Packing Cycles Faster Than Erdős-Pósa*

Daniel Lokshtanov[1], Amer E. Mouawad[1], Saket Saurabh[3], and Meirav Zehavi[1]

1 **University of Bergen, Bergen, Norway**
`daniel.lokshtanov@ii.uib.no`
2 **University of Bergen, Bergen, Norway**
`a.mouawad@ii.uib.no`
3 **University of Bergen, Bergen, Norway; and The Institute of Mathematical Sciences, Chennai, India**
`saket@imsc.res.in`
4 **University of Bergen, Bergen, Norway**
`meirav.zehavi@ii.uib.no`

Abstract

The Cycle Packing problem asks whether a given undirected graph $G = (V, E)$ contains k vertex-disjoint cycles. Since the publication of the classic Erdős-Pósa theorem in 1965, this problem received significant scientific attention in the fields of Graph Theory and Algorithm Design. In particular, this problem is one of the first problems studied in the framework of Parameterized Complexity. The non-uniform fixed-parameter tractability of Cycle Packing follows from the Robertson–Seymour theorem, a fact already observed by Fellows and Langston in the 1980s. In 1994, Bodlaender showed that Cycle Packing can be solved in time $2^{\mathcal{O}(k^2)} \cdot |V|$ using exponential space. In case a solution exists, Bodlaender's algorithm also outputs a solution (in the same time). It has later become common knowledge that Cycle Packing admits a $2^{\mathcal{O}(k \log^2 k)} \cdot |V|$-time (deterministic) algorithm using exponential space, which is a consequence of the Erdős-Pósa theorem. Nowadays, the design of this algorithm is given as an exercise in textbooks on Parameterized Complexity. Yet, no algorithm that runs in time $2^{o(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$, beating the bound $2^{\mathcal{O}(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$, has been found. In light of this, it seems natural to ask whether the $2^{\mathcal{O}(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$ bound is essentially optimal. In this paper, we answer this question negatively by developing a $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})} \cdot |V|$-time (deterministic) algorithm for Cycle Packing. In case a solution exists, our algorithm also outputs a solution (in the same time). Moreover, apart from beating the bound $2^{\mathcal{O}(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$, our algorithm runs in time linear in $|V|$, and its space complexity is polynomial in the input size.

1998 ACM Subject Classification G.2.2 Graph Algorithms, I.1.2 Analysis of Algorithms

Keywords and phrases Parameterized Complexity, Graph Algorithms, Cycle Packing, Erdős-Pósa Theorem

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.71

1 Introduction

The Cycle Packing problem asks whether a given undirected graph $G = (V, E)$ contains k vertex-disjoint cycles. Since the publication of the classic Erdős-Pósa theorem in 1965 [15], this problem received significant scientific attention in the fields of Graph Theory and Algorithm Design. In particular, Cycle Packing is one of the first problems studied in the framework of Parameterized Complexity. In this framework, each problem instance is

* A full version of the paper is available at `https://arxiv.org/abs/1707.01037`.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 71; pp. 71:1–71:15
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

associated with a parameter k that is a non-negative integer, and a problem is said to be *fixed-parameter tractable (FPT)* if the combinatorial explosion in the time complexity can be confined to the parameter k. More precisely, a problem is FPT if it can be solved in time $f(k) \cdot |I|^{\mathcal{O}(1)}$ for some function f, where $|I|$ is the input size. For more information, we refer the reader to recent monographs such as [14] and [10].

In this paper, we study the Cycle Packing problem from the perspective of Parameterized Complexity. In the standard parameterization of Cycle Packing, the parameter is the number k of vertex-disjoint cycles. The non-uniform fixed-parameter tractability of Cycle Packing follows from the Robertson–Seymour theorem [38],[1] a fact already observed by Fellows and Langston in the 1980s. In 1994, Bodlaender showed that Cycle Packing can be solved in time $2^{\mathcal{O}(k^2)} \cdot |V|$ using exponential space [5]. Notably, in case a solution exists, Bodlaender's algorithm also outputs a solution in time $2^{\mathcal{O}(k^2)} \cdot |V|$.

The Erdős-Pósa theorem states that there exists a function $f(k) = \mathcal{O}(k \log k)$ such that for each non-negative integer k, every undirected graph either contains k vertex-disjoint cycles or it has a feedback vertex set consisting of $f(k)$ vertices [15]. It is well known that the treewidth (tw) of a graph is not larger than its feedback vertex set number (fvs), and that a naive dynamic programming (DP) scheme solves Cycle Packing in time $2^{\mathcal{O}(tw \log tw)} \cdot |V|$ and exponential space (see, e.g., [10]). Thus, the existence of a $2^{\mathcal{O}(k \log^2 k)} \cdot |V|$-time (deterministic) algorithm that uses exponential space can be viewed as a direct consequence of the Erdős-Pósa theorem. Nowadays, the design of this algorithm is given as an exercise in textbooks on Parameterized Complexity such as [14] and [10]. In case a solution exists, this algorithm does not output a solution (though we remark that with a certain amount of somewhat non-trivial work, it is possible to modify this algorithm to also output a solution).

Prior to our work, no algorithm that runs in time $2^{o(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$, beating the bound $2^{\mathcal{O}(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$, has been found. In light of this, it seemed tempting to ask whether the $2^{\mathcal{O}(k \log^2 k)} \cdot |V|^{\mathcal{O}(1)}$ bound is essentially optimal. In particular, two natural directions to explore in order to obtain a faster algorithm necessarily lead to a dead end. First, Erdős and Pósa [15] proved that the bound $f(k) = \mathcal{O}(k \log k)$ in their theorem is essentially tight as there exist infinitely many graphs and a constant c such that these graphs do not contain k vertex-disjoint cycles and yet their feedback vertex set number is larger than $ck \log k$. Second, Cygan et al. [11] proved that the bound $2^{\mathcal{O}(tw \log tw)} \cdot |V|^{\mathcal{O}(1)}$ is also likely to be essentially tight in the sense that unless the Exponential Time Hypothesis (ETH) [20] is false, Cycle Packing cannot be solved in time $2^{o(tw \log tw)} \cdot |V|^{\mathcal{O}(1)}$ (however, it might still be true that Cycle Packing is solvable in time $2^{o(fvs \log fvs)} \cdot |V|^{\mathcal{O}(1)}$).

1.1 Related Work

The Cycle Packing problem admits a factor $\mathcal{O}(\log |V|)$ approximation algorithm [30], and it is quasi-NP-hard to approximate within a factor of $\mathcal{O}(\log^{\frac{1}{2}-\epsilon} |V|)$ for any $\epsilon > 0$ [18]. In the context of kernelization with respect to the parameter k, Cycle Packing does not admit a polynomial kernel unless NP $\subseteq$ coNP/Poly [4]. Recently, Lokshtanov et al. [31] obtained a 6-approximate kernel with $\mathcal{O}((k \log k)^2)$ vertices along with a $(1+\epsilon)$-approximate kernel with $k^{\mathcal{O}(f(\epsilon))}$ vertices for some function f. We would like to mention that in case one seeks k edge-disjoint cycles rather than k vertex-disjoint cycles, the problem becomes significantly simpler in the sense that it admits a kernel with $\mathcal{O}(k \log k)$ vertices [4].

