kernel* for a parameterized problem L is a polynomial-time algorithm that takes as its input an instance (x, k) of L and returns an instance (x', k') of the same problem such that (i) $(x, k) \in L$ if and only if $(x', k') \in L$ and (ii) $|x'| + k' \leq f(k)$ for some computable function $f\colon \mathbb{N} \to \mathbb{N}$. The function f is the *size* of the kernel; a kernel is *polynomial* if f is a polynomial.

We conclude this section with two auxiliary results used in our article. We need the following classical result of Erdős and Szekeres [18].

Proposition 2.3 (Special case of Erdős-Szekeres Theorem [18]). *Each sequence of more than $(r-1)^2$ distinct real numbers contains either an increasing subsequence of length r or a decreasing subsequence of length r.*

For kernelization for Edge-Disjoint Shortest Cycle Packing on planar graphs, we use the algorithm of Frank and Tardos [22] to compress the weights as it is standard for kernelization algorithms [19].

Proposition 2.4 ([22]). *There is an algorithm that, given a vector $w \in \mathbb{Q}^r$ and an integer N, in (strongly) polynomial time finds a vector $\overline{w} \in \mathbb{Z}^r$ with $\|\overline{w}\|_\infty \leq 2^{4r^3} N^{r(r+2)}$ such that $\mathsf{sign}(w \cdot b) = \mathsf{sign}(\overline{w} \cdot b)$ for all vectors $b \in \mathbb{Z}^r$ with $\|b\|_1 \leq N - 1$.*

3 Min-Sum Cycle Packing Is in XP

In this section, we prove Theorem 1 and show that Min-Sum Cycle Packing and Min-Sum Edge-Disjoint Cycle Packing can be solved in $n^{O(k^6)}$ time. The algorithms for these two problems are based on the same argument. We therefore focus on Min-Sum Cycle Packing and then explain how to extend the algorithm for the case of edge-disjoint cycles.

First, we show that if a path and a cycle intersect in a complicated way, then we can construct a collection of many pairwise disjoint short cycles. Before this, we introduce some notation about the intersection of a path and a cycle.

Let P be a path in G and C a cycle in G so that P and C intersect in at least one vertex. A *chord* of P relative to C is a maximal non-empty subpath $R \subseteq P$ so that the endpoints of R are in $V(C)$, the internal vertices of R are disjoint from $V(C)$, and $E(R)$ is disjoint from $E(C)$. A *tail* of P relative to C is a maximal non-empty subpath $T \subseteq P$ whose one endpoint is an endpoint of P and is disjoint from $V(C)$, the other endpoint is in $V(C)$, and all internal vertices are not contained in $V(C)$. Note that P has between zero and two tails relative to C. Note also that $E(P)$ is the disjoint union of $E(P) \cap E(C)$, the edges of the chords of P relative to C, and the edges of the tails of P relative to C.

Observation 3.1. *The number of chords of P relative to C is the number of connected components of $P \cap C$ minus one.*

Lemma 3.2. *Let (G, w) be an edge-weighted graph, C a cycle in G of length at most ℓ, and P a path in G of length at most ℓ. If P has at least $128k^3$ chords relative to C, then G contains a collection of k pairwise vertex-disjoint cycles with total length at most ℓ.*

Proof. We assume that P has at least $128k^3$ chords relative to C and construct the desired collection of pairwise disjoint cycles.

First, we select a collection $\mathcal{R}$ of at least $64k^3$ chords of P so that the chords in $\mathcal{R}$ are vertex-disjoint. This is done simply by following along P and selecting every second chord. Now, we index $V(C)$ with integers from 1 to $|V(C)|$, in an order following the cycle. Then we associate with each chord $R \in \mathcal{R}$ the pair $(s(R), t(R)) \in [|V(C)|]$ of numbers such that $s(R) < t(R)$ correspond to

the two endpoints of R (which are in $V(C)$). Note that as the chords are vertex disjoint, all of the numbers associated to them are distinct.

Let $R_1, R_2 \in \mathcal{R}$ be two chords with $s(R_1) < s(R_2)$. We say that R_1 and R_2 are:

— *consecutive* if $s(R_1) < t(R_1) < s(R_2) < t(R_2)$,
— *crossing* if $s(R_1) < s(R_2) < t(R_1) < t(R_2)$, and
— *parallel* if $s(R_1) < s(R_2) < t(R_2) < t(R_1)$.

Note that any two chords are either consecutive, crossing, or parallel.

We then show that we can obtain a large enough subcollection of $\mathcal{R}$ such that every pair of chords in this collection are related in the same way (i.e., one of consecutive, crossing, or parallel).

Claim 3.3. *There is $\mathcal{R}' \subseteq \mathcal{R}$ with $|\mathcal{R}'| \geq 4k$ so that all chords in $\mathcal{R}'$ are either pairwise consecutive, pairwise crossing, or pairwise parallel.*

Proof of the Claim. We say $i \in [|V(C)|]$ *touches* a chord $R \in \mathcal{R}$ if $s(R) \leq i \leq t(R)$. First, suppose that no $i \in [|V(C)|]$ touches more than $16k^2$ chords. Then, the greedy algorithm that repeatedly chooses chords R with the smallest $t(R)$ and discards all other chords touched by $t(R)$ manages to collect a set of at least $8k \geq 4k$ pairwise consecutive chords.

Otherwise, there is $i \in [|V(C)|]$ that touches at least $16k^2 + 1$ chords, implying that there is a collection $\mathcal{R}_1 \subseteq \mathcal{R}$ of at least $16k^2$ chords R with $s(R) < i < t(R)$ (recall that the chords in $\mathcal{R}$ are vertex disjoint). By the Erdős-Szekeres theorem (Proposition 2.3), there exists a subcollection $\mathcal{R}_2 \subseteq \mathcal{R}_1$ of size at least $4k$ so that when the chords in $\mathcal{R}_2$ are sorted by $s(R)$, the endpoints $t(R)$ are either all in increasing order, or all in decreasing order. In the former case, the chords in $\mathcal{R}_2$ are pairwise crossing, and in the latter case, the chords in $\mathcal{R}_2$ are pairwise parallel. ◁

We next consider the three cases arising from the above claim. First, if we have a collection $\mathcal{R}'$ of $4k$ pairwise consecutive chords, then by forming for each chord $R \in \mathcal{R}'$ a cycle consisting of R and the path in C between $s(R)$ and $t(R)$, we obtain a collection of $4k$ pairwise vertex-disjoint cycles. As the edges of these cycles come from $E(P) \cup E(C)$, their total length is at most 2ℓ, so by choosing the k shortest of them we obtain a collection of k pairwise vertex-disjoint cycles with a total length at most $\ell/2$.

Second, suppose we have a collection $\mathcal{R}'$ of $4k$ pairwise crossing chords. We index them as $R_1, \ldots, R_{4k}$, so that $s(R_i) < s(R_j)$ if $i < j$. Note that now, $s(R_{4k}) < t(R_1)$ and $t(R_i) < t(R_j)$ whenever $i < j$. For each $i \in [2k]$, we construct a cycle C_i by taking the union of R_{2i-1} and R_{2i}, and connecting them by two paths in C, the first between $s(R_{2i-1})$ and $s(R_{2i})$, and the second between $t(R_{2i-1})$ and $t(R_{2i})$. We obtain a collection of $2k$ pairwise vertex-disjoint cycles. The edges of these cycles are contained in $E(P) \cup E(C)$, and therefore their total length is at most 2ℓ. By choosing the k shortest of them we obtain k pairwise disjoint cycles of total length of at most ℓ.

For the final case of $\mathcal{R}'$ comprising $4k$ pairwise parallel chords, We use the same indexing $R_1, \ldots, R_{4k}$ as in the previous paragraph, so that $s(R_i) < s(R_j)$ if $i < j$. Note that now, $s(R_{4k}) < t(R_{4k})$ and $t(R_i) > t(R_j)$ whenever $i < j$. We again construct a cycle C_i for each $i \in [2k]$, by taking the union of R_{2i-1} and R_{2i}, and connecting them by two paths in C, the first between $s(R_{2i-1})$ and $s(R_{2i})$, and the second between $t(R_{2i})$ and $t(R_{2i-1})$. We obtain a collection of $2k$ pairwise vertex-disjoint cycles. The edges of these cycles are contained in $E(P) \cup E(C)$, and therefore their total length is at most 2ℓ and by choosing the k shortest of them we obtain k pairwise disjoint cycles with a total length of at most ℓ. □

Our algorithm for Min-Sum Cycle Packing is based on a graph-theoretical lemma that bounds the number of paths of length at most ℓ in no-instances. We first prove this lemma in the case when $k = 1$, which will be used in the proof of the general case.

Lemma 3.4. *If an edge-weighted n-vertex graph (G, w) does not contain a cycle of length at most ℓ, then the number of paths of length at most $\ell/2$ in G is at most $\binom{n}{2}$.*

Proof. Suppose there are two vertices $a, b \in V(G)$ with two distinct (but not necessarily vertex- or edge-disjoint) a-b-paths P_1 and P_2 in G, both of length at most $\ell/2$. Now, the sum of edge weights in $P_1 \cup P_2$ is at most ℓ and must contain a cycle, that is, G contains a cycle of length at most ℓ. Hence, for each pair $a, b \in V(G)$, there is at most one a-b-path of length at most $\ell/2$, and therefore the total number of paths of length at most $\ell/2$ is at most $\binom{n}{2}$. □

We also note that such bounds on the number of paths of a certain length can be boosted to higher lengths.

Observation 3.5. *Let (G, w) be an edge-weighted graph, so that the number of paths of length at most ℓ in G is at most N. For any integer $c \geq 1$, the number of paths of length at most $c \cdot \ell$ in G is then at most $(2N)^c$.*

Proof. Any path P of length at most $c \cdot \ell$ can be constructed by a sequence $((s_1, t_1), \ldots, (s_c, t_c))$ of vertex pairs, where:

- s_i is the vertex reached after walking a distance of $(i-1) \cdot \ell$ along P (or the next vertex afterwards in case distance $(i-1) \cdot \ell$ ends on some edge), and
- t_i is the vertex reached after walking a distance of $i \cdot \ell$ along P (or the last vertex before).

Note that P is completely determined by the sequence and there is a path of length at most ℓ between each pair (s_i, t_i). Since each such path can be used in either direction, that is, each path forms a tuple (a, b) and a tuple (b, a), the total number of possible paths of length at most $c \cdot \ell$ is at most $(2N)^c$. □

We are now ready to prove our main graph-theoretical lemma, which is at the heart of our algorithm.

Lemma 3.6. *If an edge-weighted n-vertex graph (G, w) does not contain a collection of k pairwise vertex-disjoint cycles of total length at most ℓ, then the number of paths of length at most ℓ in G is in $n^{O(k^5)}$.*

Proof. Let $\tilde{C}$ be a subgraph of G comprising a maximal collection of vertex-disjoint cycles each of length at most ℓ/k. Note that $\tilde{C}$ consists of at most $k-1$ cycles as otherwise G contains k vertex-disjoint cycles of total length at most ℓ. Since $G - V(\tilde{C})$ does not contain a cycle of length at most ℓ/k, Lemma 3.4 implies that the number of paths of length at most $\ell/(2 \cdot k)$ in $G - V(\tilde{C})$ is at most $\binom{n}{2} \leq n^2/2$. Finally, Observation 3.5 allows us to conclude that the number of paths of length at most ℓ in $G - V(\tilde{C})$ is at most $(2 \cdot n^2/2)^{2k} = n^{4k}$. This also implies that the number of paths of length at most ℓ in G whose internal vertices are disjoint from $V(\tilde{C})$ is at most $n^2 \cdot n^{4k} \leq n^{6k}$. Now, let P be a path in G of length at most ℓ.

Claim 3.7. *The number of connected components of $P \cap \tilde{C}$ is at most $128k^4 - 1$.*

Proof of the Claim. Suppose the number of connected components of $P \cap \tilde{C}$ is more than $128k^3 \cdot (k-1)$. Since $\tilde{C}$ is the union of at most $k-1$ cycles of length at most ℓ/k each, by the pigeonhole principle, there is a cycle $C \subseteq \tilde{C}$ such that the number of connected components of $P \cap C$ is more than $128k^3$. Observation 3.1 now implies that P has at least $128k^3$ chords relative to C and Lemma 3.2 therefore states that G contains a collection of k pairwise vertex-disjoint cycles of total length at most ℓ, which is a contradiction. ◁

Now, P can be constructed as the union of the subpaths of P that are internally vertex-disjoint from $V(\tilde{C})$ and the connected components of $P \cap \tilde{C}$, which are paths. Notice that the number of subpaths of P that are internally disjoint with $V(\tilde{C})$ is by Observation 3.1 at most the number of the connected components of $P \cap \tilde{C}$ plus one, that is, at most $128k^4$. On the other hand, the number of paths that are subgraphs of $\tilde{C}$ is at most n^2. Hence, there are at most $(n^2)^{128k^4}$ possible choices for subpaths of P that are connected components of $P \cap \tilde{C}$ and for each of the at most $128k^4$ chords and/or tails of P relative to $\tilde{C}$, there are at most n^{6k} choices for paths which are internally vertex-disjoint from $\tilde{C}$ as noted earlier. The number of paths of length at most ℓ in G is therefore at most:

$$(n^2)^{128k^4} \cdot (n^{6k})^{128k^4} = n^{256k^4} n^{768k^5} \in n^{O(k^5)}.$$

□

Now, we are ready to translate Lemma 3.6 into an algorithm for Min-Sum Cycle Packing. For this, we need the following easy lemma.

Lemma 3.8. *There is an algorithm that, given an edge-weighted n-vertex graph (G, w) and integers ℓ and N, in time $N \cdot n^{O(1)}$ either returns all paths in G of length at most ℓ, or concludes that the number of such paths is more than N.*

Proof. The set of all paths of length 0 is easy to generate, as they are just the vertices of G. Now, let $i \geq 1$ and let $\mathcal{P}_{i-1}$ be the set of paths of length at most $i-1$ in G. Given $\mathcal{P}_{i-1}$, we can generate in time $|\mathcal{P}_{i-1}| \cdot n^{O(1)}$ the set $\mathcal{P}_i$ of all paths of length at most i by first generating $|\mathcal{P}_{i-1}| \cdot n^{O(1)}$ candidates by trying to extend each path by one edge (and also including all paths in $\mathcal{P}_{i-1}$), then filtering out the obtained subgraphs that are not paths, and finally deduplicating the output by bucket sorting. By repeating this until either $i = \ell$ or $|\mathcal{P}_i| > N$, we obtain the desired algorithm. □

We now prove the main theorem of the section.

Theorem 1. *Min-Sum Cycle Packing and Min-Sum Edge-Disjoint Cycle Packing can be solved in $n^{O(k^6)}$ time, where n is the number of vertices in the graph.*

Proof. We demonstrate an algorithm for Min-Sum Cycle Packing and then explain how to adapt it for Min-Sum Edge-Disjoint Cycle Packing.

