Packing Cycles Faster Than Erdős-Pósa

Daniel Lokshtanov[1], Amer E. Mouawad[1], Saket Saurabh[1,2], and Meirav Zehavi[1]

1   University of Bergen, Norway
    {daniel.lokshtanov,a.mouawad,meirav.zehavi}@ii.uib.no
2   The Institute of Mathematical Sciences, Chennai, India
    saket@imsc.res.in

───── **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.

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 [16], 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 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 [15] and [11].

   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 [39],[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 [3]. 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 [16]. 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., [11]). 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 [15] and [11]. 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 [16] 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. [12] 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) [21] 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 [31], and it is quasi-NP-hard to approximate within a factor of $\mathcal{O}(\log^{\frac{1}{2}-\epsilon} |V|)$ for any $\epsilon > 0$ [19]. In the context of kernelization with respect to the parameter k, CYCLE PACKING does not admit a polynomial kernel unless NP $\subseteq$ coNP/Poly [6]. Recently, Lokshtanov et al. [32] obtained a 6-approximate kernel with $\mathcal{O}((k \log k)^2)$ vertices along with a $(1+\epsilon)$-approximate kernel with $k^{\mathcal{O}(f(\epsilon) \log k)}$ 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 [6].

Focusing on structural parameters, Bodlaender et al. [4] obtained polynomial kernels with respect to the vertex cover number, vertex-deletion distance to a cluster graph and the max leaf number. In planar graphs, Bodlaender et al. [5] 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. [14] 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. [18] showed that CYCLE PACKING admits a linear kernel, and Fomin et al. [17] showed that it also admits an EPTAS. When the input graph is a directed graph, CYCLE PACKING is W[1]-hard [40], but it admits an FPT approximation scheme [20]. In fact, CYCLE PACKING in

---

[1] The paper [39] was already available as a manuscript in 1986 (see, e.g., [3]).

directed graphs was the first W[1]-hard problem shown to admit such a scheme. Krivelevich et al. [31] 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 [37, 41, 36, 30, 28, 29]. Another well-known variant, where the cycles need to contain a prescribed set of vertices, has also been extensively investigated [25, 34, 26, 24, 27]. Furthermore, a combination of these two variants has been considered in [24, 23].

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 [38] 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 [7] 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 [8], which states that every graph of treewidth at least $f(k) = \mathcal{O}(k^{98} \cdot \text{polylog}(k))$ contains the $k \times k$ grid as a minor (the constant 98 has been improved to 36 in [9] and to 19 in [10]). Following the seminal work by Robertson and Seymour [38], 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 [3]. 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.*

Our technique relies on several combinatorial arguments that might be of independent interest, and whose underlying ideas might be relevant to the design of other parameterized algorithms. Let us now outline the structure of our proof, specifying the main ingredients that we develop along the way.

- First, we show that in time linear in $|V|$, it is easy to bound $|E|$ by $\mathcal{O}(k \log k \cdot |V|)$ (Assumption 1).
- Second, we give an algorithmic version of the Erdős-Pósa theorem that runs in time linear in $|V|$ and which outputs either a solution or a small feedback vertex set (Theorem 2).

- Then, we show that given a graph $G = (V, E)$ and a feedback vertex set F, a shortest cycle in G can be found in time $\mathcal{O}(|F| \cdot (|V| + |E|))$ (Lemma 3).
- We proceed by interleaving an application of a simple set of reduction rules (Rules A1, A2 and A3) with a computation of a "short" cycle. Thus, given some $g > 6$, we obtain a set S of size smaller than gk such that the girth of the "irreducible component" of $G - S$ is larger than g (Lemma 5). Here, the irreducible component of $G - S$ is the graph obtained from $G - S$ by applying our reduction rules.
- Next, we show that the number of vertices in the above mentioned irreducible component is actually "small" — for some fixed constant c, it can be bounded by $(2ck \log k)^{1 + \frac{6}{g-6}} + 3ck \log k$ (Lemma 6). The choice of $g = \frac{48 \log k}{\log \log k} + 6$ results in the bound $3ck \log k + 2ck \log^{1.5} k$ (Corollary 7).[2]
- Now, we return to examine the graph $G - S$ rather than only its irreducible component. The necessity of this examination stems from the fact that our reduction rules, when applied to $G - S$ rather than G, do not preserve solutions. We first give a procedure which given any set X, modifies the graph $G - X$ in a way that both preserves solutions and gets rid of many leaves (Lemma 8). We then use this procedure to bound the number of leaves, as well as other "objects", in the reducible component of $G - S$ (Lemma 9).
- At this point, the graph G may still contain many vertices: the reducible component of $G - S$ may contain "long" induced paths (which are not induced paths in G). We show that the length of these paths can be shortened by "guessing" permutations that provide enough information describing the relations between these paths and the vertices in S. Overall, we are thus able to bound the entire vertex-set of G by $\mathcal{O}(k \log^{1.5} k)$ in time $2^{\mathcal{O}(\frac{k \log^2 k}{\log \log k})} \cdot |V|$ and polynomial space (Lemma 10).
- Finally, we apply a DP scheme (Lemma 11). Here, to ensure that the space complexity is polynomial in the input size, we rely on the principle of inclusion-exclusion.