Focusing on structural parameters, Bodlaender et al. [2] obtained polynomial kernels with respect to the vertex cover number, vertex-deletion distance to a cluster graph and

[1] The paper [38] was already available as a manuscript in 1986 (see, e.g., [5]).

the max leaf number. In planar graphs, Bodlaender et al. [3] solved Cycle Packing in subexponential time $2^{\mathcal{O}(\sqrt{k})} \cdot |V|^{\mathcal{O}(1)}$, and showed that this problem admits a linear kernel. In the more general class of H-minor-free graphs, Dorn et at. [13] also solved Cycle Packing in subexponential time $2^{\mathcal{O}(\sqrt{k})} \cdot |V|^{\mathcal{O}(1)}$. Moreover, for apex-minor-free graphs, Fomin et al. [17] showed that Cycle Packing admits a linear kernel, and Fomin et al. [16] showed that it also admits an EPTAS. When the input graph is a directed graph, Cycle Packing is W[1]-hard [39], but it admits an FPT approximation scheme [19]. In fact, Cycle Packing in directed graphs was the first W[1]-hard problem shown to admit such a scheme. Krivelevich et al. [30] obtained a factor $\mathcal{O}(|V|^{\frac{1}{2}})$ approximation algorithm for Cycle Packing in directed graphs and showed that this problem is quasi-NP-hard to approximate within a factor of $\mathcal{O}(\log^{1-\epsilon}|V|)$ for any $\epsilon > 0$.

Several variants of Cycle Packing have also received significant scientific attention. For example, the variant of Cycle Packing where one seeks k *odd* vertex-disjoint cycles has been widely studied [36, 40, 35, 29, 27, 28]. Another well-known variant, where the cycles need to contain a prescribed set of vertices, has also been extensively investigated [24, 33, 25, 23, 26]. Furthermore, a combination of these two variants has been considered in [23, 22].

Finally, we briefly mention that inspired by the Erdős-Pósa theorem, a class of graphs $\mathcal{H}$ is said to have the Erdős-Pósa property if there is a function $f(k)$ for which given a graph G, it either contains k vertex-disjoint subgraphs such that each of these subgraphs is isomorphic to a graph in $\mathcal{H}$, or it contains a set of $f(k)$ vertices that hits each of its subgraphs that is isomorphic to a graph in $\mathcal{H}$. A fundamental result in Graph Theory by Robertson and Seymour [37] states the the class of all graphs that can be contracted to a fixed planar graph H has the Erdős-Pósa property. Recently, Chekuri and Chuzhoy [6] presented a framework that leads to substantially improved functions $f(k)$ in the context of results in the spirit of the Erdős-Pósa theorem. Among other results, these two works are also related to the recent breakthrough result by Chekuri and Chuzhoy [7], which states that every graph of treewidth at least $f(k) = \mathcal{O}(k^{98} \cdot \mathrm{polylog}(k))$ contains the $k \times k$ grid as a minor (the constant 98 has been improved to 36 in [8] and to 19 in [9]). Following the seminal work by Robertson and Seymour [37], numerous papers (whose survey is beyond the scope of this paper) investigated which other classes of graphs have the Erdős-Pósa property, which are the "correct" functions f associated with them, and which generalizations of this property lead to interesting discoveries.

1.2 Our Contribution

In this paper, we show that the running time of the algorithm that is a consequence of the Erdős-Pósa theorem is not essentially tight. For this purpose, we develop a $2^{\mathcal{O}(\frac{k\log^2 k}{\log\log k})} \cdot |V|$-time (deterministic) algorithm for Cycle Packing. In case a solution exists, our algorithm also outputs a solution (in time $2^{\mathcal{O}(\frac{k\log^2 k}{\log\log k})} \cdot |V|$). Moreover, apart from beating the bound $2^{\mathcal{O}(k\log^2 k)} \cdot |V|^{\mathcal{O}(1)}$, our algorithm runs in time linear in $|V|$, and its space complexity is polynomial in the input size. Thus, we also improve upon the classical $2^{\mathcal{O}(k^2)} \cdot |V|$-time algorithm by Bodlaender [5]. Our result is summarized in the following theorem.

▶ **Theorem 1.** *There exists a (deterministic) polynomial-space algorithm that solves* Cycle Packing *in time* $2^{\mathcal{O}(\frac{k\log^2 k}{\log\log k})} \cdot |V|$. *In case a solution exists, it also outputs a solution.*

At a high level to prove Theorem 1, the main idea is to look back at the classic algorithm based on the Erdős-Pósa property and to perform some trade-offs to get an improvement. Specifically, the idea is to compute a "relaxed feedback vertex set": a set S of size

$\mathcal{O}(k \log k / \log\log k)$ so that $G - S$ is not necessarily a forest, but it has no short cycles (of length $\mathcal{O}(\log k / \log\log k)$), even after contracting long detached paths to single edges. This creates a "relaxed" modulator that is smaller by a $\mathcal{O}(\log\log k)$ multiplicative factor than if we required S to be a feedback vertex set. After some cleaning step on this structure, and using the relaxed modulator, one can roughly guess the interaction between S and $G - S$ in the solution; this step amounts to guessing one of around $|S|!$ options, which is $\mathcal{O}(2^{k \log^2 k / \log\log k})$. Having fixed the interaction, we can contract any remaining long paths so that the whole remaining graph has $\mathcal{O}(k \log^{3/2} k)$ vertices; it is important here that this step goes through even when we work with the "relaxed" modulator S as explained above, instead of a normal feedback vertex set as in the classic algorithm. As the size of the graph is already bounded, a simple dynamic programming on subsets suffices to finish the proof. In order to achieve polynomial space usage, we employ a standard inclusion-exclusion procedure instead.

2 Preliminaries

We use standard terminology from the book of Diestel [12] for those graph-related terms that are not explicitly defined here. We only consider finite graphs possibly having self-loops and multi-edges. Moreover, using an appropriate reduction rule we restrict the maximum multiplicity of an edge to be 2. For a graph G, we use V and E to denote the vertex and edge sets of the graph G, respectively. For a vertex $v \in V$, we use $\deg_G(v)$ to denote the degree of v, i.e the number of edges incident on v, in the (multi) graph G. We also use the convention that a self-loop at a vertex v contributes 2 to its degree. For a vertex subset $S \subseteq V$, we let $G[S]$ and $G - S$ denote the graphs induced on S and $V \setminus S$, respectively. For a vertex subset $S \subseteq V$, we use $N_G(S)$ and $N_G[S]$ to denote the open and closed neighborhoods of S in G, respectively. That is, $N_G(S) = \{v \mid \{u, v\} \in E, u \in S\} \setminus S$ and $N_G[S] = N_G(S) \cup S$. In case $S = \{v\}$, we simply let $N(v) = N(S)$ and $N[v] = N[S]$. For a graph $G = (V, E)$ and an edge $e \in E$, we let G/e denote the graph obtained by contracting e in G. For $E' \subseteq \binom{V}{2}$, i.e. a subset of edges, we let $G + E'$ denote the (multi) graph obtained after adding the edges in E' to G, and we let G/E' denote the (multi) graph obtained after contracting the edges of E' in G. The girth of a graph is denoted by $\mathrm{girth}(G)$, its minimum degree by $\delta(G)$, and its maximum degree by $\Delta(G)$. A graph with no cycles has infinite girth.