We first construct a decision algorithm, that given (G, w, k, ℓ), decides whether G contains a collection of k pairwise vertex-disjoint cycles of total length at most ℓ. We first apply the algorithm of Lemma 3.8 with $N = n^{256k^4} n^{1,024k^5} \in n^{O(k^5)}$ (as specified in the proof of Lemma 3.6). If it concludes that the number of paths of length at most ℓ is more than N, then we conclude that the answer is yes. Otherwise, we obtain the collection of all paths of G of length at most ℓ, which has size at most $N \in n^{O(k^5)}$. By trying to extend each with an edge, we obtain the collection of size at most N of all cycles of G of length at most ℓ. Now, we try all possibilities of selecting k cycles from this collection, yielding the running time $\binom{N}{k} \in n^{O(k^6)}$.

Now this algorithm can be turned into an algorithm that finds a collection of k pairwise vertex-disjoint cycles with total length at most ℓ by self-reduction as follows. We repeatedly attempt to remove edges and check if the solution still exists after an edge is removed. These cause only a polynomial (in n) overhead in the running time. This concludes the proof for Min-Sum Cycle Packing.

For Min-Sum Edge-Disjoint Cycle Packing, we observe that if the above algorithm for Min-Sum Cycle Packing concludes that the graph contains k vertex-disjoint cycles of total length at most ℓ, then these k cycles are also edge disjoint. Otherwise, we have that the number of cycles of length at most ℓ is at most N and we can test for each selection of k of them whether they are edge-disjoint in $N^{O(k)}$ time. This concludes the proof. □

Next, we generalize Theorem 1 for Min-Vector Cycle Packing and Edge-Disjoint Min-Vector Cycle Packing.

Theorem 5. *Min-Vector Cycle Packing and Edge-Disjoint Min-Vector Cycle Packing can be solved in $n^{O(k^6)}$ time.*

Proof. We give a decision algorithm. However, it can be turned into an algorithm for finding a solution similarly to the proof of Theorem 1.

Let $(\ell_1, \ldots, \ell_k)$ be the given vector, and assume without loss of generality that $\ell_1 \leq \ell_i$ for all i. Similarly as in the proof of Theorem 5, we apply the combination of Lemmas 3.6 and 3.8 with the parameters k and ℓ_1 to, in time $n^{O(k^5)}$, either conclude that G contains a collection of k pairwise (vertex- and edge-)disjoint cycles of total length at most ℓ_1, or enumerate all (at most $N \in n^{O(k^5)}$) cycles of G of length at most ℓ_1. In the former case, we are done as any such collection gives us a solution, and in the second case, we branch on the cycles to guess the first cycle, delete its vertices (or edges in the case of Edge-Disjoint Min-Vector Cycle Packing) and recursively apply the algorithm to find the $k-1$ other cycles. This branching algorithm runs in a total time of $(n^{O(k^5)})^k = n^{O(k^6)}$. □

4 Lower Bound for Min-Sum Cycle Packing

In this section, we prove our computational lower bound for Min-Sum Cycle Packing. For convenience, we restate our result.

Theorem 2. *Min-Sum Cycle Packing is* W[1]*-hard in subcubic graphs with unit edge weights when parameterized by k.*

Proof. We reduce from Multicolored Clique which is well-known to be W[1]-complete [12]. We recall that the task of the problem is, given a graph G whose set of vertices is partitioned into ℓ sets $V_1, \ldots, V_\ell$ (called *color classes*), to decide whether G has a clique of size ℓ with exactly one vertex from each color class.

Let (G, ℓ) be an instance of Multicolored Clique and let $V_i = \{v_1^i, v_2^i, \ldots, v_{\nu_i}^i\}$ be the set of vertices of color i for each $i \in [\ell]$. Let ν be the maximum number of vertices of any color, let Δ be the maximal degree of G, and let $\gamma = (\nu-1)(3\Delta+1)-1$. For each color $i \in [\ell]$, we create a vertex-selection gadget as follows. We start with 6ν vertices $w_a^{i,1}, u_a^{i,1}, w_a^{i,2}, u_a^{i,2}, w_a^{i,3}$, and $u_a^{i,3}$ for each $a \in [\nu]$. We add an edge $\{w_a^{i,j}, u_a^{i,j}\}$ for each $i \in [\ell]$, $j \in [3]$, and each $a \in [\nu]$. Next, for each $i \in [\ell]$, $j \in [3]$, $a \in [\nu]$, we add $3\Delta - 1$ vertices $v_{a,b}^{i,j}$ where $b \in [3\Delta - 1]$. We add the edge $\{v_{a,b}^{i,j}, v_{a,b+1}^{i,j}\}$ for each $i \in [\ell]$, $j \in [3]$, $a \in [\nu]$, and each $b \in [3\Delta - 2]$. Moreover, we add the edges $\{u_a^{i,j}, v_{a,1}^{i,j}\}$ and $\{v_{a,3\Delta-1}^{i,j}, w_{a+1}^{i,j}\}$ for each $i \in [\ell]$, $j \in [3]$, and $a \in [\nu-1]$ and the edges $\{u_\nu^{i,j}, v_{\nu,1}^{i,j}\}$ and $\{v_{\nu,3\Delta-1}^{i,j}, w_1^{i,j \bmod 3+1}\}$ for each $i \in [\ell]$ and each $j \in [3]$. Note that the entire constructed cycle has length $3\nu(3\Delta+1)$. Finally, we add paths of length 2γ between all pairs of w and u vertices that have distance exactly $(\nu-1)(3\Delta+1)-1 = \gamma$. We call these paths *chords*. Figure 2 gives an illustration of the above construction.

Next, we encode the edges of the input graph. For each color $i \in [\ell]$ and each vertex $v_a^i \in V_i$, arbitrarily assign a distinct number from $[\Delta]$ to each incident edge. Let $f(v_a^i, e)$ be the assigned number. For each edge $e = \{v_a^i, v_b^j\} \in E$, we add paths of length $\lceil \frac{8}{5}\gamma \rceil$ between $v_{a,3f(v_a^i,e)-2}^{i,1}$ and $v_{b,3f(v_b^j,e)-2}^{j,1}$ and between $v_{a,3f(v_a^i,e)-1}^{i,1}$ and $v_{b,3f(v_b^j,e)-1}^{j,1}$. We say that these two paths *encode* the edge e. Finally, we set $k = 3\ell + \binom{\ell}{2}$.

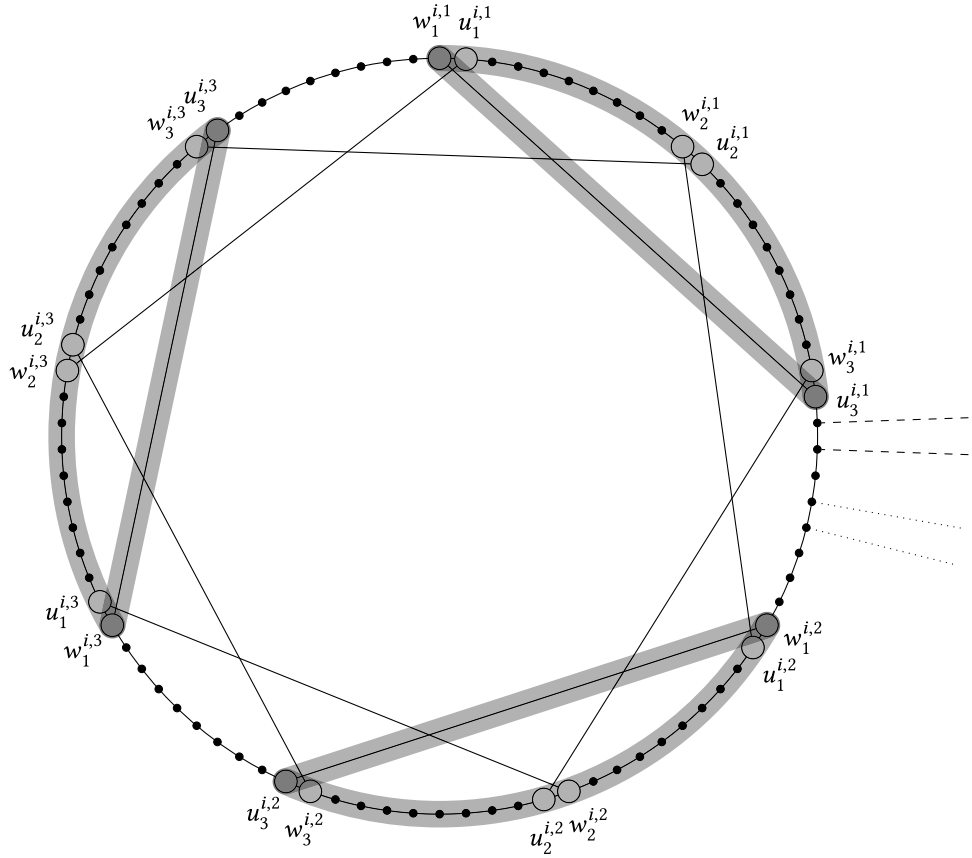

Fig. 2. An example of the vertex-selection gadget with $\nu = \Delta = 3$ for one color i. The chords depict paths of length $2(\nu - 1)(3\Delta - 1) = 32$ and one of the ways to pick three vertex-disjoint cycles using three chords is highlighted. This solution picks vertex v_3^i. Paths encoding two edges incident to v_3^i are shown by dashed and dotted lines, respectively.

Since the above construction takes polynomial time to compute, $k \leq 4\ell^2$, and each vertex has degree at most three, it only remains to prove that the input instance contains a multicolored clique (of size ℓ) if and only if the constructed graph contains k vertex-disjoint cycles of total length at most $L = 9\ell\gamma + \binom{\ell}{2}2(\lceil \frac{8}{5}\gamma \rceil + 1)$. For the backward direction, note that all cycles that use vertices from more than one vertex-selection gadget have length at least $2(\lceil \frac{8}{5}\gamma \rceil + 1) > 3\gamma$. Moreover, all cycles within one vertex-selection gadget that use more than one chord have length at least 4γ. Hence, any solution of total length at most L contains at least 3ℓ cycles of total length at most $9\ell\gamma$. By construction within one vertex-selection gadget, one can pick at most 3 vertex-disjoint cycles of average length at most 3γ and this is only achievable if one picks three equally spaced chords and all vertices from the initial cycle of the vertex-selection gadget with the exception of three sets of v vertices between two consecutive w and u vertices as depicted in Figure 2. We say that such a solution avoiding the v vertices between $w_a^{i,1}$ and $u_{a+1}^{i,1}$ (or $u_1^{i,2}$ if $a = \nu$) *picks* vertex v_a^i. If we select a solution that picks a vertex for each vertex-selection gadget, then it remains to find $\binom{\ell}{2}$ vertex-disjoint cycles that use vertices from at least two vertex-selection gadgets of total length at most $\binom{\ell}{2}2(\lceil \frac{8}{5}\gamma \rceil + 1)$. Since each path between two vertex-selection gadgets is of length $\lceil \frac{8}{5}\gamma \rceil$,

each vertex in a vertex-selection gadget has at most one incident edge leaving the vertex-selection gadget, and two paths between vertex-selection gadgets have adjacent endpoints if and only if they encode an edge in G, it follows that any solution of total length at most L contains $\binom{\ell}{2}$ pairs of paths encoding edges of G. By construction, this corresponds to a set of $\binom{\ell}{2}$ edges between ℓ vertices of different colors, that is, to a multicolored clique (of size ℓ).

For the forward direction, note that if there exists a multicolored clique, then choosing the respective vertex in each vertex-selection gadget and taking the cycles encoding all edges between the chosen vertices results in a set of k vertex-disjoint cycles of total length exactly L. This concludes the proof. □

Since two cycles in subcubic graphs are edge-disjoint if and only if they are vertex-disjoint, we can defer the same hardness result for Min-Sum Edge-Disjoint Cycle Packing.

Corollary 4.1. *Min-Sum Edge-Disjoint Cycle Packing is* W[1]*-hard in subcubic graphs with unit edge weights when parameterized by* k.

5 Packing Shortest Cycles in Planar Graphs

In this section, we consider packings of disjoint shortest cycles in planar graphs and prove Theorems 3 and 4. In Section 5.1, we construct a tree structure of a laminar family of shortest cycles that represent all minimum cycles. In Section 5.2, we prove Theorem 3 and Section 5.3 contains the proof of Theorem 4.

5.1 Laminar Decomposition for Shortest Cycles

In this subsection, we construct a laminar family of disjoint cycles representing all shortest cycles in a planar graph and decompose this family into a tree. Throughout this subsection, we assume that considered graphs are not forests, that is, they have cycles.

We use the following folklore properties of shortest cycles which we prove for completeness. Given two distinct cycles C_1 and C_2 of a graph G with a non-empty intersection, we say that C_1 and C_2 *touch* if $C_1 \cap C_2$ is a path (possibly trivial, that is, having a single vertex).

Lemma 5.1. *Let G be a weighted graph and let C_1 and C_2 be distinct shortest cycles with at least one common vertex. Then:*

- *either C_1 and C_2 touch,*
- *or $V(C_1 \cap C_2) = \{s, t\}$ for distinct s and t at distance $g(G)/2$ in both cycles, and $C_1 = P_1 \cup P_2$ and $C_2 = Q_1 \cup Q_2$ where P_1, P_2, Q_1, and Q_2 are distinct internally vertex-disjoint s-t-paths of length $g(G)/2$.*

Proof. Assume that C_1 and C_2 do not touch. Then C_1 has two distinct internally vertex-disjoint paths P_1 and P_2 such that the endpoints of these paths are in C_2 and the internal vertices and all the edges are outside C_2. Then, the length of one of these paths, say, P_1 is upper-bounded by $g(G)/2$. Let s and t be the endpoints of P_1. Denote by Q_1 and Q_2 two distinct s-t-paths in C_2. Since $S_1 = P_1 \cup Q_1$ and $S_2 = P_1 \cup Q_2$ are cycles, the length of S_1 and S_2 is at least $g(G)$. Therefore, the paths P_1, Q_1, and Q_2 have the same length $g(G)/2$. Also, we have that the length of P_2 is $g(G)/2$. Thus, P_1, P_2, Q_1, and Q_2 are distinct internally vertex-disjoint s-t-paths of length $g(G)/2$. □

We need the following additional notation for plane graphs.

Definition 5.2 (Laminar Family). We say that two cycles C_1 and C_2 in G cross if C_1 has at least one edge in the internal face of C_2 and, symmetrically, C_2 has at least one edge in the internal face of C_1. A family $\mathcal{C}$ of cycles is laminar if cycles in $\mathcal{C}$ do not cross.

Let G be a weighted plane graph. We introduce the following partial order on the family of all shortest cycles of G. For two shortest cycles C_1 and C_2, we write $C_1 \leq C_2$ if every vertex and edge of C_1 is a vertex or an edge of C_2 or is embedded in the internal face of C_2. We also write $C_1 <_v C_2$ if $C_1 \leq C_2$ and $V(C_1) \cap V(C_2) = \emptyset$ (that is, C_1 is completely inside C_2), and we write $C_1 <_e C_2$ if $C_1 \leq C_2$ and $E(C_1) \cap E(C_2) = \emptyset$ (i.e., C_1 and C_2 may share vertices but not edges).