A *path* in a graph is a sequence of distinct vertices $v_0, v_1, \ldots, v_\ell$ such that $\{v_i, v_{i+1}\}$ is an edge for all $0 \le i < \ell$. A *cycle* in a graph is a sequence of distinct vertices $v_0, v_1, \ldots, v_\ell$ such that $\{v_i, v_{(i+1) \bmod \ell+1}\}$ is an edge for all $0 \le i \le \ell$. Both a double edge and a self-loop are cycles. If P is a path from a vertex u to a vertex v in the graph G then we say that u and v are the end vertices of the path P and P is a (u, v)-path. For a path P, we use $V(P)$ and $E(P)$ to denote the sets of vertices and edges in the path P, respectively, and length of P is denoted by $|P|$ (i.e, $|P| = |V(P)|$). For a cycle C, we use $V(C)$ and $E(C)$ to denote the sets of vertices and edges in the cycle C, respectively, and the length of C, denoted by $|C|$, is $|V(C)|$. For a path or a cycle Q we use $N_G(Q)$ and $N_G[Q]$ to denote the sets $N_G(V(Q))$ and $N_G[V(Q)]$, respectively. For a collection of paths/cycles $\mathcal{Q}$, we use $|\mathcal{Q}|$ to denote the number of paths/cycles in $\mathcal{Q}$ and $V(\mathcal{Q})$ to denote the set $\bigcup_{Q \in \mathcal{Q}} V(Q)$. We say a path P in G is a *degree-two path* if all vertices in $V(P)$, including the end vertices of P, have degree exactly 2 in G. We say P is a *maximal degree-two path* if no proper superset of P also forms a degree-two path. We note that the notions of *walks* and *closed walks* are defined exactly as paths and cycles, respectively, except that their vertices need not be distinct. Finally, a *feedback vertex set* is a subset F of vertices such that $G - F$ is a forest.

Below we formally state some of the key results that will be used throughout the paper, starting with the classic Erdős-Pósa theorem [15].

▶ **Proposition 2** ([15]). *There exists a constant c' such that every (multi) graph either contains k vertex-disjoint cycles or it has a feedback vertex set of size at most $c'k \log k$.*

Observe that any (multi) graph $G = (V, E)$ whose feedback vertex set number is bounded by $c'k \log k$ has less than $(2c'k \log k + 1) \cdot |V|$ edges (recall that we restrict the multiplicity of an edge to be 2). Indeed, letting F denote a feedback vertex set of minimum size, the worst case (in terms of $|E|$) is obtained when $G - F$ is a tree, which contains $|V| - |F| - 1$ edges, and between every pair of vertices $v \in F$ and $u \in V$, there exists an edge of multiplicity 2. Thus, by Proposition 2, in case $|E| > (2c'k \log k + 1) \cdot |V|$, the input instance is a yes-instance, and after we discard an arbitrary set of $|E| - (2c'k \log k + 1) \cdot |V|$ edges, it remains a yes-instance. A simple operation which discards at least $|E| - (2c'k \log k + 1) \cdot |V|$ edges and can be performed in time $\mathcal{O}(k \log k \cdot |V|)$.

▶ **Assumption 3.** *We assume that $|E| = \mathcal{O}(k \log k \cdot |V|)$.*

Now, we state our algorithmic version of Proposition 2. The proof partially builds upon the proof of the Erdős-Pósa theorem in the book [12], and it is given in the full version of the paper.

▶ **Theorem 4.** *There exists a constant c and a polynomial-space algorithm such that given a (multi) graph G and a non-negative integer k, in time $k^{\mathcal{O}(1)} \cdot |V|$ it either outputs k vertex-disjoint cycles or a feedback vertex set of size at most $ck \log k = r$.*

Next, we state two results relating to cycles of average and short lengths.

▶ **Proposition 5** ([1]). *Any (multi) graph $G = (V, E)$ on n vertices with average degree d contains a cycle of length at most $2 \log_{d-1} n + 2$.*

Itai and Rodeh [21] showed that given a (multi) graph $G = (V, E)$, an "almost" shortest cycle (if there is any) in G can be found in time $\mathcal{O}(|V|^2)$. To obtain a linear dependency on $|V|$ (given a small feedback vertex set), we prove the following result. A complete proof will appear in the full version of the paper.

▶ **Lemma 6.** *Given a (multi) graph $G = (V, E)$ and a feedback vertex set F of G, a shortest cycle (if there is any) in G can be found in time $\mathcal{O}(|F| \cdot (|V| + |E|))$.*

Finally, we state a result that will be used (in Lemma 11) to bound the size of a graph we obtain after performing simple preprocessing operations as well as repetitive removal of short cycles.

▶ **Proposition 7** ([34], Lemma 9). *Let $T = (V, E)$ be a forest on N vertices. Let $M' = \{\{i, j\} \in E \mid deg_T(i) = deg_T(j) = 2\}$ and $L = \{a \in V \mid deg_T(a) \leq 1\}$. Then there exists $M \subseteq M'$ such that M is a matching and $|W| \geq \frac{N}{4}$ where $W = L \cup M$.*

3 Removing Leaves, Induced Paths, and Short Cycles

As is usually the case when dealing with cycles in a graph, we first define three rules which help getting rid of vertices of degree at most 2 as well as edges of multiplicity larger than 2. It is not hard to see that all three Reduction Rules A1, A2, and A3 are safe, i.e. they preserve solutions in the reduced graph.

▶ **Reduction Rule A1.** *Delete vertices of degree at most 1.*

▶ **Reduction Rule A2.** *If there is a vertex v of degree exactly 2 that is not incident to a self-loop, then delete v and connect its two (not necessarily distinct) neighbors by a new edge.*

▶ **Reduction Rule A3.** *If there is a pair of vertices u and v in V such that $\{u, v\}$ is an edge of multiplicity larger than 2, then reduce the multiplicity of the edge to 2.*

Observe that the entire process that applies these rules exhaustively can be done in time $\mathcal{O}(|V| + |E|) = \mathcal{O}(k \log k \cdot |V|)$. Indeed, in time $\mathcal{O}(|V|)$ we first remove the vertex-set of each maximal path between a leaf and a degree-two vertex. No subsequent application of Rule A2 or Rule A3 creates vertices of degree at most one. Now, we iterate over the set of degree-two vertices. For each degree-two vertex that is not incident to a self-loop, we apply Rule A2. Next, we iterate over E, and for each edge of multiplicity larger than two, we apply Rule A3. At this point, the only new degree-two vertices that can be created are vertices incident to exactly one edge, whose multiplicity is two. Therefore, during one additional phase where we exhaustively apply Rule A2, the only edges of multiplicity larger than two that can be created are self-loops. Thus, after one additional iteration over E, we can ensure that no rule among Rules A1, A2 and A3 is applicable.

Since these rules will be applied dynamically and iteratively, we define an operator, denoted by $\mathsf{reduce}(G)$, that takes as input a graph G and returns the (new) graph G' that results from an exhaustive application of Rules A1, A2 and A3.

▶ **Definition 8.** For a (multi) graph G, we let $G' = \mathsf{reduce}(G)$ denote the graph obtained after an exhaustive application of Reduction Rules A1, A2 and A3. $|\mathsf{reduce}(G)|$ denotes the number of vertices in $\mathsf{reduce}(G)$. Moreover, $\mathsf{img}(\mathsf{reduce}(G))$ denotes the pre-image of $\mathsf{reduce}(G)$, i.e. $\mathsf{img}(\mathsf{reduce}(G))$ is the set of vertices in G which are not deleted in $\mathsf{reduce}(G)$.

▶ **Observation 9.** *For a graph $G = (V, E)$ and a set $E' \subseteq \binom{V}{2}$ it holds that $|\mathsf{reduce}(G+E')| \leq |\mathsf{reduce}(G)| + 2|E'|$.*

The first step of our algorithm consists of finding, in time linear in $|V|$, a set S satisfying the conditions specified in Lemmata 10 and 11. Intuitively, S will contain vertices of "short" cycles in the input graph, where short will be defined later.

▶ **Lemma 10.** *Given a (multi) graph $G = (V, E)$ and two integers $k > 0$ and $g > 6$, there exists an $k^{\mathcal{O}(1)} \cdot |V|$-time algorithm that either finds k vertex-disjoint cycles in G or finds a (possibly empty) set $S \subseteq V$ such that girth($\mathsf{reduce}(G - S)) > g$ and $|S| < gk$.*

Proof. We proceed by constructing such an algorithm. First, we apply the algorithm of Theorem 4 which outputs either k vertex-disjoint cycles or a feedback vertex set F of size at most $ck \log k = r$. In the former case we are done. In the latter case, i.e. the case where a feedback vertex set F is obtained, we apply the following procedure iteratively (initially, we set $S = \emptyset$):

(1) Apply Lemma 6 to find a shortest cycle C in $\mathsf{reduce}(G)$.
(2) If no cycle was found or $|C| > g$ then return S.
(3) Otherwise, i.e. if $|C| \leq g$, then add the vertices of C to S, delete those vertices from G to obtain G', set $G = G'$, and repeat from Step (1).

Note that if Step (3) is applied k times then we can terminate and return the corresponding k vertex-disjoint cycles in G. Hence, when the condition of Step (2) is satisfied, i.e. when the described procedure terminates, the size of S is at most $g(k-1) < gk$ and girth($\mathsf{reduce}(G - S)) > g$. Since the algorithm of Theorem 4 runs in time $k^{\mathcal{O}(1)} \cdot |V|$, and each iteration of Steps (1)-(3) is performed in time $\mathcal{O}((k \log k)^2 \cdot |V|)$, we obtain the desired time complexity. ◀

▶ **Lemma 11.** *Given a (multi) graph $G = (V, E)$ and two integers $k > 0$ and $g > 6$, let S denote the set obtained after applying the algorithm of Lemma 10 (assuming no k vertex-disjoint cycles obtained). Then $|\mathsf{reduce}(G - S)| \leq (2ck \log k)^{1+\frac{6}{g-6}} + 3ck \log k$.*

Proof. Let $G' = (V', E') = \mathsf{reduce}(G - S)$ and $|V'| = n'$. First, recall that G admits a feedback vertex set of size at most $ck \log k = r$. Since Reduction Rules A1, A2 and A3 do not increase the feedback vertex set of the graph (see, e.g., [34], Lemma 1), G' also admits a feedback vertex set F of size at most r. Let T denote the induced forest on the remaining $N = n' - r$ vertices in G'. Moreover, from Lemma 10, we know that $\text{girth}(G') > g > 6$.

Next, we apply Proposition 7 to T to get W. Now with every element $a \in W$ we associate an unordered pair of vertices of F as follows. Assume $a \in L$, i.e. a is a vertex of degree 0 or 1. Since the degree of a is at least 3 in G', a has at least two neighbors in F. We pick two of these neighbors arbitrarily and associate them with a. We use $\{x_a, y_a\}$ to denote this pair. If $a = \{u, v\}$ is an edge from M then each of u and v has degree at least 3 in G' and each has at least one neighbor in F. We pick one neighbor for each and associate the pair $\{x_u, x_v\}$ with a. Note that since $\text{girth}(G') > 6$, $x_u \neq x_v$ and $x_a \neq y_a$.

We now construct a new multigraph $G^\star = (V^\star, E^\star)$ with vertex set $V^\star = F$ as follows. For every vertex $a \in W$ we include an edge in $E^\star$ between x_a and y_a, and for every edge $a = \{u, v\} \in W$ we include an edge in $E^\star$ between x_u and x_v. By Proposition 7, we know that W is of size at least $\frac{N}{4}$. It follows that $G^\star$ has at least $\frac{N}{4}$ edges and hence its average degree is at least $\frac{N}{2r}$ as $|V^\star| = ck \log k = r$. Note that if $G^\star$ has a cycle of length at most ℓ, then G' has a cycle of length at most 3ℓ, as any edge of the cycle in $G^\star$ can be replaced by a path of length at most 3 in G'. Combining this with the fact that $\text{girth}(G') > g > 6$, we conclude that $G^\star$ contains no self-loops or parallel edges. Hence $G^\star$ is a simple graph with average degree at least $\frac{N}{2r}$. By Proposition 5, $G^\star$ must have a cycle of length at most

$$2 \log_{\frac{N}{2r}-1} r + 2 = \frac{2 \log r}{\log(\frac{N}{2r} - 1)} + 2$$

which implies that G' must have a cycle of length at most

$$\frac{6 \log r}{\log(\frac{N}{2r} - 1)} + 6.$$

Finally, by using the fact that $\text{girth}(G') > g$ and substituting N and r, we get

$$\begin{aligned}
\frac{6 \log r}{\log(\frac{N}{2r} - 1)} + 6 > g \quad & \iff \log r > \frac{(g-6)}{6} \log\left(\frac{N - 2r}{2r}\right) \\
& \iff \log r > \frac{(g-6)}{6} \log(N - 2r) - \frac{(g-6)}{6} \log(2r) \\
& \iff \frac{\log r + \frac{(g-6)}{6} \log(2r)}{\frac{(g-6)}{6}} > \log(N - 2r) \\
& \implies \frac{\log(2r) + \frac{(g-6)}{6} \log(2r)}{\frac{(g-6)}{6}} > \log(N - 2r) \\
& \iff (1 + \frac{6}{g-6}) \log(2r) > \log(N - 2r) \\
& \iff (1 + \frac{6}{g-6}) \log(2ck \log k) > \log(n' - 3ck \log k) \\
& \iff (2ck \log k)^{1+\frac{6}{(g-6)}} + 3ck \log k > n'.
\end{aligned}$$

This completes the proof. ◀

The usefulness of Lemma 11 comes from the fact that by setting $g = \frac{48 \log k}{\log \log k} + 6$, we can guarantee that $|\,\mathsf{reduce}(G - S)| < 3ck \log k + 2ck \log^{1.5} k$, and therefore we can beat the $\mathcal{O}(k \log^2 k)$ bound. That is, we have the following consequence.

▶ **Corollary 12.** *Given a (multi) graph $G = (V, E)$ and an integer $k > 0$, let S denote the set obtained after applying the algorithm of Lemma 10 with $g = \frac{48 \log k}{\log \log k} + 6$ (assuming no k vertex-disjoint cycles obtained). Then $|\,\mathsf{reduce}(G - S)| \leq 3ck \log k + 2ck \log^{1.5} k$.*

Proof. By Lemma 11, $|\,\mathsf{reduce}(G - S)| \leq (2ck \log k)^{1+\frac{\log \log k}{8 \log k}} + 3ck \log k$. Assuming $k > \log k > c > 2$, we have $(2ck \log k)^{1+\frac{\log \log k}{8 \log k}} = (2ck \log k)(2ck \log k)^{\frac{\log \log k}{8 \log k}} \leq (2ck \log k)k^{\frac{4 \log \log k}{8 \log k}}$. Now note that $k^{\frac{4 \log \log k}{8 \log k}} \leq \log^{0.5} k$. Hence, $(2ck \log k)^{1+\frac{\log \log k}{8 \log k}} \leq 2ck \log k \log^{\frac{1}{2}} k \leq 2ck \log^{1.5} k$. This completes the proof. ◀

4 Bounding the Core of the Remaining Graph

At this point, we assume, without loss of generality, that we are given a graph $G = (V, E)$, a positive integer k, $g = \frac{48 \log k}{\log \log k} + 6$, and a set $S \subseteq V$ such that $\text{girth}(\mathsf{reduce}(G - S)) > g$, $|S| < gk$, and $|\,\mathsf{reduce}(G - S)| \leq 3ck \log k + 2ck \log^{1.5} k$.