For a cycle C in G, we denote by G_C the subgraph of G composed by the vertices and edges of C and the vertices and edges of G embedded in the internal face of C. We say that G is *clean* if each vertex and each edge of G is included in some shortest cycle. We show that a clean graph always has a facial shortest cycle in the same way as other uncrossable families [36].

Lemma 5.3. *Let G be a clean weighted plane graph. Then, there is an internal face $f \in F(G)$ whose frontier is a shortest cycle.*

Proof. Let C be the shortest cycle that is minimal with respect to the partial order $(\leq)$. We claim that the internal face of C is a face of G. For the sake of contradiction, assume that this is not the case. Then, G either has a vertex v embedded in the internal face of C or an edge e with its endpoints in C which is embedded in the internal face of C. In both cases, because G is clean, there is a cycle C' containing either v or e. Since C is minimal, $C' \not\leq C$. Hence, C and C' cross and C' has either a vertex or an edge in the external face of C. By Lemma 5.1, $V(C \cap C') = \{s, t\}$ for distinct s and t at distance $g(G)/2$ in both cycles, and $C = P_1 \cup P_2$ and $C' = Q_1 \cup Q_2$ where P_1, P_2, Q_1, and Q_2 are distinct internally vertex-disjoint s-t-paths of length $g(G)/2$. Notice that either Q_1 or Q_2 contain v or e. By symmetry, assume that this holds for Q_1. Because P_1, P_2, and Q_1 are distinct internally vertex-disjoint, we have that Q_1 is drawn in the internal face of C. This means that $S = P_1 \cup Q_1$ is a shortest cycle and $S \leq C$. However, this contradicts the assumption that C is minimal. This concludes the proof. □

Let C be a shortest cycle in a weighted plane graph G. We say that C is *splittable* if there are two vertices $s, t \in V(C)$ at distance $g(G)/2$ from each other in C such that G_C has an s-t-path P of length $g(G)/2$ distinct from the two s-t-paths in C; we say that C is *unsplittable*, otherwise. We call s and t *poles*. Notice that for poles s and t on a splittable shortest cycle, the distance between them in G is $g(G)/2$. We need the following observation about splittable cycles.

Lemma 5.4. *Let C be a splittable shortest cycle in a weighted plane graph G with poles s and t. Then, any two distinct shortest s-t-paths in G_C are internally vertex-disjoint and $\{s, t\}$ is a unique pair of poles on C.*

Proof. Suppose that P_1 and P_2 are distinct s-t-paths in G_C of length $g(G)/2$. To show that P_1 and P_2 are internally vertex-disjoint, note that if P_1 and P_2 have a common vertex distinct from s and t then $P_1 \cup P_2$ has a cycle whose length is less than $g(G)$. This proves that any two shortest s-t-paths in G_C are internally vertex-disjoint.

Consider the second property and let P be a shortest s-t-path in G_C that is distinct from the two s-t paths along C. Assume that there is a pair of vertices $\{s', t'\} \neq \{s, t\}$ on C at distance $g(G)/2$ such that there is an s'-t'-path P' of length $g(G)/2$ in G_C that is not a path in C. Note that $\{s', t'\} \cap \{s, t\} = \emptyset$. Otherwise, one of the two s-t-paths in C or one of the two s'-t'-paths in C is strictly shorter than $g(G)/2$. Let Q be the s-t-path in C containing s' and let Q' be the s'-t'-path in C containing s. By the first claim of the lemma, P and Q are internally vertex-disjoint, and, similarly, P' and Q' are internally vertex-disjoint as well. Then, $S = P \cup Q$ and $S' = P' \cup Q'$ are shortest cycles. Notice that $S \cap S'$ contains an s-s'-path in C which is nontrivial (contains at least two vertices) because $s \neq s'$. Also, because P and P' are paths in G_C, the paths have a common vertex v in the internal face of C and, therefore, $S \cap S'$ is not a path. By Lemma 5.1, we have that S and S' intersect in exactly two vertices, implying that $s = s'$; a contradiction. This concludes the proof. □

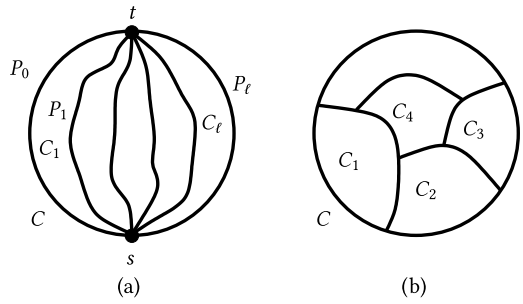

Fig. 3. Construction of $\mathcal{P}_s(C)$ and $\mathcal{C}_s(C)$ for a splittable cycle (a) and construction of $\mathcal{C}_u(C) = \{C_1, C_2, C_3, C_4\}$ for an unsplittable cycle (b).

Let C be a splittable shortest cycle in a weighted plane graph G with poles s and t. We denote by $\mathcal{P}_s(C) = \{P_0, \ldots, P_\ell\}$ the inclusion-maximal family of distinct s-t-paths in G_C. By Lemma 5.4, $\mathcal{P}_s(C)$ is unique. We assume that the paths on $\mathcal{P}_s(C)$ are ordered in the clockwise order from the perspective of s (see Figure 3(a)) and $C = P_0 \cup P_\ell$. For $i \in [\ell]$, we define the cycle $C_i = P_{i-1} \cup P_i$, and set $\mathcal{C}_s(C) = \{C_1, \ldots, C_\ell\}$. We use the following crucial property of splittable shortest cycles.

Lemma 5.5. *Let C be a splittable shortest cycle in a weighted plane graph G. Then, for any shortest cycle S in G_C, either $S = P \cup Q$ for two paths $P, Q \in \mathcal{P}_s(C)$ or $S \le R$ for some cycle $R \in \mathcal{C}_s(C)$.*

Proof. Let S be a shortest cycle in G_C and assume that $S \not\le R$ for any $R \in \mathcal{C}_s(C)$. Suppose that there is $R \in \mathcal{C}_s(C)$ such that S and R cross. Then by Lemma 5.1, $V(S \cap R) = \{s', t'\}$ for distinct s' and t' at distance $g(G)/2$ in both cycles, and $S = P_1 \cup P_2$ and $R = Q_1 \cup Q_2$ where P_1, P_2, Q_1, and Q_2 are distinct internally vertex-disjoint s'-t'-paths of length $g(G)/2$. Because S and R cross, we can assume without loss of generality that the edges and internal vertices of P_1 are embedded in the internal face of R and the edges and internal vertices of P_2 are in the external face of R. If $\{s', t'\} = \{s, t\}$ then we have that P_1 is an s-t-path in G_C internally vertex disjoint with the s-t-paths of $\mathcal{P}_s(C)$ forming R. Then P_1 is internally vertex disjoint with every path of $\mathcal{P}_s(C)$. However, this would contradict the maximality of $\mathcal{P}_s(C)$. Thus, $\{s', t'\} \neq \{s, t\}$. Moreover, note that $\{s', t'\} \cap \{s, t\} = \emptyset$ as otherwise, we would contradict the fact that Q_1 and Q_2 are paths of length $g(G)/2$. But now, we have that s and t are embedded in distinct faces of S. This is impossible because $s, t \in V(C)$ and C is the frontier of the external face of G_C (recall that S cannot contain s or t as it already contains s' and t'). This contradiction shows that S and R do not cross for any $R \in \mathcal{C}_s(C)$.

Since S and R do not cross for any $R \in \mathcal{C}_s(C)$, the definition of $\mathcal{C}_s(C)$ implies that $S = P \cup Q$ for two paths $P, Q \in \mathcal{P}_s(C)$. This concludes the proof. □

Now we deal with unsplittable shortest cycles. Consider an unsplittable shortest cycle C in a weighted plane graph G. We define $\mathcal{C}_u(C)$ to be the set of all shortest cycles of G_C distinct from C, that are maximal cycles of this type with respect to $(\le)$ (see Figure 3(b)). We use the following property of $\mathcal{C}_u(C)$.

Lemma 5.6. *Let C be an unsplittable shortest cycle in a weighted plane graph G. Then $\mathcal{C}_u(C)$ is laminar and for any shortest cycle $S \neq C$ in G_C, $S \le R$ for some cycle $R \in \mathcal{C}_u(C)$.*

Proof. To show that $\mathcal{C}_u(C)$ is laminar, consider distinct $R_1, R_2 \in \mathcal{C}_u(C)$ and assume that R_1 and R_2 cross. By Lemma 5.1, $V(R_1 \cap R_2) = \{s', t'\}$ for distinct s' and t' at distance $g(G)/2$ in both cycles, and $R_1 = P_1 \cup P_2$ and $R_2 = Q_1 \cup Q_2$ where P_1, P_2, Q_1, and Q_2 are distinct internally vertex-disjoint s'-t'-paths of length $g(G)/2$. Because R_1 and R_2 cross, we can assume without loss of generality

that the edges and internal vertices of Q_1 are embedded in the internal face of R_1 and the edges and internal vertices of P_2 are in the external face of R_2. Consider $R = P_1 \cup Q_2$. We have that $R_1 \leq R$ and $R_2 \leq R$. Recall that R_1 and R_2 are maximal with respect to $(\leq)$ shortest cycles distinct from C. Since $R \notin \mathcal{C}_u(C)$, $R = C$. However, we have that $s, t \in V(C)$ and G_C has four internally vertex-disjoint s-t-paths of length $g(G)/2$. This means that C is splittable and contradicts our assumptions. Thus, $\mathcal{C}_u(C)$ is laminar.

To see the second claim, it is sufficient to observe that already by the definition of $\mathcal{C}_u(C)$, for any shortest $S \neq C$ in G_C, $S \leq R$ for some maximal (with respect to $(\leq)$) shortest cycle of G_C that is distinct from C. This concludes the proof. □

Now we are ready to construct a tree representing a laminar family of shortest cycles.

Let G be a clean planar weighted graph. Consider an arbitrary embedding of G in the plane. By Lemma 5.3, there is a face f in this embedding whose frontier is a shortest cycle C. Then, there is another planar embedding of G such that C is a facial cycle of the external face of the embedded graph. We fix this embedding and construct the rooted tree $\mathcal{T}(G)$, called the LSCT, whose nodes are shortest cycles.

We start constructing $\mathcal{T}(G)$ from the facial cycle C of the external face which is set to be the root. Then, we construct $\mathcal{T}(G)$ by the recursive analysis of shortest cycles C that are current leaves of the already constructed tree.

- If C is a facial cycle for an internal face then we set C to be a leaf of $\mathcal{T}(G)$.
- If C a splittable cycle then we construct $\mathcal{C}_s(C)$ and set the cycles of $\mathcal{C}_s(C)$ to be the children of C in $\mathcal{T}(G)$; we say that C is an S-node in this case.
- If C is an unsplittable cycle then we construct $\mathcal{C}_u(C)$ and set the cycles of $\mathcal{C}_u(C)$ to be the children of C in $\mathcal{T}(G)$; we say that C is a U-node.

The properties of LSCTs are summarized in the following lemma.

Lemma 5.7. *Suppose that the LSCT $\mathcal{T}(G)$ is constructed for a clean planar weighted graph G. Then the nodes of $\mathcal{T}(G)$ form a laminar family of shortest cycles such that*:

(i) *for any two nodes C and C' of $\mathcal{T}(G)$, $C \leq C'$ in the planar embedding used in the construction of $\mathcal{T}(G)$ if and only if C is a descendant of C' in $\mathcal{T}(G)$,*
(ii) *for any shortest cycle C of G, either C is a node of $\mathcal{T}(G)$ or there is an S-node R of $\mathcal{T}(G)$ and two paths $P, Q \in \mathcal{P}_s(R)$ such that $C = P \cup Q$, and*
(iii) *the facial shortest cycles of G distinct from the root of $\mathcal{T}(G)$ are the leaves of $\mathcal{T}(G)$.*

Furthermore, the LSCT can be constructed in polynomial time.

Proof. Properties (i)–(iii) follow directly from Lemmas 5.5 and 5.6, and the construction of $\mathcal{T}(G)$. To prove that $\mathcal{T}(G)$ can be constructed in polynomial time, recall that a planar embedding of G can be found in linear time [28]. Then we can consider all the faces and find a face f whose frontier is a shortest cycle C. This allows us to construct an embedding of G where the frontier of the external face is a shortest cycle, in polynomial time. Then, following the steps of the algorithm, for each C that is a current leaf of the already constructed tree, we verify whether C is splittable or not. This can be done in polynomial time using Dijkstra's algorithm [15]. If C is splittable, then we construct the unique family of paths $\mathcal{P}_s(C)$ by greedily finding shortest s-t-paths. This allows us to construct $\mathcal{C}_s(C)$ in polynomial time. If C is unsplittable, then we can, for example, list all shortest cycles in G_C using the well-known fact that the number of such cycles is quadratic in the number of vertices and they can be enumerated in polynomial time (e.g., by using the algorithm of Karger and Stein [29] for the enumeration of all minimum cuts in the dual graph). Then, we can find all

maximal shortest cycles with respect to ($\preceq$) distinct from C in G_C and obtain $\mathcal{C}_u(C)$ in polynomial time. Summarizing, we conclude that the overall running time is polynomial. This concludes the proof. □

We conclude this section with some structural observations about solutions of Shortest Cycle Packing and Edge-Disjoint Shortest Cycle Packing on planar graphs. Let G be a clean weighted planar graph and let k be a positive integer. Recall that we assume that G has a fixed planar embedding with the frontier of the external face being a shortest cycle. For LSCT $\mathcal{T}(G)$, we define special cycle packings.

Definition 5.8. A packing $\mathcal{C} = \{C_1, \ldots, C_k\}$ of k (vertex/edge)-disjoint shortest cycles is *$\mathcal{T}$-maximal* if:

(i) the number of facial cycles of internal faces in the packing is maximum, and
(ii) there is no packing $\mathcal{C}' = \{C'_1, \ldots, C'_k\}$ distinct from $\mathcal{C}$ such that $C'_i \preceq C_i$ for all $i \in [k]$.

Our algorithm uses the following property.