Even though the number of vertices in $\mathsf{reduce}(G - S)$ is bounded, the number of vertices in $G - S$ is unbounded. In what follows, we show how to bound the number of "objects" in $G - S$, where an object is either a vertex in $G - S$ or a degree-two path in $G - S$. The next lemma is a refinement extending a lemma by Lokshtanov et al. [31] (Lemma 5.2).

▶ **Lemma 13.** *Let $G = (V, E)$ be a (multi) graph and let $X \subseteq V$ be any subset of the vertices of G. Suppose there are more than $|X|^2(2|X| + 1)$ vertices in $G - X$ whose degree in $G - X$ is at most one. Then, there is either an isolated vertex w in $G - X$ or an edge $e \in E$ such that (G, k) is a yes-instance of* Cycle Packing *if and only if either $(G - \{w\}, k)$ or $(G/e, k)$ is a yes-instance. Moreover, there is an $\mathcal{O}(|X|^2 \cdot k \log k \cdot |V|)$-time algorithm that given G and X, outputs sets $V_X \subseteq V \setminus X$ and $E_X \subseteq E(G - X)$ such that, for the graph $G' = (G/E_X) - V_X$, it holds that (G, k) is a yes-instance of* Cycle Packing *if and only if (G', k) is a yes-instance of* Cycle Packing*, and $G' - X$ contains at most $|X|^2(2|X| + 1)$ vertices whose degree in $G' - X$ is at most one.*

Armed with Lemma 13, we are now ready to prove the following result. For a forest T, we let $T_{\leq 1}$, T_2, and $T_{\geq 3}$, denote the sets of vertices in T having degree at most one in T, degree exactly two in T, and degree larger than two in T, respectively. Moreover, we let $\mathcal{P}$ denote the set of all maximal degree-two paths in T.

▶ **Lemma 14.** *Let $G = (V, E)$, S, k, and g be as defined above. Let $R = \mathsf{img}(\mathsf{reduce}(G - S)) \subseteq (V \setminus S)$ denote the pre-image of $\mathsf{reduce}(G - S)$ in $G - S$. Then, $T = G - S - R$ is a forest and for every maximal degree-2 path in $\mathcal{P}$ there are at most two vertices on the path having neighbors in R (in the graph $G - S$). Moreover, in time $k^{\mathcal{O}(1)} \cdot |V|$, we can guarantee that $|T_{\leq 1}|$, $|\mathcal{P}|$, and $|T_{\geq 3}|$ are bounded by $k^{\mathcal{O}(1)}$.*

Proof. To see why $T = G - S - R$ must be a forest it is sufficient to note that for any cycle in $G - S$ at least one vertex from that cycle must be in $R = \mathsf{img}(\mathsf{reduce}(G - S))$ (see Figure 1). Recall that, since $\text{girth}(\mathsf{reduce}(G - S)) > 6$, every vertex in R has degree at least 3 in $G - S$. Now assume there exists some path $P \in \mathcal{P}$ having exactly three (the same argument holds for any number) distinct vertices u, v and w (in that order) each having at least one neighbor in R (possibly the same neighbor). We show that the middle vertex v must have been in R,

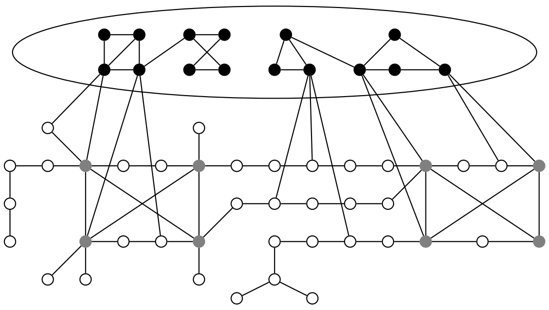

Figure 1 A graph G (not all edges shown), the set S (in black), the set R (in gray), and the set $T = G - R - S$ (in white).

contradicting the fact that $T = G - S - R$. Consider the graph $G - S$ and apply Reduction Rules A1, A2 and A3 exhaustively (in $G - S$) on all vertices in the tree containing P except for u, v and w. Regardless of the order in which we apply the reduction rules, the path P will eventually reduce to a path on three vertices, namely u, v, and w. To see why v must be in R observe that even if the other two vertices have degree two in the resulting graph, after reducing them, v will have degree at least three (into R) and is therefore non-reducible.

Next, we bound the size of $T_{\leq 1}$, which implies a bound on the sizes of $T_{\geq 3}$ and $\mathcal{P}$. To do so, we simply invoke Lemma 13 by setting $X = S \cup R$. Since $|S| < gk$, $g = \frac{48 \log k}{\log \log k} + 6$ and $|R| \leq 3ck \log k + 2ck \log^{1.5} k$, we get that $|T_{\leq 1}| \leq |S \cup R|^2(2|S \cup R| + 1) = k^{\mathcal{O}(1)}$. Since in a forest, it holds that $|T_{\geq 3}| < |T_{\leq 1}|$, the bound on $|T_{\geq 3}|$ follows. Moreover, in a forest, it also holds that $|\mathcal{P}| < |T_{\leq 1}| + |T_{\geq 3}|$ – if we arbitrarily root each tree in the forest at a leaf, one end vertex of a path in $\mathcal{P}$ will be a parent of a different vertex from $T_{\leq 1} \cup T_{\geq 3}$ – the bound on $|\mathcal{P}|$ follows as well. ◀

5 Guessing Permutations

This section is devoted to proving the following lemma. Note that assuming the statement of the lemma, the only remaining task (to prove Theorem 1) is to develop an algorithm running in time $\mathcal{O}(2^{|V|} \cdot \text{poly}(|V|))$ and using polynomial space, which we present in Section 6.

▶ **Lemma 15.** *Given an instance (G, k) of* Cycle Packing*, we can in time $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})} \cdot |V|$ and polynomial space compute $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})}$ instances of* Cycle Packing *of the form (G', k), where the number of vertices in G' is bounded by $\mathcal{O}(k \log^{1.5} k)$, such that (G, k) is a yes-instance if and only if at least one of the instances (G', k) is a yes-instance.*[2]

Proof. We fix $g = \frac{48 \log k}{\log \log k} + 6$. Using Lemma 10, we first compute a set S in time $k^{\mathcal{O}(1)} \cdot |V|$. Then, we guess which vertices to delete from S – that is, which vertices do not participate in a solution – in time $\mathcal{O}(2^{gk}) = 2^{\mathcal{O}(\frac{k \log k}{\log \log k})}$. Here, guesses refer to different choices which lead to the construction of different instances of Cycle Packing that are returned at the end (recall that we are allowed to return up to $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})}$ different instances). Combining Lemma 10 and Corollary 12, we now have a set $S \subseteq V$ such that $|S| = \mathcal{O}(\frac{k \log k}{\log \log k})$, and $|\mathsf{reduce}(G - S)| = \mathcal{O}(k \log^{1.5} k)$.

[2] In practice, to use polynomial space, we output the instances one-by-one.

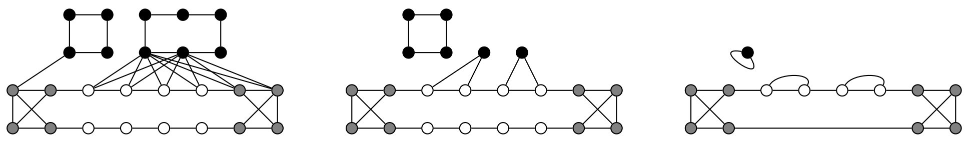
Figure 2 [Left] A graph G (not all edges shown), the set S (in black), the set R (in gray), and the set $T = G - R - S$ (in white). [Center] the graph obtained after guessing vertices in S and their neighbors in a solution. [Right] Example of a reduced instance.

Applying Lemma 14 with $R = \mathsf{img}(\mathsf{reduce}(G - S)) \subseteq (V \setminus S)$, we get a forest $T = G - (S \cup R)$ such that for every maximal degree-two path in $\mathcal{P}$ there are at most two vertices on the path having neighbors in R (in the graph $G - S$). In addition, the size of R is bounded by $\mathcal{O}(k \log^{1.5} k)$, and the sizes $|T_{\leq 1}|$, $|\mathcal{P}|$ and $|T_{\geq 3}|$ are bounded by $k^{\mathcal{O}(1)}$ (see Figure 1).

For every vertex in S (which is assumed to participate in a solution), we now guess its two neighbors in a solution (see Figure 2). Note however that we only have a (polynomial in k) bound for $|S|$, $|R|$, $|T_{\leq 1}|$, $|\mathcal{P}|$ and $|T_{\geq 3}|$, but not for the length of paths in $\mathcal{P}$ and therefore not for the entire graph G. We let $Z_{\mathcal{P}}$ denote the set of vertices in $V(\mathcal{P})$ having neighbors in R. The size of $Z_{\mathcal{P}}$ is at most $2|\mathcal{P}|$. Moreover, we let $\mathcal{P}^\star$ denote the set of paths obtained after deleting $Z_{\mathcal{P}}$ from $\mathcal{P}$. Note that the size of $\mathcal{P}^\star$ is upper bounded by $|\mathcal{P}| + |Z_{\mathcal{P}}| \leq 3|\mathcal{P}|$, and that vertices in $V(\mathcal{P}^\star)$ are adjacent only to vertices in $V(\mathcal{P}^\star) \cup Z_{\mathcal{P}} \cup S$. Now, we create a set of "objects", $O = S \cup R \cup T_{\leq 1} \cup T_{\geq 3} \cup Z_{\mathcal{P}} \cup \mathcal{P}^\star$. We also denote $\widetilde{O} = O \setminus \mathcal{P}^\star$. We then guess, for each vertex in S, which two objects in O constitute its neighbors, denoted by $\ell(v)$ and $r(v)$, in a solution. It is possible that $\ell(v) = r(v)$. Since $|O| = k^{\mathcal{O}(1)}$, we can perform these guesses in $k^{\mathcal{O}(\frac{k \log k}{\log \log k})}$, or equivalently $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})}$, time. We can assume that if $\ell(v) \in \widetilde{O}$, then $\ell(v)$ is a neighbor of v, and otherwise v has a neighbor on the path $\ell(v)$, else the current guess is not correct, and we need not try finding a solution subject to it. The same claim holds for $r(v)$. If $\ell(v) = r(v) \in \widetilde{O}$, then $\{v, \ell(v)\}$ is an edge of multiplicity two, and otherwise if $\ell(v) = r(v)$, then v has (at least) two neighbors on the path $\ell(v)$.

Next, we fix some arbitrary order on $\mathcal{P}^\star$, and for each path in $\mathcal{P}^\star$, we fix some arbitrary orientation. We let $S^\star$ denote the multiset containing two occurrences of every vertex $v \in S$, denoted by v_ℓ and v_r. We guess an order of the vertices in $S^\star$. The time spent for guessing such an ordering is bounded by $|S|!$, which in turn is bounded by $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})}$. The ordering, assuming it is guessed correctly, satisfies the following conditions. For each path $P \in \mathcal{P}^\star$, we let $\ell(P)$ and $r(P)$ denote the sets of vertices $v \in S$ such that $\ell(v) \in V(P)$ and $r(v) \in V(P)$, respectively. Now, for any two vertices $u, v \in \ell(P)$, if $u_\ell < v_\ell$ according to the order that we guessed, then the neighbor $\ell(u)$ of u appears before the neighbor $\ell(v)$ of v on P. Similarly, for any two vertices $u, v \in r(P)$, if $u_r < v_r$, then $r(u)$ appears before $r(v)$ on P. Finally, for any two vertices $u \in \ell(P)$ and $v \in r(P)$, if $u_\ell < v_r$, then $\ell(u)$ appears before $r(v)$ on P, and otherwise $r(v)$ appears before $\ell(u)$ on P.

Given a correct guess of $\ell(v)$ and $r(v)$, for each v in S, as well as a correct guess of a permutation of $S^\star$, for each path in $\mathcal{P}^\star$, we let $\{x_v, y_v\}$ denote the two guessed neighbors of a vertex v in S. Note that if $\ell(v)$ $(r(v))$ is in $\widetilde{O}$ then $x_v = \ell(v)$ $(y_v = r(v))$. Otherwise, we assign neighbors to a vertex by a greedy procedure which agrees with the guessed permutation on $S^\star$; that is, for every path $P \in \mathcal{P}^\star$, we iterate over $\ell(P) \cup r(P)$ according to the guessed order, and for each vertex in it, assign its first neighbor on P that is after the last vertex that has already been assigned (if such a vertex does not exist, we determine that the current guess is incorrect and proceed to the next one). We let $X = \{x_v \mid v \in S\}$ and $Y = \{y_v \mid v \in S\}$. We also let E_S be the set of edges incident on a vertex in S, and we let $E' = \{\{x_v, y_v\} \mid v \in S\}$

denote the set of all pairs of guesses. Finally, to obtain an instance (G', k), we delete the vertex set $W = S \setminus (X \cup Y)$ from G, we delete the edge set E_S from G, we add instead the set of edges E', and finally we apply the reduce operator, i.e. $G' = \mathsf{reduce}((G - W - E_S) + E')$.