Lemma 5.9. *Suppose that the LSCT $\mathcal{T}(G)$ is constructed for a clean weighted planar graph G and assume that G is embedded in the plane according to the construction of $\mathcal{T}$. Let $\mathcal{C}$ be a $\mathcal{T}$-maximal packing of k (vertex/edge)-disjoint shortest cycles. Then, $\mathcal{C}$ is laminar and for each $C \in \mathcal{C}$, either C is a facial cycle of an internal face or there is a $C' \in \mathcal{C}$ such that $C' \preceq C$.*

Proof. If $\mathcal{C} = \{C_1, \ldots, C_k\}$ is a packing of vertex-disjoint cycles, then $\mathcal{C}$ is trivially laminar. Suppose that $\mathcal{C}$ is a packing of edge-disjoint cycles and assume that there are distinct $i, j \in [k]$ such that C_i and C_j cross. Then by Lemma 5.1, $V(C_i \cap C_j) = \{s, t\}$ for distinct s and t at distance $g(G)/2$ in both cycles, and $C_i = P_1 \cup P_2$ and $C_j = Q_1 \cup Q_2$ where P_1, P_2, Q_1, and Q_2 are distinct internally vertex-disjoint s-t-paths of length $g(G)/2$. Since the cycles cross, we can assume without loss of generality that the edges and internal vertices of Q_1 are in the internal face of C_i and the edges and internal vertices of P_2 are in the internal face of C_j. Let $C'_i = P_1 \cup Q_1$ and $C'_j = P_2 \cup Q_2$. Notice that C_i and C_j are not facial cycles and it holds that $C'_i \preceq C_i$ and $C'_j \preceq C_j$. Consider $\mathcal{C}'$ obtained by the replacement of C_i and C_j with C'_i and C'_j, respectively. We have that $\mathcal{C}' \neq \mathcal{C}$ is a packing of k edge-disjoint shortest cycles such that the number of facial cycles is at least as large as in $\mathcal{C}$. Moreover, because $C'_i \preceq C_i$ and $C'_j \preceq C_j$, the existence of $\mathcal{C}'$ contradicts that $\mathcal{C}$ is a $\mathcal{T}$-maximal packing. Thus, $\mathcal{C}$ is laminar.

To prove the second claim, suppose that $C_i \in \mathcal{C}$ is not a facial cycle of an internal face of G for some $i \in [k]$ and there is no $j \in [k]$ distinct from i such that $C_j \preceq C_i$. Consider G_{C_i}. Then by Lemma 5.3, there is an internal face f of G_{C_i} whose frontier is a shortest cycle. Denote by C'_i such a cycle. We have that $C'_i \neq C_i$ and $C'_i \preceq C_i$. Furthermore, because $\mathcal{C}$ is laminar by the already proved first claim, $\mathcal{C}'$ obtained from $\mathcal{C}$ by the replacement of C_i with C'_i is a packing of disjoint cycles. However, the number of facial cycles of internal faces in $\mathcal{C}'$ is bigger than the number of such cycles in $\mathcal{C}$. This contradicts the assumption that $\mathcal{C}$ is $\mathcal{T}$-maximal. This proves the second claim and completes the proof of the lemma. □

5.2 FPT Algorithm for Shortest Cycle Packing

In this subsection, we prove Theorem 3 which we restate here.

Theorem 3. *Shortest Cycle Packing is solvable in time $k^{O(k)} \cdot n^{O(1)}$ on planar graphs with n vertices.*

Proof. Let (G, w, k) be an instance of Shortest Cycle Packing where G is a planar graph. If G is a forest, then we have a trivial no-instance. Thus, we assume that this is not the case. Then,

we preprocess G and delete every edge and every vertex that is not included in a shortest cycle as these edges and vertices are irrelevant for our problem. From now on, we assume that G is clean.

Next, we construct a planar embedding of G such that the frontier of the external face is a shortest cycle. From this point, we assume that G is a plane graph. We then construct the LSCT $\mathcal{T}(G)$. Our aim is to find a $\mathcal{T}$-maximal solution by using the properties given by Lemma 5.9.

By Lemma 5.7 (ii), for any shortest cycle C of G, either C is a node of $\mathcal{T}(G)$ or there is an S-node R of $\mathcal{T}(G)$ and two paths $P, Q \in \mathcal{P}_s(R)$ such that $R = P \cup Q$ where $C \neq P \cup Q$ and $P \cup Q \notin \mathcal{C}_s(R)$. If C is a shortest cycle of the second type, then we say that C is a *non-tree* cycle.

In the following, we wish to achieve the property that each cycle of a solution is a node of the tree. Since this may not always be the case, we ensure this property by modifying the tree at selected nodes.

Let C be an S-node of $\mathcal{T}(G)$. We say that C is *large* if $|\mathcal{C}_s(C)| \geq 3k + 3$ and C is *small*, otherwise. We prove that for large S-nodes, we already have the desired property.

Claim 5.10. *Let $\mathcal{C}$ be a $\mathcal{T}$-maximal solution to (G, w, k). Then for any large S-node C of $\mathcal{T}(G)$, if $\mathcal{C}$ contains a cycle $R = P \cup Q$ for two distinct paths $P, Q \in \mathcal{P}_s(C)$ then either $R = C$ or $R \in \mathcal{C}_s(C)$.*

Proof of the Claim. Let C be a large S-node of $\mathcal{T}(G)$ with poles s and t. Let $\mathcal{P}_s(C) = \{P_0, \ldots, P_\ell\}$ and $\mathcal{C}_s(C) = \{C_1, \ldots, C_\ell\}$. We assume that the paths in $\mathcal{P}_s(C)$ are ordered in the clockwise order from the perspective of pole s and $C_i = P_{i-1} \cup P_i$ for each $i \in [\ell]$, as shown in Figure 3(a). Suppose that there is $R \in \mathcal{C}$ distinct from C such that $R = P_i \cup P_j$ for $i, j \in [0, \ell]$ with $i < j - 1$. Notice that R is not a facial cycle.

Consider $\mathcal{S} = \mathcal{C} \setminus \{R\}$. By the fact that $s, t \in V(R)$, Lemma 5.7, and the laminarity of $\mathcal{C}$, we have that for every $S \in \mathcal{S}$, $s, t \notin V(S)$ and either S has no edges and vertices in the internal face of C or there is $i \in [\ell]$ such that $S \leq C_i$. Because C is large, $\ell \geq 3k + 3$. Then by the pigeon hole principle, there is $i \in [2, \ell - 1]$ such that there is no $S \in \mathcal{S}$ with $S \leq C_{i-1}, C_i, C_{i+1}$. This means that $\mathcal{S}' = \mathcal{S} \cup \{C_i\}$ is a packing of k vertex-disjoint shortest cycles. By Lemma 5.3, there is a facial shortest cycle R' of an internal face of G_{C_i}. As there is no $S \in \mathcal{S}$ with $S \leq C_i$, we have that $\mathcal{C}' = \mathcal{S} \cup \{R'\}$ is a packing of k vertex-disjoint shortest cycles. However, $\mathcal{C}'$ is obtained from $\mathcal{C}$ by replacing a non-facial cycle R with a facial cycle R'. This contradicts the assumption that $\mathcal{C}$ is a $\mathcal{T}$-maximal solution and proves the claim. ◁

By Claim 5.10, it remains to deal with non-tree cycles formed by two paths of $\mathcal{P}(C)$ for small S-nodes. For this, we apply the *random separation* technique [8, 12]. For simplicity, we give a randomized Monte Carlo procedure and explain how to derandomize this step in the conclusion of the proof of the theorem.

Let $\mathcal{P} = \bigcup_C \mathcal{P}_s(C)$ where the union is taken over all small S-nodes C of $\mathcal{T}(G)$, that is, $\mathcal{P}$ is the family of all paths that may be parts of non-tree cycles in a solution. We color the paths of $\mathcal{P}$ randomly by two colors *red* and *blue* in such a way that a path $P \in \mathcal{P}$ is colored red with probability $\frac{2}{3k+3}$ and P is colored blue with probability $\frac{3k+1}{3k+3}$. Consider a solution $\mathcal{C}$ such that if $\mathcal{C}$ contains a non-tree cycle $R = P \cup Q$ for two distinct paths $P, Q \in \mathcal{P}_s(C)$ for a small S-node C of $\mathcal{T}(G)$ then (i) P and Q are colored red and (ii) all other paths in $\mathcal{P}_s(C)$ are blue. We say that such a solution $\mathcal{C}$ is *colorful*.

To see the reason behind the coloring, assume that $\mathcal{C}$ is a $\mathcal{T}$-maximal solution to the considered instance. Let $R_1, \ldots, R_\ell$ be the non-tree cycles in $\mathcal{C}$ formed by pairs of paths from $\mathcal{P}_s(C_1), \ldots, \mathcal{P}_s(C_\ell)$ for small S-nodes $C_1, \ldots, C_\ell$ of $\mathcal{T}(G)$. Let $\mathcal{P}' = \bigcup_{i=1}^{\ell} \mathcal{P}_s(C_i)$. Clearly, $\ell \leq k$. The probability that all 2ℓ paths from $\mathcal{P}'$ forming the non-tree cycles $R_1, \ldots, R_\ell$ are red is at least $\left(\frac{2}{3k+3}\right)^{2\ell} \geq \left(\frac{2}{3k+3}\right)^{2k}$. Since C_i is a small S-node for $\mathcal{T}(G)$, $|\mathcal{P}_s(C_i)| \leq 3k + 3$ for each $i \in [\ell]$. Hence, the total number of

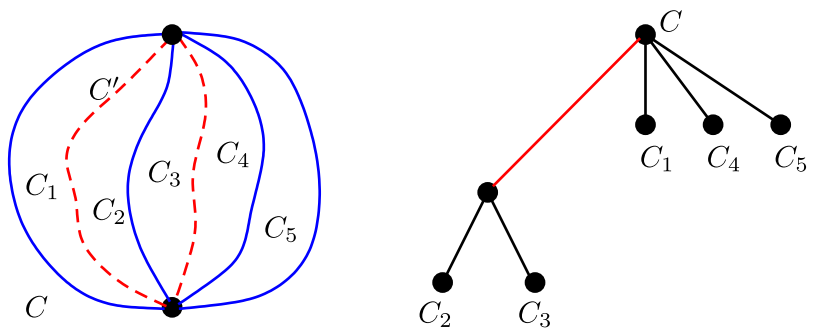

Fig. 4. Construction of $\mathcal{T}^*$. In the tree on the right side of the figure, the parent node of C_2 and C_3 corresponds to the cycle C' (red dashed lines) on the left.

paths in $\mathcal{P}'$ that are not parts of non-tree cycles is at most $(3k+1)\ell \leq (3k+1)k$. The probability that all these paths are colored blue is at least:

$$\left(\frac{3k+1}{3k+3}\right)^{(3k+1)k} \geq \left(1-\frac{2}{3k+3}\right)^{\frac{3k+3}{2}2k} \geq \left(\frac{8}{27}\right)^{2k}.$$

Thus, the probability that all paths in $\mathcal{P}'$ forming non-tree cycles are red and all other paths are blue is at least $\left(\frac{16}{81(k+1)}\right)^{2k}$. This implies that if we try $N = \left(\frac{81(k+1)}{16}\right)^{2k} = k^{O(k)}$ random colorings, then the probability that the desired property that all paths in $\mathcal{P}'$ forming non-tree cycles are red and all other paths are blue is not fulfilled for any of the colorings is at most $\left(1-\frac{1}{N}\right)^N \leq \frac{1}{e}$.

From now on, we assume that a coloring of the paths of $\mathcal{P}$ is given such that if there is a solution, then a $\mathcal{T}$-maximal solution is colorful.

We construct the tree $\mathcal{T}^*$ from $\mathcal{T}(G)$ by splitting selected small S-nodes. For every small S-node C such that there are two red (dashed) paths in $\mathcal{P}_s(C)$ forming a non-tree cycle C' and all other paths in $\mathcal{P}_s(C)$ are blue (solid), we do the following (see Figure 4):

—create a new node C' and make it a child of C, and
—make every cycle $S \in \mathcal{C}_s(C)$ such that $S \leq C'$ a child of C'; the other cycles in $\mathcal{C}_s(C)$ remain to be children of C.

Observe that every node of $\mathcal{T}(G)$ is a node of $\mathcal{T}^*$ and, by Lemma 5.7 and the construction of $\mathcal{T}^*$, we have that the nodes of $\mathcal{T}^*$ form a laminar family of shortest cycles such that the following two properties hold.

Property (i): For any two nodes C and C' of $\mathcal{T}^*$, $C \leq C'$ in the planar embedding of G if and only if C is a descendant of C' in $\mathcal{T}^*$.

Property (ii): The facial shortest cycles of G distinct from the root of $\mathcal{T}^*$ are the leaves of $\mathcal{T}^*$.

Furthermore, if a colorful $\mathcal{T}$-maximal solution to (G, w, k) exists, then each cycle of the solution is a node of $\mathcal{T}^*$. We use these properties to construct a recursive branching algorithm that finds a solution if a colorful $\mathcal{T}$-maximal solution exists.

The algorithm takes G, k, and $\mathcal{T}^*$ as its parameters; these parameters are modified in the recursive calls.

In the first step, we construct the map graph M with G as its planar embedding whose vertices are the faces of G with their frontiers being shortest facial cycles. Then we call the algorithm from

Proposition 2.2 to check whether M has an independent set of size k. If such a set exists then the frontiers of the faces of G in the independent set compose a packing of k vertex-disjoint shortest cycles. Thus, we can conclude that (G, w, k) is a yes-instance and stop. Assume that this is not the case.

Because there is no packing of k vertex-disjoint shortest cycles that contains only facial cycles, by Lemma 5.9, we conclude that if there is a colorful $\mathcal{T}$-maximal solution then any solution contains a non-facial cycle C and a facial cycle C' such that $C' <_v C$. Moreover, by properties (i) and (ii) of $\mathcal{T}^*$, there is a node C of $\mathcal{T}^*$ in the solution such that (a) C is a non-facial cycle, (b) there is a facial cycle C' in the solution such that $C' <_v C$, and moreover, we can choose C such that (c) any cycle $C'' \neq C$ in the solution such that $C'' \leq C$ is facial. Our goal is to identify C among all those shortest cycles that satisfy conditions (a)–(c),

For this, we first mark the nodes of $\mathcal{T}^*$ that are candidates to be C. Namely, we mark every node C of $\mathcal{T}^*$ such that there is a facial shortest cycle, that is, a leaf C' of $\mathcal{T}^*$, with the property that $C' <_v C$. If no node was marked, then we conclude that there is no colorful $\mathcal{T}$-maximal solution and so, we return that (G, w, k) is a no-instance and stop. We now assume that this is not the case and consider the subgraph $\mathcal{T}^\alpha$ of $\mathcal{T}^*$ induced by the marked nodes. Notice that $\mathcal{T}^\alpha$ is a subtree of $\mathcal{T}^*$ with the same root as $\mathcal{T}^*$, since any ancestor of a marked node is marked by property (i) of $\mathcal{T}^*$.

Denote by $\mathcal{L}$ the set of leaves of $\mathcal{T}^\alpha$ and let $\mathcal{U}$ be the set of nodes of $\mathcal{T}^\alpha$ that includes the root, the nodes of $\mathcal{L}$, and all the nodes with at least two children. Notice that because of Lemma 5.3, for each $C \in \mathcal{L}$, there is a facial shortest cycle $C' <_v C$. Then by the laminarity of the cycles of $\mathcal{T}^\alpha$ and property (i) of $\mathcal{T}^\alpha$, we have that $|\mathcal{L}| < k$. Otherwise, the map graph M would have an independent set of size k. However, this was already checked and ruled out. Since $|\mathcal{L}| < k$, we have that $|\mathcal{U}| \leq 2|\mathcal{L}| - 1 \leq 2k - 3$. Denote by $\mathcal{S}$ the set of all nontrivial S_1-S_2-paths in $\mathcal{T}^\alpha$ with their endpoints S_1 and S_2 in $\mathcal{U}$ such that the internal vertices have degree two in $\mathcal{T}^\alpha$. Note that $|\mathcal{S}| \leq |\mathcal{U}| - 1 \leq 2k - 4$.