▶ **Claim 16.** *Let (G', k) be one of the instances generated by the above procedure. Then, the number of vertices in G' is bounded by $\mathcal{O}(k \log^{1.5} k)$.*

Proof. Recall that by Corollary 12, we know that $|\mathsf{reduce}(G - S)| = \mathcal{O}(k \log^{1.5} k)$. Moreover, we have $|E'| = |S| = \mathcal{O}(\frac{k \log k}{\log \log k})$. Combining Observation 9 with the fact that $G' = \mathsf{reduce}((G-W-E_S)+E')$, we get $|\mathsf{reduce}((G-W-E_S)+E')| \leq |\mathsf{reduce}(G-W-E_S)|+2|E'|$. Since in $G-W-E_S$ all vertices of S have degree zero, $|\mathsf{reduce}(G-W-E_S)| \leq |\mathsf{reduce}(G-S)|$. Hence, we conclude that $|\mathsf{reduce}((G - W - E_S) + E')| = \mathcal{O}(k \log^{1.5} k)$, as needed. ◀

▶ **Claim 17.** *(G, k) is a yes-instance if and only if at least one of the generated instances (G', k) is a yes-instance.*

Proof. Assume that (G, k) is a yes-instance and let $\mathcal{C} = \{C_1, C_2, \ldots\}$ be an optimal cycle packing, i.e set of maximum size of vertex-disjoint cycles, in G. Note that if no cycle in $\mathcal{C}$ intersects with S then $\mathcal{C}$ is also an optimal cycle packing in $G - S$. By the safeness of our reduction rules, $\mathcal{C}$ is also an optimal cycle packing in $\mathsf{reduce}(G - S)$. Since we generate one instance for every possible intersection between an optimal solution and S, the case where no vertex from S is picked corresponds to the instance (G', k), with $G' = \mathsf{reduce}(G - S)$. Hence, in what follows we assume that some cycles in $\mathcal{C}$ intersect with S. Consider any cycle C which intersects with S and let $P_C = \{u_0, u_1, \ldots, u_f\}$ denote any path on this cycle such that $u_0, u_f \notin S$ but $u_i \in S$ for $0 < i < f$. We claim that, for some G', all such paths will be replaced by edges of the form $\{u_0, u_f\}$ in $\mathsf{reduce}((G - W - E_S) + E')$. Again, due to our exhaustive guessing, for some G' we would have guessed, for each i, $\ell(u_i) = u_{i-1}$ and $r(u_i) = u_{i+1}$. Consequently, $P_C \setminus \{u_0, u_f\}$ is a degree-two path in $(G - W - E_S) + E'$ and therefore an edge in $\mathsf{reduce}((G - W - E_S) + E')$. Using similar arguments, it is easy to show that if C is completely contained in S then this cycle is contained in G' as a loop on some vertex of the cycle.

For the other direction, let (G', k) be a yes-instance and let $\mathcal{C}' = \{C'_1, C'_2, \ldots\}$ be an optimal cycle packing in G'. We assume, without loss of generality, that $\mathcal{C}'$ is a cycle packing in $(G - W - E_S) + E'$, as one can trace back all reduction rules to obtain the graph $(G - W - E_S) + E'$. If no cycle in $\mathcal{C}'$ uses an edge $\{u_0, u_f\} \in E'$ then we are done, as $(G - W - E_S)$ is a subgraph of G. Otherwise, we claim that all such edges either exist in G or can be replaced by vertex disjoint paths $P = \{u_0, u_1, \ldots, u_f\}$ (on at least three vertices) in G such that $u_i \in S$ for $0 < i < f$. If either u_0 or u_f is in $X \cup Y \subseteq S$ then the former case holds. It remains to prove the latter case. Recall that for every vertex in S we guess its two neighbors from $O = S \cup R \cup T_{\leq 1} \cup T_{\geq 3} \cup Z_{\mathcal{P}} \cup \mathcal{P}^\star$. Hence, if $\{u_0, u_f\} \subseteq \widetilde{O} = O \setminus \mathcal{P}^\star$ then one can easily find a path (or singleton) in $G[S]$ to replace this edge by simply backtracking the neighborhood guesses. Now assume that $\{u_0, u_f\} \not\subseteq \widetilde{O}$ and recall that no vertex in a path in $\mathcal{P}^\star$ can have neighbors in R. Hence, any cycle containing such an edge must intersect with S (in G). Assuming we have correctly guessed the neighbors of vertices in S (as well as a permutation for $\mathcal{P}^\star$), we can again replace this edge with a path in S. ◀

Combining Claims 16 and 17 concludes the proof of the theorem. ◀

6 Dynamic Programming and Inclusion-Exclusion

Finally, we give an exact exponential-time algorithm for CYCLE PACKING. For this purpose, we use DP and the principle of inclusion-exclusion, inspired by the work of Nederlof [32]. Due to space constraints, the details are given in the full version of the paper.

▶ **Lemma 18.** *There exists a (deterministic) polynomial-space algorithm that in time $\mathcal{O}(2^{|V|} \cdot \mathrm{poly}(|V|))$ solves* CYCLE PACKING. *In case a solution exists, it also outputs a solution.*

We would like to mention that if one does not care about polynomial space, then Lemma 18 can be obtained by a straightforward dynamic programming on subsets.

7 Conclusion

In this paper we have beaten the best known $2^{\mathcal{O}(k \log^2 k)} \cdot |V|$-time algorithm for CYCLE PACKING that is a consequence of the Erdős-Pósa theorem. For this purpose, we developed a deterministic algorithm that solves CYCLE PACKING in time $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})} \cdot |V|$. Two additional advantageous properties of our algorithm is that its space complexity is polynomial in the input size and that in case a solution exists, it outputs a solution (in time $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})} \cdot |V|$). Our technique relies on combinatorial arguments that may be of independent interest. These arguments allow us to translate any input instance of CYCLE PACKING into $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})}$ instances of CYCLE PACKING whose sizes are small and can therefore be solved efficiently.

It remains an intriguing open question to discover the "true" running time, under reasonable complexity-theoretic assumptions, in which one can solve CYCLE PACKING on general graphs. In particular, we would like to pose the following question: Does there exist a $2^{\mathcal{O}(k \log k)} \cdot |V|^{\mathcal{O}(1)}$-time algorithm for CYCLE PACKING? This is true for graphs of bounded maximum degree as one can easily bound the number of vertices by $\mathcal{O}(k \log k)$ and then apply Lemma 18. Moreover, Bodlaender et al. [4] proved that this is also true in case one seeks k edge-disjoint cycles rather than k vertex-disjoint cycles. On the negative side, recall that (for general graphs) the bound $f(k) = \mathcal{O}(k \log k)$ in the Erdős-Pósa theorem is essentially tight, and that it is unlikely that CYCLE PACKING is solvable in time $2^{o(tw \log tw)} \cdot |V|^{\mathcal{O}(1)}$ [11]. However, we do not rule out the existence of an algorithm solving CYCLE PACKING in time $2^{\mathcal{O}(fvs)} \cdot |V|^{\mathcal{O}(1)}$. Thus, the two most natural attempts to obtain a $2^{\mathcal{O}(k \log k)} \cdot |V|^{\mathcal{O}(1)}$-time algorithm – either replacing the bound $\mathcal{O}(k \log k)$ in the Erdős-Pósa theorem by $\mathcal{O}(k)$ or speeding-up the computation based on DP to run in time $2^{\mathcal{O}(tw)} \cdot |V|^{\mathcal{O}(1)}$ – lead to a dead end.

Acknowledgements. We would like to thank the reviewers for several suggestions and insightful remarks that have improved the presentation of the paper.

References

1 Noga Alon, Shlomo Hoory, and Nathan Linial. The Moore bound for irregular graphs. *Graphs and Combinatorics*, 18(1):53–57, 2002. doi:10.1007/s003730200002.

2 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernel bounds for path and cycle problems. *Theor. Comput. Sci.*, 511:117–136, 2013. doi:10.1016/j.tcs.2012.09.006.

3 Hans L. Bodlaender, Eelko Penninkx, and Richard B. Tan. A linear kernel for the k-disjoint cycle problem on planar graphs. In Seok-Hee Hong, Hiroshi Nagamochi, and Takuro Fukunaga, editors, *Algorithms and Computation, 19th International Symposium, ISAAC 2008,*

Gold Coast, Australia, December 15-17, 2008. Proceedings, volume 5369 of *Lecture Notes in Computer Science*, pages 306–317. Springer, 2008. doi:10.1007/978-3-540-92182-0_29.

4 Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theor. Comput. Sci.*, 412(35):4570–4578, 2011. doi:10.1016/j.tcs.2011.04.039.

5 H. L. Bodlaender. On disjoint cycles. *Int. J. Found. Comput. Sci.*, 5(1):59–68, 1994.

6 Chandra Chekuri and Julia Chuzhoy. Large-treewidth graph decompositions and applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 291–300. ACM, 2013. doi:10.1145/2488608.2488645.

7 Chandra Chekuri and Julia Chuzhoy. Polynomial bounds for the grid-minor theorem. *J. ACM*, 63(5):40:1–40:65, 2016.

8 Julia Chuzhoy. Excluded grid theorem: Improved and simplified. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 645–654. ACM, 2015. doi:10.1145/2746539.2746551.

9 Julia Chuzhoy. Excluded grid theorem: Improved and simplified (invited talk). In Rasmus Pagh, editor, *15th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2016, June 22-24, 2016, Reykjavik, Iceland*, volume 53 of *LIPIcs*, pages 31:1–31:1. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.SWAT.2016.31.

10 M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized algorithms*. Springer, 2015.

11 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 150–159. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.23.

12 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

13 Frederic Dorn, Fedor V. Fomin, and Dimitrios M. Thilikos. Catalan structures and dynamic programming in h-minor-free graphs. *J. Comput. Syst. Sci.*, 78(5):1606–1622, 2012. doi:10.1016/j.jcss.2012.02.004.

14 R. Downey and M. Fellows. *Fundamentals of parameterized complexity*. Springer, 2013.

15 P. Erdős and L. Pósa. On independent circuits contained in a graph. *Canad. J. Math.*, 17:347–352, 1965.

16 Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. Bidimensionality and EPTAS. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 748–759. SIAM, 2011. doi:10.1137/1.9781611973082.59.

17 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Bidimensionality and kernels. In Moses Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 503–510. SIAM, 2010. doi:10.1137/1.9781611973075.43.

18 Zachary Friggstad and Mohammad R. Salavatipour. Approximability of packing disjoint cycles. *Algorithmica*, 60(2):395–400, 2011. doi:10.1007/s00453-009-9349-5.

19 Martin Grohe and Magdalena Grüber. Parameterized approximability of the disjoint cycle problem. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, volume 4596 of *Lecture Notes in Computer Science*, pages 363–374. Springer, 2007. doi:10.1007/978-3-540-73420-8_33.

20 R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

21 Alon Itai and Michael Rodeh. Finding a minimum circuit in a graph. *SIAM J. Comput.*, 7(4):413–423, 1978. doi:10.1137/0207033.

22 Felix Joos. Parity linkage and the erdős-pósa property of odd cycles through prescribed vertices in highly connected graphs. In Ernst W. Mayr, editor, *Graph-Theoretic Concepts in Computer Science – 41st International Workshop, WG 2015, Garching, Germany, June 17-19, 2015, Revised Papers*, volume 9224 of *Lecture Notes in Computer Science*, pages 339–350. Springer, 2015. doi:10.1007/978-3-662-53174-7_24.

23 Naonori Kakimura and Ken-ichi Kawarabayashi. Half-integral packing of odd cycles through prescribed vertices. *Combinatorica*, 33(5):549–572, 2013. doi:10.1007/s00493-013-2865-6.

24 Naonori Kakimura, Ken-ichi Kawarabayashi, and Dániel Marx. Packing cycles through prescribed vertices. *J. Comb. Theory, Ser. B*, 101(5):378–381, 2011. doi:10.1016/j.jctb.2011.03.004.

25 Ken-ichi Kawarabayashi and Yusuke Kobayashi. Fixed-parameter tractability for the subset feedback set problem and the s-cycle packing problem. *J. Comb. Theory, Ser. B*, 102(4):1020–1034, 2012. doi:10.1016/j.jctb.2011.12.001.

26 Ken-ichi Kawarabayashi, Daniel Král', Marek Krcál, and Stephan Kreutzer. Packing directed cycles through a specified vertex set. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 365–377. SIAM, 2013. doi:10.1137/1.9781611973105.27.

27 Ken-ichi Kawarabayashi and Atsuhiro Nakamoto. The erdos-pósa property for vertex- and edge-disjoint odd cycles in graphs on orientable surfaces. *Discrete Mathematics*, 307(6):764–768, 2007. doi:10.1016/j.disc.2006.07.008.

28 Ken-ichi Kawarabayashi and Bruce A. Reed. Highly parity linked graphs. *Combinatorica*, 29(2):215–225, 2009. doi:10.1007/s00493-009-2178-y.

29 Ken-ichi Kawarabayashi and Paul Wollan. Non-zero disjoint cycles in highly connected group labelled graphs. *J. Comb. Theory, Ser. B*, 96(2):296–301, 2006. doi:10.1016/j.jctb.2005.08.001.

30 Michael Krivelevich, Zeev Nutov, Mohammad R. Salavatipour, Jacques Yuster, and Raphael Yuster. Approximation algorithms and hardness results for cycle packing problems. *ACM Trans. Algorithms*, 3(4):48, 2007. doi:10.1145/1290672.1290685.

31 D. Lokshtanov, F. Panolan, M.S. Ramanujan, and S. Saurabh. Lossy kernelization. *arXiv:1604.04111v2*, to appear in STOC 2017.

32 Jesper Nederlof. Fast polynomial-space algorithms using inclusion-exclusion. *Algorithmica*, 65(4):868–884, 2013. doi:10.1007/s00453-012-9630-x.

33 M. Pontecorvi and Paul Wollan. Disjoint cycles intersecting a set of vertices. *J. Comb. Theory, Ser. B*, 102(5):1134–1141, 2012. doi:10.1016/j.jctb.2012.05.004.

34 Venkatesh Raman, Saket Saurabh, and C.R. Subramanian. Faster fixed parameter tractable algorithms for finding feedback vertex sets. *ACM Trans. Algorithms*, 2(3):403–415, 2006. doi:10.1145/1159892.1159898.

35 Dieter Rautenbach and Bruce A. Reed. The Erdős-Pósa property for odd cycles in highly connected graphs. *Combinatorica*, 21(2):267–278, 2001. doi:10.1007/s004930100024.

36 Bruce A. Reed. Mangoes and blueberries. *Combinatorica*, 19(2):267–296, 1999. doi:10.1007/s004930050056.

37 Neil Robertson and Paul D. Seymour. Graph minors. V. Excluding a planar graph. *J. Comb. Theory, Ser. B*, 41(1):92–114, 1986. doi:10.1016/0095-8956(86)90030-4.

38 Neil Robertson and Paul D. Seymour. Graph minors .xiii. the disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006.

39 Aleksandrs Slivkins. Parameterized tractability of edge-disjoint paths on directed acyclic graphs. *SIAM J. Discrete Math.*, 24(1):146–157, 2010. doi:10.1137/070697781.

40 Carsten Thomassen. The Erdős-Pósa property for odd cycles in graphs of large connectivity. *Combinatorica*, 21(2):321–333, 2001. doi:10.1007/s004930100028.