For each node C in $\mathcal{T}^\alpha$, consider the map graph M_C such that G_C is its planar embedding and the vertices of M_C are those faces of G_C with shortest cycles as their frontiers and such that for any of their facial cycles R, we have that $R <_v C$, that is, all these faces are in the internal face of C and the facial cycles do not touch C.

Recall that we aim to find the non-facial cycle C satisfying conditions (a)–(c) stated above. For this, we guess the path in $\mathcal{S}$ containing C and the number r of facial cycles C' in the solution such that $C' <_v C$. This is done by branching on all possible choices of a path in $\mathcal{S}$ and a positive integer $r \leq k - 1$.

Assume that r and an S_1-S_2-path $P \in \mathcal{S}$ are given. Since $S_1, S_2 \in \mathcal{U}$, we can assume without loss of generality that S_2 is a descendant of S_1 in $\mathcal{T}^\alpha$. For each node S in P, we check whether there is a packing of r vertex-disjoint facial cycles C' such that $C' <_v S$. This is done by calling the algorithm from Proposition 2.2 for M_S. If such a node does not exist, then we conclude that either C does not lie on P or the solution does not have r vertex-disjoint facial cycles C' such that $C' <_v C$. So, we discard this choice of P and r. On the other hand, if we found a cycle S and r vertex-disjoint facial cycles C' such that $C' <_v S$ for $r = k - 1$, then these r facial shortest cycles together with S compose a packing of k vertex-disjoint shortest cycles. Thus, we return yes and stop. Otherwise, among all nodes S with the above property (lying on P and having r vertex-disjoint facial cycles C' such that $C' <_v S$), we select the node S^* closest to S_2 in P.

To argue that the choice of S^* is feasible in the sense that if the guess of the path containing C and r was correct, then we can take $C = S^*$, assume that C is a non-facial cycle in a hypothetical colorful $\mathcal{T}$-maximal solution $\mathcal{C}$ such that C is a node of P, conditions (a)–(c) are fulfilled, and there are exactly r facial cycles C' in the solution with $C' <_v C$. We additionally assume that $\mathcal{C}$ is a solution where C is at a minimum distance from S_2 in $\mathcal{T}^*$. Let $\mathcal{C}' \subseteq \mathcal{C}$ be the set of all the $r + 1$ cycles C' in the solution with $C' \leq C$. By the choice of S^*, $S^* \leq C$ and G_{S^*} has a packing

$\mathcal{C}''$ of $r+1$ vertex-disjoint shortest cycles that includes S^* and r facial cycles $C' <_v S^*$. Consider $\mathcal{C}^* = (\mathcal{C} \setminus \mathcal{C}') \cup \mathcal{C}''$, that is, we replace the cycles of $\mathcal{C}'$ with the cycles of $\mathcal{C}''$. Because $\mathcal{C}$ is laminar and $S^* \leq C$, we obtain that $\mathcal{C}^*$ is a packing of k vertex-disjoint cycles. Because we choose $\mathcal{C}$ where C is at a minimum distance from S_2 in $\mathcal{T}^\alpha$, we obtain that $S^* = C$. This means that the choice of S^* is feasible if P and r were correctly guessed.

Next, we call our algorithm recursively. Since we would like to avoid rebuilding $\mathcal{T}^*$ because we are looking for a colorful solution defined for $\mathcal{T}^*$, we construct new parameters G', k', and $(\mathcal{T}^*)'$ as follows:

- construct G' from G by deleting the vertices and edges of G that are embedded in the internal face of S^*,
- set $k' = k - r$, and
- construct $(\mathcal{T}^*)'$ from $\mathcal{T}^*$ by deleting the nodes that are proper descendants of S^* (i.e., descendants of S^* distinct from S^*) in $\mathcal{T}^*$.

Observe that because we do not delete S^*, we can use $(\mathcal{T}^*)'$ to represent the shortest cycles in the obtained graph G'.

Then, we call the algorithm for G', k', and $(\mathcal{T}^*)'$. If the algorithm returns yes, then we conclude that (G, w, k) is a yes-instance and stop. If the algorithm returns no, then we discard the current choice of P and r. To complete the description of the branching algorithm, we observe that we branch on at most $2k-4$ paths P in $\mathcal{S}$ and at most $k-1$ choices of r. If the algorithm fails to find a solution for all choices, then we conclude that there is no solution and return no.

Since $r \geq 1$, we have that the depth of the recursion is at most k, that is, the algorithm is finite. Notice that if the algorithm concludes that there is a packing $\mathcal{C}$ of k vertex-disjoint shortest cycles, then it may happen that $\mathcal{C}$ is not a $\mathcal{T}$-maximal solution. However, it is sufficient for us that $\mathcal{C}$ is a solution to (G, w, k). From the other side, if (G, w, k) is a yes-instance, then the instance has a $\mathcal{T}$-maximal solution. Hence, if there is a colorful solution of this type, then the algorithm returns yes. This concludes the correctness proof for the branching algorithm.

Recall that we call this algorithm for at most $N = \left(\frac{81(k+1)}{16}\right)^{2k}$ random colorings of $\mathcal{P}$. If for one of the colorings, we obtain that (G, w, k) is a yes-instance then we return yes and stop. Otherwise, if we fail to find a solution for all colorings, then we return no. The algorithm may return a false negative answer but the probability of this event is at most $\frac{1}{e}$. This concludes the description and the correctness proof of the algorithm.

To evaluate the running time, note that we can verify whether G is a forest, compute the girth, and delete the vertices and edges that are not included in shortest cycles in polynomial time by Dijkstra's algorithm [15]. Then, $\mathcal{T}(G)$ and the corresponding planar embedding of G can be constructed in polynomial time by Lemma 5.7. Given $\mathcal{T}(G)$, it is straightforward to construct the set of paths $\mathcal{P}$ in polynomial time. Then, for each random coloring of $\mathcal{P}$, it is also easy to construct $\mathcal{T}^*$ in polynomial time. Then, we call our branching algorithm.

For the evaluation of the running time of the branching algorithm, we note the map graph M can be constructed in polynomial time for the given planar embedding of G. The algorithm from Proposition 2.2 runs in $2^{O(k)} \cdot n^2$ time. If the algorithm reports that M has no independent set of size k, we construct $\mathcal{T}^\alpha$ from $\mathcal{T}^*$ and this can be done in polynomial time. Further, we branch on at most $2k-4$ choices of $P \in \mathcal{S}$ and at most $k-1$ choices of r, that is, we have $O(k^2)$ possibilities. Because P has at most n nodes, we call the algorithm from Proposition 2.2 at most n times. Thus, we can find the cycle C^* (if exists) in $2^{O(k)} \cdot n^3$ time. Then we call our branching algorithm recursively for G', k', and $(\mathcal{T}^*)'$ which can be easily constructed in polynomial time. Since the depth of the

recursion tree is at most k, we obtain that the total running time of the branching algorithm is $k^{O(k)} \cdot n^{O(1)}$.

We run the branching algorithm for at most $N = \left(\frac{81(k+1)}{16}\right)^{2k} = k^{O(k)}$ random colorings. Then the total running time is $k^{O(k)} \cdot n^{O(1)}$. Taking into account that the running time of the previous steps is polynomial, the overall running time is $k^{O(k)} \cdot n^{O(1)}$.

The described algorithm is a randomized Monte Carlo algorithm with one-sided error due to applying the random separation technique. More precisely, we consider random colorings of the paths in the set $\mathcal{P} = \bigcup_C \mathcal{P}_s(C)$ where the union is taken over all small S-nodes C of $\mathcal{T}(G)$. Notice that we use a non-uniform probability distribution. To derandomize the algorithm, one can use the results of Fomin et al. [20] tailored for dealing with random colorings of this type. More precisely, we can replace random colorings by *separating collections*; we refer to Fomin et al. [20] for the definition of separating collections and the details of their construction. The obtained deterministic algorithm has a slightly worse running time. However, the overall running time still can be written as $k^{O(k)} \cdot n^{O(1)}$. This concludes the proof. □

5.3 Kernelization for Edge-Disjoint Shortest Cycle Packing

In this subsection, we prove Theorem 4 which we restate here.

Theorem 4. *Edge-Disjoint Shortest Cycle Packing on planar graphs admits a polynomial kernel such that the output graph has $O(k^2)$ vertices.*

Proof. Let (G, w, k) be an instance of Edge-Disjoint Shortest Cycle Packing where G is a planar graph. We assume that G is clean, as we can safely remove all vertices and edges that are not part of any shortest cycle. If the graph is empty, we return a trivial no-instance. Otherwise, G is clean and non-empty, therefore by Lemma 5.3, there is also an internal face f whose frontier is a shortest cycle, in some fixed planar embedding of G. We consider another embedding of G where f is the outer face, and compute the LSCT $\mathcal{T}(G)$ as described in the proof of Theorem 3.

We first bound the number of leaves in $\mathcal{T}(G)$.

Claim 5.11. *If $\mathcal{T}(G)$ has at least $4k$ leaves, then (G, w, k) is a yes-instance.*

Proof of the Claim. Consider the graph G_F, where the vertex set is the set of shortest cycles in G that are found in the leaves of $\mathcal{T}(G)$, and a pair of vertices is adjacent if and only if the respective cycles share edges. Since the shortest cycles in the leaves of $\mathcal{T}(G)$ are frontiers of internal faces in the embedding of G, G_F is a subgraph of the dual graph G^* of G, and is therefore planar. By Observation 2.1, a planar graph with at least $4k$ vertices has an independent set of size k. The vertices of this independent set in G_F correspond to k edge-disjoint shortest cycles in G, which proves the claim. ◁

By Claim 5.11, if there are at least $4k$ leaves in $\mathcal{T}(G)$, we stop and return a trivial yes-instance. Thus, in the following we assume that $\mathcal{T}(G)$ has less than $4k$ leaves.

For a shortest cycle C in G, we say that its *extension* is any shortest cycle C' with $C <_e C'$, and we say that C' is its *lowest extension* if C' is an extension of C and for every other extension C'' of C, it holds that $C' \leq C''$.

Claim 5.12. *Every shortest cycle C in G that has at least one extension, has a unique lowest extension.*

Proof of the Claim. Observe that for any shortest cycles $C' \leq C''$ in G, if $C <_e C'$, then $C <_e C''$. Consider first extensions of C that are cycles in $\mathcal{T}(G)$. By Lemma 5.7, all cycles C' of $\mathcal{T}(G)$ with $C \leq C'$ form a rooted subpath in $\mathcal{T}(G)$; by the above, the set of extensions of C in $\mathcal{T}(G)$ forms a rooted subpath of this path. If the subpath is empty, then the root cycle itself shares edges

with C, in which case C has no extension. Otherwise, let C^* be the lowest cycle on this subpath; for each other extension C' in $\mathcal{T}(G)$, $C^* \leq C'$. If C^* is a U-node, by Lemma 5.6 $C \leq C'$ for some $C' \in \mathcal{C}_u(C^*)$, and for every other shortest cycle C'' in G_{C^*} with $C \leq C''$, it holds that $C'' \leq C'$. Since $C' \in \mathcal{T}(G)$ is not an extension of C, there are no other extensions in G_{C^*} and C^* is indeed the unique lowest extension.

Now, consider the case where C^* is an S-node. Let s, t be the poles of C^* and let $\mathcal{P}_s(C) = \{P_0, \ldots, P_\ell\}$ be the inclusion-wise maximal family of s-t-paths in $G_{C'}$ as in the construction of $\mathcal{C}_s(C^*)$. By Lemma 5.5, either there exists $C' \in \mathcal{C}_s(C^*)$ such that $C \leq C'$, or $C = P_i \cup P_j$ for some $i < j \in [\ell]_0$. In the latter case, observe that $i > 0$ and $j < \ell$ since C shares no edges with C^*, and all non-tree cycles in G_{C^*} that are extensions of C are of the form $P_{i'} \cup P_{j'}$ for some $i' < i$ and $j < j' \leq \ell$. Hence, in this case $P_{i-1} \cup P_{j+1}$ is the lowest extension of $P_i \cup P_j$. Otherwise, if $C \leq C'$ for some $C' \in \mathcal{C}_s(C^*)$, then C' shares edges with C by the choice of C^* as $C' \in \mathcal{T}(G)$. Let $h \in [\ell]$ be such that $C' = P_{h-1} \cup P_h$, let i be the smallest index in $\{h-1, h\}$ such that P_i shares edges with C, and let j be the largest. By the same argument as in the previous case, $P_{i-1} \cup P_{j+1}$ is the lowest extension of C. ◁

Due to Claim 5.12, for every shortest cycle C in G with at least one extension, we denote by $L(C)$, the unique lowest extension of C. We say that the *extension chain* $\mathcal{U}(C)$ of C is the sequence $C_0, C_1, \ldots, C_r$ of shortest cycles in G such that $C_0 = C$, $C_i = L(C_{i-1})$ for each $i \in [r]$, and the cycle C_r has no extension. Notice that, given $\mathcal{T}(G)$, $\mathcal{U}(C)$ can be found in polynomial time by tracing the path from C to the root in the tree.

We now construct the set of *marked* cycles $\mathcal{C}_M$, which is a subset of all shortest cycles of G. We start by describing the set of *base* cycles $\mathcal{C}_B \subseteq \mathcal{C}_M$, which contains:

—every leaf cycle of $\mathcal{T}(G)$,
—every U-node that has at least two children, and
—every S-node together with the cycles $P_0 \cup P_{\ell-1}$ and $P_1 \cup P_\ell$, where $\{P_0, \ldots, P_\ell\} = \mathcal{P}_s(C)$.

If there exists a cycle $C \in \mathcal{C}_B$ with a large enough extension chain, i.e., with $|\mathcal{U}(C)| \geq k$, we stop and return a trivial yes-instance. This is safe, since by definition all of the cycles in $\mathcal{U}(C)$ are edge-disjoint. Otherwise, we define $\mathcal{C}_M$ to be all cycles of $\mathcal{C}_B$ together with their extension chains, plus the root of $\mathcal{T}(G)$ and the set of all non-tree cycles: $\mathcal{C}_M = \{r\} \cup \mathcal{C}_N \cup \bigcup_{C \in \mathcal{C}_B} \mathcal{U}(C)$, where $\mathcal{C}_N$ denotes the set of all shortest cycles in G that are not in $\mathcal{T}(G)$. We first observe that $\mathcal{C}_M$ is closed under taking the lowest extension.

Claim 5.13. *For every cycle $C \in \mathcal{C}_M$, its lowest extension $L(C)$ also belongs to $\mathcal{C}_M$, if it exists.*

Proof of the Claim. For each $C \in \mathcal{C}_B$, the statement is immediate for C and all cycles in $\mathcal{U}(C)$, since their lowest extensions are added to $\mathcal{U}(C)$ exhaustively by the definition.

It remains to consider non-tree cycles. Let C be an S-node in $\mathcal{T}(G)$ with $\mathcal{P}_s(C) = \{P_0, \ldots, P_\ell\}$ and consider a non-tree cycle $C' = P_i \cup P_j$ for some $i < j \in [\ell]_0$. By the proof of Claim 5.12, if $i > 0$ and $j < \ell$, the lowest extension of C' is $P_{i-1} \cup P_{j+1}$, which is either another non-tree cycle or the S-node C; both are part of $\mathcal{C}_M$ by definition. Finally, if $i = 0$ we claim that $L(C') = L(P_0 \cup P_{\ell-1})$, where $P_0 \cup P_{\ell-1} \in \mathcal{C}_B$ and $L(P_0 \cup P_{\ell-1}) \in \mathcal{C}_M$. This holds since both lowest extensions are above C, while the intersection between C and C' is P_0, which is the same as the intersection between C and $P_0 \cup P_{\ell-1}$. The argument for $j = \ell - 1$ and $L(C') = L(P_1 \cup P_\ell)$ is analogous. ◁

By construction, we have an upper-bound on the size of $\mathcal{C}_M$, which we prove in the next claim.

Claim 5.14. *The size of $\mathcal{C}_M$ is $O(k^2)$.*

Proof of the Claim. We first bound the number of base cycles, i.e., the size of $\mathcal{C}_B$. Recall that we have already bounded the number of leaves in $\mathcal{T}(G)$ by $4k - 1$. This also implies that the number

of internal vertices of $\mathcal{T}(G)$ that have at least two children is less than $4k$. Therefore, the total number of U-nodes and S-nodes in $\mathcal{C}_B$ is less than $4k$. By definition, $\mathcal{C}_B$ contains two additional cycles for each S-node; from the above, the number of such cycles is less than $8k$. The total size of $\mathcal{C}_B$ is therefore less than $16k$.

For every $C \in \mathcal{C}_B$, the size of its extension chain $\mathcal{U}(C)$ is at most $k-1$. Thus, the cycles of $\mathcal{C}_B$ together with their extension chains amount to less than $16k^2$ cycles. In addition to these cycles, $\mathcal{C}_M$ contains the root and the non-tree cycles $\mathcal{C}_N$. By the above, we have less than $4k$ S-nodes, and in total they have less than $8k$ children. Since an S-node with p children contains at most $p \cdot (p+1)/2$ non-tree cycles, there are at most $4k \cdot (8k+1)$ non-tree cycles in G in total. The size of $\mathcal{C}_M$ can be therefore upper-bounded by $52 \cdot k^2 = O(k^2)$. ◁

We now show the main property of the set of marked cycles $\mathcal{C}_M$, that no other cycles are needed in order to find a $\mathcal{T}$-maximal solution.

Claim 5.15. *For every $\mathcal{T}$-maximal solution $\mathcal{C}$, it holds that $\mathcal{C} \subseteq \mathcal{C}_M$.*

Proof of the Claim. Consider a $\mathcal{T}$-maximal solution $\mathcal{C}$, and assume by contradiction that it is not contained in $\mathcal{C}_M$. Let C be the minimal cycle in $\mathcal{C}$ with respect to $(\leq)$ that is not in $\mathcal{C}_M$. Since $\mathcal{C}_M$ contains all leaves and internal vertices of $\mathcal{T}(G)$ with at least two children and all non-tree shortest cycles of G, we have that C is a U-node in $\mathcal{T}(G)$ with exactly one child C'. If $C' \in \mathcal{C}$, then $C' \in \mathcal{C}_M$ by the choice of C; but then $C' <_e C$, and C is the lowest extension of C', which contradicts the assumption that $C \notin \mathcal{C}_M$, since by Claim 5.13, $\mathcal{C}_M$ is closed under taking the lowest extension.

Therefore, $C' \notin \mathcal{C}$, and let $\mathcal{C}' = \mathcal{C} \cup \{C'\} \setminus \{C\}$ be a set of k shortest cycles obtained from $\mathcal{C}$ by replacing C with C'. Since $\mathcal{C}$ is $\mathcal{T}$-maximal, it cannot be that $\mathcal{C}'$ is a solution, thus C' must share edges with some other cycle $C'' \in \mathcal{C} \setminus \{C\}$; this cycle must be contained in G_C, since otherwise it also shares edges with C. By the choice of C, C'' is in the set of marked cycles $\mathcal{C}_M$. We claim that C is then the lowest extension of C'', which again contradicts the assumption that $C \notin \mathcal{C}_M$. Indeed, since both C and C'' are part of the solution $\mathcal{C}$, and $C'' \leq C$, we have that $C'' <_e C$. On the other hand, C shares edges with C', which is the only other maximal cycle in G_C with respect to $(\leq)$. Thus, no other shortest cycle in G_C is an extension of C'', and $C = L(C'')$. This finishes the proof of the claim. ◁

For a set of cycles $\mathcal{C}$ in G, let $\bigcup \mathcal{C}$ be a shortcut for $\bigcup_{C \in \mathcal{C}} C$, i.e., the graph obtained by taking the union of all cycles in $\mathcal{C}$, which is a subgraph of G. Let $H = \bigcup \mathcal{C}_M$; in other words, H is the subgraph of G obtained by removing all vertices and edges that are not in any of the cycles in $\mathcal{C}_M$. Since H is a subgraph of G, we treat the weight function w as a weight function on edges of H as well. From Claim 5.15, it follows immediately that (G, w, k) is a yes-instance of Edge-Disjoint Shortest Cycle Packing if and only if (H, w, k) is a yes-instance. Therefore, we focus on (H, w, k) for the remainder of the proof.

Observe also that the LSCT $\mathcal{T}(H)$ can be seen as the result of dissolving degree-2 nodes in $\mathcal{T}(G)$. Here, dissolving a degree-2 node means removing the node from the graph while making its neighbors adjacent. Indeed, by construction of $\mathcal{C}_M$, the only shortest cycles in G that are not necessarily in $\mathcal{C}_M$ are U-nodes with exactly one child. Removing vertices and edges of such a cycle C that are not part of $\mathcal{C}_M$ results in the child of C being directly adjacent to the parent of C in the new LSCT. Note that all leaves, S-nodes, U-nodes with at least two children and the root node are preserved between $\mathcal{T}(G)$ and $\mathcal{T}(H)$.

While H is constructed from only $O(k^2)$ cycles of G, it is not necessarily a kernel yet, since the number of vertices involved in these cycles may be large. We now argue that the number of vertices in H can be reduced by showing that only a few vertices in H have degree more than 2. We identify the vertices of degree at least 3 by following the LSCT $\mathcal{T}(H)$. For each $C \in \mathcal{T}(H)$, we define the set of vertices $B(C) \subseteq V(H_C)$ as follows. If C is an S-node, then $B(C) = \{s, t\}$, where s, t

are the poles of C. If C is a U-node, let $B(C)$ be the set of vertices of degree at least three in the graph $\bigcup\{C\} \cup \mathcal{C}_u(C)$, i.e., we consider only the cycle C and its children in $\mathcal{T}(H)$.

Claim 5.16. *For each U-node C of $\mathcal{T}(H)$ with p children, the size of $B(C)$ is at most $p \cdot (p+1)$.*

Proof of the Claim. The family $\{C\} \cup \mathcal{C}_u(C)$ contains $p+1$ cycles. By Lemma 5.1, every two of these cycles either touch or do not intersect. Therefore, for every two cycles C_1 and C_2 there are at most two vertices of degree at least 3 in $C_1 \cup C_2$, which are the endpoints of the shared path $C_1 \cap C_2$. On the other hand, every vertex of degree at least three in $\bigcup \mathcal{C}_u(C) \cup \{C\}$ has two incident edges that belong to some distinct cycles $C_1, C_2 \in \{C\} \cup \mathcal{C}_u(C)$, which means that this vertex has degree at least three also in $C_1 \cup C_2$. Thus, the size of $B(C)$ is at most twice the number of pairs of cycles in $\{C\} \cup \mathcal{C}_u(C)$, showing the claim. ◁

Claim 5.17. *The number of vertices of degree at least three in H is in $O(k^2)$.*

Proof of the Claim. Let $B = \bigcup_{C \in \mathcal{T}(H)} B(C)$. Let v be a vertex of degree at least 3 in H, we claim that B contains v. By definition, B contains the poles of all splittable cycles, so it remains to consider the case where v is not a pole of any splittable cycle. Since the degree of v is at least three in H, there exist two distinct shortest cycles C_1, C_2 in H that contain distinct edges incident to v. Assume C_1 is a non-tree cycle, then by Lemma 5.7, (ii), there exist a splittable cycle $C \in \mathcal{T}(H)$ with $\mathcal{P}_s(C) = \{P_0, \ldots, P_\ell\}$ and $C_1 = P_i \cup P_j$ for some $0 \le i < j \le \ell$. Since v is not a pole of C, v is an internal vertex of P_i or P_j; in any case, there exists a cycle in $\mathcal{C}_s(C)$ that contains the same edges incident to v as C_1, so C_1 may be replaced by this cycle. Since an analogous argument can be applied to C_2, from now on we assume that both C_1 and C_2 are nodes of $\mathcal{T}(H)$.

Let C_1, C_2 be the closest pair of cycles with this property, where the distance is measured over the tree $\mathcal{T}(H)$, and let C_1 be the highest node in $\mathcal{T}(H)$ among C_1, C_2. We claim that either C_1 is the parent of C_2 in $\mathcal{T}(H)$, or C_1 and C_2 have a common parent in $\mathcal{T}(H)$. Assume the contrary, and consider two cases depending on the locations of C_1 and C_2 in $\mathcal{T}(H)$.

Case 1 ($C_2 \le C_1$). Consider the parent C of C_2 in $\mathcal{T}(H)$, $C \le C_1$ and $C \ne C_1$. The cycle C contains v, since otherwise v is an inner vertex of H_C and cannot be on the cycle C_1. If C contains the same edges incident to v as C_2, then C_1, C fulfill the same property but are closer in $\mathcal{T}(H)$ than C_1, C_2, which is a contradiction. Otherwise, C contains an edge incident to v that is not contained in C_2, and the pair (C, C_2) fulfills the property, leading to a contradiction.

Case 2 ($C_2 \not\le C_1$ and $C_1 \not\le C_2$). Let C be the parent of C_2 in $\mathcal{T}(H)$. By the assumption, C is not the parent of C_1, and also $C_1 \not\le C$ since otherwise C_1 has lower depth in $\mathcal{T}(H)$ than C_2. Similarly to the previous case, C contains v, as otherwise C_1 and C_2 cannot share the vertex. Again, either C contains the same edges incident to v as C_2, or it contains an edge incident to v that is not part of C_2. Thus, either C_1 and C or C and C_2 fulfill the property while being closer in $\mathcal{T}(H)$ than C_1 and C_2, which is a contradiction.

We now have that either C_1 is the parent of C_2, or there is a cycle C that is the parent of both C_1 and C_2 in $\mathcal{T}(H)$. If C_1 is the parent of C_2, and C_1 is an S-node, then v is a pole, which we assume is not the case; the same holds if C is an S-node and $C_1, C_2 \in \mathcal{C}_s(C)$. If C_1 is the parent of C_2 and C_1 is a U-node, then $C_1 \cup C_2$ is a subgraph of $\bigcup \mathcal{C}_u(C_1) \cup \{C_1\}$, so $v \in B(C_1)$. By the same argument, if C is a U-node and $C_1, C_2 \in \mathcal{C}_u(C)$, then v has degree at least three in $C_1 \cup C_2$ and so in $\bigcup \mathcal{C}_u(C) \cup \{C\}$, thus $v \in B(C)$. This concludes the proof that B contains all vertices of degree at least 3 in H.

It remains to bound the number of elements in B. For each $C \in \mathcal{T}(H)$ that is either an S-node or a U-node with one child, it holds that $|B(C)| \le 2$. Since $|\mathcal{C}_M| \in O(k^2)$, the total contribution of these nodes to B is $O(k^2)$. Observe that $\mathcal{T}(H)$ has at most $4k$ leaves, since the leaves of $\mathcal{T}(H)$ and the leaves of $\mathcal{T}(G)$ are the same. Therefore, the total number of children of all U-nodes with at least two children in $\mathcal{T}(H)$ is at most $8k$. Since a U-node with p children contributes at most

$p \cdot (p+1)$ vertices to B by Claim 5.16, all U-nodes with at least two children contribute in total at most $8k \cdot (8k+1) \in O(k^2)$ vertices to B. The total size of B is therefore also $O(k^2)$. ◁

We now exhaustively apply the following reduction rules, in order to remove all degree-2 vertices in H.

Reduction Rule 1. *If there is a vertex of degree 2 in H, dissolve it: remove the vertex and add an edge between its former neighbors, such that the weight of the new edge is equal to the sum of weights of the two removed edges.*

Note that 1 may create loops and parallel edges. Thus we allow for intermediate instances to be multigraphs. We introduce two more reduction rules so that the resulting instance is a simple graph. Notice that while the original graph G is clean, these reduction rules, namely Reduction Rule 3, can spoil this property. It is possible to introduce an additional rule that would delete all edges that are not included in any shortest cycle. However, we just don't need the graph to be clean at this stage of the kernelization algorithm. Thus, we simply state the rules without this assumption.

Reduction Rule 2. *If there is a loop e in H, remove it from the graph. If $w(e) = g(G)$, additionally decrease k by 1. If this increases the length of the shortest cycle, return a trivial no-instance unless k becomes 0, in which case return a trivial yes-instance.*

Reduction Rule 3. *If there are two parallel edges e_1 and e_2 in H, remove both and decrease k by 1 if $w(e_1) + w(e_2) = g(G)$. If this increases the length of the shortest cycle, return a trivial no-instance unless k becomes 0, in which case return a trivial yes-instance. Otherwise, if $w(e_1) + w(e_2) > g(G)$, then remove one of the parallel edges with the highest weight without changing k.*

Claim 5.18. *Reduction Rules 1–3 are safe.*

Proof of the Claim. First, for Reduction Rule 1, observe that every cycle in H either remains unchanged, or is replaced by a cycle that passes through the newly-created edge, while the length of the cycle and the set of remaining edges stays the same. Therefore, the length of all cycles is unchanged, any two cycles that were edge-disjoint stay edge-disjoint, and any two cycles that shared edges still share edges after applying the reduction rule.

Second, consider a loop e to which Reduction Rule 2 is applied. If $w(e) > g(G)$, no shortest cycle passes through e, thus it can be safely removed. If $w(e) = g(G)$, the only shortest cycle passing through e is the one that contains the edge e and nothing else, as only simple cycles are considered. Adding this cycle to the solution is safe as no other cycle in H can share edges with it.

Finally, let e_1, e_2 be the parallel edges in Reduction Rule 3, with the endpoints u, v. We start with the case $w(e_1) + w(e_2) = g(G)$. If $k = 1$, then the cycle C containing e_1 and e_2 is indeed a solution; if $k > 1$ and $g(H') > g(H)$, where $H' = H - e_1 - e_2$, then (H, w, k) is a no-instance: Otherwise, each cycle in the solution must use a different edge in $\{e_1, e_2\}$, so $k = 2$, but then the remaining u-v-paths of the two cycles in H' form a cycle of length $g(G) = g(H)$ in H', contradicting $g(H) < g(H')$. Now, with $g(H') = g(H)$, we claim that a packing of k edge-disjoint shortest cycles exist in H if and only if a packing of $k-1$ edge-disjoint shortest cycles exists in $H' = H$. The backward direction is clear, since the cycle C containing e_1 and e_2 can be added to any packing in H'. In the forward direction, if the statement does not hold, then the solution in H does not contain the cycle C, but contains at least one cycle with an edge in $\{e_1, e_2\}$. If there is just one such cycle, then it can be replaced by C, so that the remaining $k-1$ cycles form a packing in H. If there are two cycles C_1 and C_2, containing e_1 and e_2 respectively, let P_1 be the u-v-path in C_1 other than e_1, and P_2 be the u-v-path in C_2 other than e_2. Since $w(C_1) + w(C_2) = 2g(G)$ and $w(e_1) + w(e_2) = g(G)$, it holds that $w(P_1) + w(P_2) = g(G)$ and the cycle $C' = P_1 \cup P_2$ is also a shortest cycle. However, we can then replace C_1 and C_2 with C and C' to get a packing of k edge-disjoint shortest cycles in H that contains C.

In the other case, when $w(e_1) + w(e_2) > g(G)$, let without loss of generality be $w(e_2) \geq w(e_1)$. If $w(e_1) < w(e_2)$, then no shortest cycle may pass through e_2, so removing the edge e_2 is safe. If $w(e_1) = w(e_2) > g(G)/2$, then no shortest cycle can contain both e_1 and e_2. Also, it cannot be that two edge-disjoint shortest cycles contain e_1 and e_2, respectively, as then by joining $C_1 - e_1$ and $C_2 - e_2$ we get a cycle that is shorter than $g(G)$. Thus, in any solution at most one cycle passes through either e_1 or e_2. Thus, it is safe to remove e_2 since any such cycle may pass through e_1 instead. ◁

By applying Reduction Rules 1–3 exhaustively, we obtain a simple graph G' with the weight function w', and an integer $k' \leq k$ such that (G, w, k) is equivalent to (G', w', k') (by Claim 5.18). Moreover, $|V(G')| = O(k^2)$, by Claim 5.17 and since Reduction Rule 1 can no longer be applied. It remains to compress the weights, so that they can be represented by $k^{O(1)}$ bits. Let $n = |V(G')|$, $m = |E(G'|$. We have that $m \leq 3n - 6$ as G' is a simple planar graph. Thus, $m = O(k^2)$. We now interpret the weight function $w' : E(G') \to \mathbb{Z}$ as a vector in $\mathbb{Z}^m$ by assigning an arbitrary numbering to the edges of G', and invoke Proposition 2.4 with w' and $N = m + 1$ to obtain a compressed vector $\overline{w} \in \mathbb{Z}^m$. From Proposition 2.4, we have that for any vector $b \in \mathbb{Z}^m$ with $||b||_1 \leq m$, $\mathsf{sign}(w' \cdot b) = \mathsf{sign}(\overline{w} \cdot b)$. Interpreting the vector $\overline{w}$ as a weight function on $E(G')$, we obtain the instance $(G', \overline{w}, k')$ of Edge-Disjoint Shortest Cycle Packing. By Proposition 2.4, $||\overline{w}||_\infty \leq 2^{4m^3} \cdot (m+1)^{m(m+2)} = 2^{O(k^6)}$, therefore each entry of $\overline{w}$ can be represented by $O(k^6)$ bits. Thus, the bit-size of $(G', \overline{w}, k')$ is $O(k^8)$. It remains to verify that $(G', \overline{w}, k')$ is indeed equivalent to (G', w', k'). We show that for every pair of cycles in G', their weights compare in the same way before and after the weight compression.

Claim 5.19. *For every two cycles C_1, C_2 in G', $w'(C_1)$ is smaller than (equal to/larger than) $w'(C_2)$ if and only if $\overline{w}(C_1)$ is smaller than (equal to/larger than) $\overline{w}(C_2)$.*

Proof of the Claim. It suffices to show that $\mathsf{sign}(w'(C_1) - w'(C_2)) = \mathsf{sign}(\overline{w}(C_1) - \overline{w}(C_2))$. For each cycle C_i with $i \in \{1, 2\}$, consider its characteristic vector $a_i \in \{0, 1\}^m$, where the jth coordinate is equal to 1 if and only if the jth edge of G' belongs to C_i. By definition, $w'(C_i) = w' \cdot a_i$ and $\overline{w}(C_i) = \overline{w} \cdot a_i$ for each $i \in \{1, 2\}$. Let $b \in \{-1, 0, 1\}^m$ be the difference vector $a_1 - a_2$, then $w'(C_1) - w'(C_2) = w' \cdot b$ and $\overline{w}(C_1) - \overline{w}(C_2) = \overline{w} \cdot b$. By construction, $||b||_1 \leq m$, so Proposition 2.4 applies to the vector b, therefore $\mathsf{sign}(w' \cdot b) = \mathsf{sign}(\overline{w} \cdot b)$, which means that $\mathsf{sign}(w'(C_1) - w'(C_2)) = \mathsf{sign}(\overline{w}(C_1) - \overline{w}(C_2))$. ◁

From Claim 5.19, we have that every shortest cycle in the instance $(G', \overline{w}, k')$ is a shortest cycle in (G', w', k'), and the other way around. Therefore, the instances are equivalent with respect to the existence of a packing of k edge-disjoint shortest cycles. Since (G', w', k') was previously shown to be equivalent to the original instance (G, w, k) of Edge-Disjoint Shortest Cycle Packing, $(G', \overline{w}, k')$ is equivalent to (G, w, k) as well, and, from the above, the bit-size of $(G', \overline{w}, k')$ is $O(k^8)$. This completes the proof of the theorem. □

By using the small family of marked cycles constructed in the proof of the theorem, we immediately get the following FPT algorithm for Edge-Disjoint Shortest Cycle Packing on planar graphs.

Corollary 5.20. *Edge-Disjoint Shortest Cycle Packing can be solved in $k^{O(k)} \cdot n^{O(1)}$ time on planar graphs.*

Proof. Given the instance (G, w, k) of Edge-Disjoint Shortest Cycle Packing, construct the set of marked cycles $\mathcal{C}_M$ as in the proof of Theorem 4. By Claim 5.15, we have that whenever there exists a solution for (G, w, k), there also exists one where all cycles are in $\mathcal{C}_M$. By Claim 5.14, we

also have that $C_M = O(k^2)$. Therefore, in time $k^{O(k)} \cdot n^{O(1)}$ we can enumerate all k-tuples of cycles in C_M and check for each tuple whether the cycles are edge-disjoint. □

Finally, combining the equivalence between Edge-Disjoint Shortest Cycle Packing and Min-Cut Packing on planar graphs, and the two results above, we get analogous results for Min-Cut Packing on planar graphs.

Corollary 5.21. *Min-Cut Packing on planar graphs admits a polynomial kernel such that the output graph has $O(k^2)$ vertices. Furthermore, the problem can be solved in $k^{O(k)} \cdot n^{O(1)}$ time on planar graphs.*

6 Kernelization Lower Bound for Shortest Cycle Packing and Edge-Disjoint Shortest Cycle Packing

Note that Theorem 1 implies that Shortest Cycle Packing and Edge-Disjoint Shortest Cycle Packing are in XP, but since the lower bound in Theorem 2 does not apply for these problems, it remains open whether they are FPT or W[1]-hard on general graphs. Here, we show that both problems do not admit polynomial kernels. Formally, we prove the following.

Theorem 6. *Shortest Cycle Packing and Edge-Disjoint Shortest Cycle Packing parameterized by solution size k do not admit polynomial kernels unless* NP $\subseteq$ coNP/poly.

To this end, we first define Disjoint Shortest Factors, a version of Disjoint Factors where all of the factors of the solution need to be shortest factors. Let Σ be a finite alphabet. We use Σ^n to denote all words of length n over Σ. Given a word $w = w_1 w_2 \dots w_n$, we denote the ith charater in w by $w[i]$, that is, $w[i] = w_i$. An x-factor of a word $w = w_1 w_2 \dots w_n$ is a substring $w_i w_{i+1} \dots w_j$ for some $i < j \in [n]$ which starts and ends with x, that is, $w_i = x = w_j$. The length of a factor $w_i w_{i+1} \dots w_j$ is $j - i$ and a x-factor is a *shortest factor* if there is no other x-factor which has smaller length. Two factors $w_1 = w_{i_1} w_{i_1+1} \dots, w_{j_1}$ and $w_2 = w_{i_2} w_{i_2+1} \dots, w_{j_1}$ intersect if $\{i_1, i_1 + 1, \dots, j_1\} \cap \{i_2, i_2 + 1, \dots, j_2\} \neq \emptyset$, that is, there is an index in which w_1 and w_2 intersect. Otherwise, the factors are disjoint. We can now define Disjoint Shortest Factors formally.

Disjoint Shortest Factors

Input:	A word w over a finite alphabet Σ with $\square \in \Sigma$.
Task:	Decide whether there is a set of pairwise disjoint shortest factors including an x-factor for each letter $x \in \Sigma \setminus \{\square\}$.

Note that we allow for a special character $\square$ for which no factor needs to be picked. Since adding two consecutive $\square$ to the start of the input word adds a shortest factor for $\square$, this does not change the computational complexity of Disjoint Shortest Factors. We show that Disjoint Shortest Factors does not admit a polynomial kernel when parameterized by the alphabet size. We believe that this result may be interesting by itself. We mention that our reduction is similar to the reduction that shows that Disjoint Factors does not admit a polynomial kernel [6].

Proposition 6.1. Disjoint Shortest Factors *parameterized by $|\Sigma|$ does not admit a polynomial kernel.*

Proof. We present an OR-cross-decomposition from 3,4-Sat, a variant of Sat where each clause contains exactly three literals and each variable appears at most four times. This version is known to be NP-complete [37]. We first present a polynomial-time many-one reduction from 3,4-Sat to Disjoint Shortest Factors. Given a formula ϕ in 3-CNF-Sat, our alphabet consists of one

character for each of the $3m$ positions in ϕ and one character for each variable x_i in ϕ. We create the word:

$$w = 123123123456456456 \dots (3m-2)(3m-1)(3m)(3m-2)(3m-1)(3m)\square\square w_{x_1} w_{x_2} \dots w_{x_n},$$

where:

$$w_{x_i} = x_i p_1^i \square\square p_1^i p_2^i \square\square p_2^i p_3^i \square\square p_3^i p_4^i \square\square p_4^i x_i n_1^i \square\square n_1^i n_2^i \square\square n_2^i n_3^i \square\square n_3^i n_4^i \square\square n_4^i x_i.$$

Therein, p_j^i (respectively n_j^i) are the positions where x_i appears for the jth time positively (respectively negated) or $\square$ if x_i does not appear j times positively (negated). Note that the shortest factor for any of the position characters has length three and for any variable character, it has length 17. If the formula ϕ is satisfiable, then we find a shortest factor for each character in w as follows. Let β be a satisfying assignment for ϕ. For each variable x_i, if $\beta(x_i)$ is false, then we select the first x_i-factor in w_{x_i} (the one through all p_j^i's). If $\beta(x_i)$ is true, then we select the second x_i-factor in w_{x_i} (the one through all n_j^i's). For each clause C_j, we select one variable v_j such that $\beta(v_j)$ satisfies C_j. Note that such a variable exists as β is a satisfying assignment. Let c_j be the position of v_j in C_j. Note that by construction, we can pick a shortest c_j-factor in w_{v_j}. For the other two positions in C_j, we can find disjoint shortest factors in the starting part of w.

In the other direction, if w admits a set of disjoint shortest factors, then we will construct a satisfying assignment for ϕ. For each variable x_i, w only contains two shortest x_i-factors and both are contained in w_{x_i}. If the first one is chosen, then we set x_i to false and otherwise, we set x_i to true. Suppose this assignment does not satisfy ϕ. Then, there exists a clause C_j that is not satisfied. Note that for each position character, there are only three shortest factors in w, two in the starting part and one in a word w_{x_i} for the variable in the respective position. Moreover in the starting part, we can find disjoint shortest factors for at most two out of the three positions $(3j-2)$, $(3j-1)$, and $3j$ in C_j. This means that there is at least one position in C_j whose position-character-factor has been chosen within a variable word w_i. This is a contradiction to the fact that the chosen assignment does not satisfy C_j.

We next present the OR-cross-decomposition. Given t instances of 3,4-Sat, where all instances contain the same number of clauses and variables, we use the previously described many-one reduction to construct words $w_1, w_2, \dots, w_t$, one for each instance. We assume that $t = 2^s$ for some integer s. Otherwise, we can copy one of the instances at most t times to ensure that the number of instances is a power of 2. Note that by construction, all words w_i have the same length $N = 9m + 35n \in O(n+m)$, all words are over the same alphabet Σ, and the shortest factor for each character has the same length in each w_i. Next, we combine all of these instances into one instance such that the whole instance is a yes-instance if and only if at least one of the words w_i is a yes-instance using s additional characters $c_1, c_2, \dots, c_s$. Let $\Sigma' = \Sigma \cup \{c_1, c_2, \dots, c_s\}$. We iteratively combine two batches of instances of the same size into one instance, that is, in the ith round, we combine two batches containing 2^{i-1} instances into one batch containing 2^i instances using character c_i. Note that after $s = \log_2(t)$ rounds, we get one instance encoding all t instances.

We next describe how two batches are merged. To this end, let W_1 and W_2 be the words corresponding to two batches and let $N_i = |W_1| = |W_2|$. We create the word:

$$W = c_i W_1 \square^{N_i/2} c_i \square^{N_i/2} W_2 c_i \square.$$

Note that $N_i = 3N_{i-1} + 4$, where $N_0 = N$. This implies that $N_i = 3^i N + 4i$. In particular, $N_s = 3^s N + 4s = t^{\log_2(3)} N + 4\log_2(t)$, that is, the whole constructed instances has size in $\text{poly}(tN)$. Moreover, $|\Sigma'| \leq N + \log_2(t)$. It hence only remains to show that the constructed instance is a

yes-instance if and only if at least one w_i is a yes-instance. To this end, first observe that all shortest c_i-factors are contained in the words for batches of 2^i input words as we introduced N_i □ between two such words when creating batches for $i + 1$.

Assume first that one word w_i is a yes-instance. Then we find a set of disjoint shortest factors for all characters in Σ' in W as follows. For each character $a \in \Sigma$, we take a shortest a-factor from w_i. Next for each c_i, we look at the batch word W_i in round i that contains w_i. Note that W_i contains the character c_i three times and between any two of them, there is a shortest c_i-factor. We pick the one that does not intersect with w_i. Note that in this way we can find a set of disjoint shortest factors for all characters in Σ'.

Now assume that the word W is a yes-instance and let S be a solution. Similar to before, the character $c_{\log_2(t)}$ is contained only three times in W. We take the shortest $c_{\log_2(t)}$-factor in S and consider the rest W' of the word W. We iteratively select the shortest c_i-factor in S for decreasing values of i. After $\log_2(t)$ iterations, we are only left with a single input word w_i. By assumption, S contains disjoint shortest factors for each character in Σ, that is, the instance w_i is a yes-instance. This concludes the proof. □

Now we can prove Theorem 6 which we restate.

Theorem 6. *Shortest Cycle Packing and Edge-Disjoint Shortest Cycle Packing parameterized by solution size k do not admit polynomial kernels unless* NP ⊆ coNP/poly.

Proof. We present a polynomial parameter transformation from Disjoint Shortest Factors. To this end, let $w \in \Sigma^n$ be a string. We assume without loss of generality that $n \geq 10$. We start with a path of length $2n$. Let the vertices be $u_1, u_2, \ldots, u_{2n}$. Next, for each $a \in \Sigma$, we add a new vertex v_a to the graph. Let $i_1^a < i_2^a < \ldots < i_j^a$ be all the positions i where $w[i] = a$ and let d_a be the size of a shortest a-factor in w. We add a path between v_a and u_{2i_p} for each $p \in [j]$. The path between v_a and u_{i_p} is of length $3n - d_a$. Finally, we set $k = |\Sigma|$.

Since the reduction can be computed in polynomial time and Disjoint Shortest Factors does not admit a polynomial kernel when parameterized by $|\Sigma|$, it only remains to show that w has k disjoint shortest factors if and only if there exists a set of k disjoint cycles of length g in the constructed graph, where g is the girth of the graph. We start by showing that the girth of the graph is $6n$. Note that the set of vertices $\{v_a \mid a \in \Sigma\}$ is a feedback vertex set of the constructed graph and hence each cycle passes through at least one of these vertices. Moreover, each cycle that passes through at least two such vertices are of length at least $4(3n - n) = 8n$ as $d_a \leq n$ for each $a \in \Sigma$. Next, each cycle that passes through exactly one vertex $v_a \in \{v_a \mid a \in \Sigma\}$ has length at least $2(3n - d_a) + 2d_a = 6n$ as any two vertices of the initial path that are connected to the same vertex v_a have distance at least $2d_a$ as any factor of w has length at least d_a. Moreover, each cycle through v_a that has length exactly $g = 6n$ corresponds to a shortest a-factor in w. □

We mention that forcing the shortest distances for all factors to be equal is basically the same as Colorful Independent Set in unit interval graphs where all monochromatic cliques have size at most 2. This admits a cubic kernel as shown by van Bevern et al. [3]. For the sake of completeness, we sketch a cubic kernel directly but based on their ideas. Observe that the distance ℓ of any shortest factor is at most k as otherwise, any shortest factor would contain some letter twice, a contradiction to the factor being shortest and all shortest factors having the same length ℓ. Next, we can add $\ell + 3$ new characters and build a quadratic-size gadget in the beginning that allows all of these to be picked in a solution. Afterwards, we can use these characters to replace deleted characters (since we cannot use a single character □ in this variant). The gadget looks as follows:

$$1, 3, 4, \ldots, \ell + 2, 1, 2, 4, 5, \ldots, \ell + 3, 3, 1, 5, 6, \ldots, \ell + 3, 3, \ldots, \ell + 3, 1, 2, \ldots, \ell, \ell + 3.$$

With this at hand, we can observe that if there are at least $2k-1$ shortest factors of a given letter, then each of these can intersect with at most two factors in any solution, that is, we can ignore this character (replace it by the new characters). Now, each of the $O(k)$ character has at most $2k$ shortest factors, each of length at most k. Keeping only characters contained in these factors and adding "separators" of length k between "connected components" results in a cubic kernel.

7 Packing Minimum Cuts

In this section, we show that packing k minimum cuts in a graph is W[1]-hard when parameterized by k. This is partially related to Shortest Cycle Packing as packing k minimum cuts in the dual of a planar graph is the same as packing k shortest cycles in the primal graph.

Theorem 7. *Min-Cut Packing is* W[1]*-hard when parameterized by solution size k on graphs with unit edge weights.*

Proof. We present a reduction from Independent Set parameterized by solution size k. Recall that the problem is W[1]-complete [12, 16]. To this end, let $(G = (V, E), k)$ be an instance of Independent Set. We create an equivalent instance $(H = (V', E'), k)$ of Min-Cut Packing as follows. Let Δ be the maximum degree in G. We set $V' = V \cup U$ where U is a set of $2\Delta + 3$ new vertices. For the edge set E', we start with the edge set $E \cup \{\{u, v\} \mid u, v \in U\}$, that is, we make U into a clique of size $2\Delta + 3$. Moreover for each vertex $v \in V$, we add edges between v and $2\Delta + 1 - \deg_G(v)$ arbitrary vertices in U. This concludes the construction.

Since the solution sizes for both instances are the same and since the reduction can be computed in polynomial time, it only remains to show that G contains an independent set of size k if and only if there are k disjoint minimum cuts in H. We first show that all minimum cuts in H separate exactly one vertex $v \in V$ from the rest of the graph. Note that each such cut has size $2\Delta + 1$. Moreover, no cut of size at most $2\Delta + 1$ can separate two vertices in U as each proper cut in a clique of size $\ell = 2\Delta + 3$ contains at least $\ell - 1 = 2\Delta + 2$ edges. Lastly, consider a cut that separates two vertices $u, v \in V$ from all vertices in U. Such a cut contains at least $2\Delta + 1 - \Delta = \Delta + 1$ edges between u and vertices in U and also at least $\Delta + 1$ edges between v and vertices in U. Since these two sets of edges are disjoint, such a cut has size at least $2\Delta + 2$ and is therefore not a minimum cut.

This basically concludes the proof as each minimum cut in H now corresponds to selecting a single vertex v in G and two cuts are disjoint if and only if the two selected vertices are non-adjacent in G, that is, a packing of k disjoint minimum cuts in H corresponds to an independent set of size k in G. □

8 Conclusion

We investigated the parameterized complexity of finding packings of vertex or edge-disjoint cycles of bounded total length. In Theorem 1, we showed that Min-Sum Cycle Packing and Min-Sum Edge-Disjoint Cycle Packing are in XP when parameterized by the number of cycles. The result is tight in the sense that both problems are proven to be W[1]-hard in Theorem 2. The interesting special cases where we pack shortest cycles are Shortest Cycle Packing and Edge-Disjoint Shortest Cycle Packing. Trivially, Theorem 1 implies that Shortest Cycle Packing and Edge-Disjoint Shortest Cycle Packing are in XP. However, our lower bound in Theorem 2 does not apply here. This leads to an intriguing open problem as to whether Shortest Cycle Packing and Edge-Disjoint Shortest Cycle Packing are FPT or W[1]-hard on general graphs. Currently, we can only rule out the existence of polynomial kernels (Theorem 6). We also observed that Min-Cut Packing—which is dual for Edge-Disjoint Shortest Cycle Packing on planar graph—becomes W[1]-hard for general case (Theorem 7).

Further, we considered the case of planar graphs. We proved that Shortest Cycle Packing and Edge-Disjoint Shortest Cycle Packing are FPT when parameterized by the number of cycles and, furthermore, Edge-Disjoint Shortest Cycle Packing admits a polynomial kernel. The most interesting open question here is whether Min-Sum Cycle Packing and Min-Sum Edge-Disjoint Cycle Packing are FPT on planar graphs. Our algorithms for Shortest Cycle Packing and Edge-Disjoint Shortest Cycle Packing crucially depend on the structure of shortest cycles in planar graphs, and we cannot apply the same ideas for the more general length constraint. Further, does Shortest Cycle Packing admit a polynomial kernel in the planar case? The kernelization algorithm for Edge-Disjoint Shortest Cycle Packing in Theorem 4 exploits the fact that a plane graph with at least $4k$ shortest facial cycles always has a packing of k edge-disjoint shortest cycles by the four-color theorem, and we do not have similar properties in the case of vertex-disjoint cycles. On the other side, the reduction in Theorem 6 is far from planar and cannot be used to infer a similar kernelization lower bound for Shortest Cycle Packing.

Another interesting open problem for planar graphs is about packings of cycles from uncrossable families (we refer to [24, 35, 36] for the definitions). Is it possible to extend our results for shortest cycles for these more general cycle families?

Finally, we would be interested to know whether the running time of our algorithms could be improved. The algorithm from Theorem 1 runs in $n^{O(k^6)}$ time. Can the running time be improved to, say, $n^{O(k)}$? For the planar case, we solve Shortest Cycle Packing and Edge-Disjoint Shortest Cycle Packing in $k^{O(k)} \cdot n^{O(1)}$ time. Is there a single-exponential in k algorithm? It may even be that that these problems can solved in subexponential time.

References

[1] Kenneth Appel and Wolfgang Haken. 1989. *Every Planar Map Is Four Colorable*. American Mathematical Society.

[2] Matthias Bentert, André Nichterlein, Malte Renken, and Philipp Zschoche. 2023. Using a geometric lens to find k-disjoint shortest paths. *SIAM Journal on Discrete Mathematics* 37, 3 (2023), 1674–1703.

[3] René van Bevern, Matthias Mnich, Rolf Niedermeier, and Mathias Weller. 2015. Interval scheduling and colorful independent sets. *Journal of Scheduling* 18, 5 (2015), 449–469.

[4] Andreas Björklund and Thore Husfeldt. 2019. Shortest two disjoint paths in polynomial time. *SIAM Journal on Computing* 48, 6 (2019), 1698–1710.

[5] Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. 2016. (Meta) kernelization. *Journal of the ACM* 63, 5, Article 44 (2016), 1–69.

[6] Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. 2011. Kernel bounds for disjoint cycles and disjoint paths. *Theoretical Computer Science* 412, 35 (2011), 4570–4578.

[7] Glencora Borradaile, Amir Nayyeri, and Farzad Zafarani. 2015. Towards single face shortest vertex-disjoint paths in undirected planar graphs. In *Proceedings of the 23rd Annual European Symposium Algorithms (ESA)*. Springer, 227–238.

[8] Leizhen Cai, Siu Man Chan, and Siu On Chan. 2006. Random separation: A new method for solving fixed-cardinality optimization problems. In *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation (IWPEC)*. Springer, 239–250.

[9] Zhi-Zhong Chen. 2001. Approximation algorithms for independent sets in map graphs. *Journal of Algorithms* 41, 1 (2001), 20–40.

[10] Zhi-Zhong Chen, Michelangelo Grigni, and Christos H. Papadimitriou. 2002. Map graphs. *Journal of the ACM* 49, 2 (2002), 127–138.

[11] Éric Colin De Verdière and Alexander Schrijver. 2011. Shortest vertex-disjoint two-face paths in planar graphs. *ACM Transactions on Algorithms* 7, 2 (2011), 1–12.

[12] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. 2015. *Parameterized Algorithms*. Springer.

[13] Samir Datta, Siddharth Iyer, Raghav Kulkarni, and Anish Mukherjee. 2018. Shortest k-disjoint paths via determinants. In *Proceedings of the 38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*. Schloss Dagstuhl—Leibniz-Zentrum Für Informatik, Article 19, 1–21.

[14] Reinhard Diestel. 2012. *Graph Theory*. Springer.

[15] Edsger W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1 (1959), 269–271.
[16] Rodney G. Downey and Michael R. Fellows. 2013. *Fundamentals of Parameterized Complexity*. Springer.
[17] Tali Eilam-Tzoreff. 1998. The disjoint shortest paths problem. *Discrete Applied Mathematics* 85, 2 (1998), 113–138.
[18] Paul Erdös and George Szekeres. 1935. A combinatorial problem in geometry. *Compositio Mathematica* 2 (1935), 463–470.
[19] Michael Etscheid, Stefan Kratsch, Matthias Mnich, and Heiko Röglin. 2017. Polynomial kernels for weighted problems. *Journal of Computer System Sciences* 84 (2017), 1–10.
[20] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. 2016. Efficient computation of representative families with applications in parameterized and exact algorithms. *Journal of the ACM* 63, 4 (2016), 29:1–29:60.
[21] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. 2020. Bidimensionality and kernels. *SIAM Journal on Computing* 49, 6 (2020), 1397–1422.
[22] András Frank and Éva Tardos. 1987. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica* 7, 1 (1987), 49–65.
[23] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability*. W.H. Freeman and Company.
[24] Michel X. Goemans and David P. Williamson. 1998. Primal-dual approximation algorithms for feedback problems in planar graphs. *Combinatorica* 18, 1 (1998), 37–59.
[25] Venkatesan Guruswami, C. Pandu Rangan, Maw-Shang Chang, Gerard J. Chang, and C. K. Wong. 1998. The vertex-disjoint triangles problem. In *Proceedings of the 24th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*. Springer, 26–37.
[26] David Hartvigsen and Russell Mardon. 1994. The all-pairs min cut problem and the minimum cycle basis problem on planar graphs. *SIAM Journal on Discrete Mathematics* 7, 3 (1994), 403–418.
[27] Ian Holyer. 1981. The NP-completeness of some edge-partition problems. *SIAM Journal on Computing* 10, 4 (1981), 713–717.
[28] John E. Hopcroft and Robert E. Tarjan. 1974. Efficient planarity testing. *Journal of the ACM* 21, 4 (1974), 549–568.
[29] David R. Karger and Clifford Stein. 1996. A new approach to the minimum cut problem. *Journal of the ACM* 43, 4 (1996), 601–640.
[30] Yusuke Kobayashi and Tatsuya Terao. 2022. One-face shortest disjoint paths with a deviation terminal. In *Proceedings of the 33rd International Symposium on Algorithms and Computation (ISAAC)*. Schloss Dagstuhl—Leibniz-Zentrum für Informatik, Article 47, 1–15.
[31] William Lochet. 2021. A polynomial time algorithm for the k-disjoint shortest paths problem. In *Proceedings of the 32nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 169–178.
[32] Mathieu Mari, Anish Mukherjee, Michał Pilipczuk, and Piotr Sankowski. 2024. Shortest disjoint paths on a grid. In *Proceedings of 35th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 346–365.
[33] Dieter Rautenbach and Friedrich Regen. 2009. On packing shortest cycles in graphs. *Information Processing Letters* 109, 14 (2009), 816–821.
[34] Neil Robertson, Daniel P. Sanders, Paul D. Seymour, and Robin Thomas. 1997. The four-colour theorem. *Journal of Combinatorial Theory, Series B* 70, 1 (1997), 2–44.
[35] Niklas Schlomberg. 2024. An improved integrality gap for disjoint cycles in planar graphs. In *Proceeding of the 51st International Colloquium on Automata, Languages, and Programming (ICALP)*. Schloss Dagstuhl—Leibniz-Zentrum für Informatik, Article 122, 1–15.
[36] Niklas Schlomberg, Hanjo Thiele, and Jens Vygen. 2023. Packing cycles in planar and bounded-genus graphs. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2069–2086.
[37] Craig A. Tovey. 1984. A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics* 8, 1 (1984), 85–89.

Received 28 October 2024; revised 11 August 2025; accepted 23 August 2025