2   Preliminaries

We use standard terminology from the book of Diestel [13] for those graph-related terms that are not explicitly defined here. We only consider finite graphs possibly having self-loops and multi-edges. Moreover, 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.

---

[2] We found these constants as the most natural ones to obtain a clean proof of *any* bound of the form $\mathcal{O}(\frac{k \log^2 k}{\log \log k})$ (that is, the constants were not optimized to obtain the bound $3ck \log k + 2ck \log^{1.5} k$).

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 $0 \leq i < \ell$. A *cycle* in a graph is a sequence of distinct vertices $v_0, v_1, \ldots, v_\ell$ such that $\{v_i, v_{(i+1) \mod \ell+1}\}$ is an edge for all $0 \leq i \leq \ell$. 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 [16].

▶ **Proposition 1** ([16]). *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 1, 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 $\mathcal{O}(k \log k \cdot |V|)$ is described in Appendix A.

▶ **Assumption 1.** *We assume that $|E| = \mathcal{O}(k \log k \cdot |V|)$.*

Now, we state our algorithmic version of Proposition 1. The proof partially builds upon the proof of the Erdős-Pósa theorem in the book [13], and it is given in Appendix B.

▶ **Theorem 2.** *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 2** ([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 [22] 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 in Appendix C.

▶ **Lemma 3.** *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 6) to bound the size of a graph we obtain after performing simple preprocessing operations as well as repetitive removal of short cycles.

▶ **Proposition 3** ([35], 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 4.** 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 1.** *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 5 and 6. Intuitively, S will contain vertices of "short" cycles in the input graph, where short will be defined later.

▶ **Lemma 5.** *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 2 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 3 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 $\mathsf{girth}(\mathsf{reduce}(G - S)) > g$. Since the algorithm of Theorem 2 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 6.** *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 5 (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., [35], 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 5, we know that $\mathsf{girth}(G') > g > 6$.

Next, we apply Proposition 3 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 $\mathsf{girth}(G') > 6$, $x_u \neq x_v$ 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 3, 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 $\mathsf{girth}(G') > g > 6$, 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 2, $G^\star$ 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

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

This completes the proof. ◀

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

▶ **Corollary 7.** *Given a (multi) graph $G = (V, E)$ and an integer $k > 0$, let S denote the set obtained after applying the algorithm of Lemma 5 with $g = \frac{48 \log k}{\log \log k} + 6$ (assuming no k vertex-disjoint cycles obtained). Then $|\,\text{reduce}(G - S)| \leq 3ck \log k + 2ck \log^{1.5} k$.*

Proof. By Lemma 6, $|\,\text{reduce}(G-S)| \leq (2ck \log k)^{1 + \frac{\log \log k}{8 \log k}} + 3ck \log k$. 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 $\text{girth}(\text{reduce}(G - S)) > g$, $|S| < gk$, and $|\,\text{reduce}(G - S)| \leq 3ck \log k + 2ck \log^{1.5} k$.

Even though the number of vertices in $\text{reduce}(G - S)$ 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. [32] (Lemma 5.2). We give a full proof in Appendix D.

▶ **Lemma 8.** *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 8, 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.
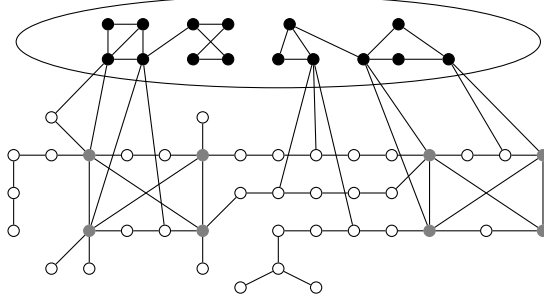
Figure 1 A graph G (not all edges shown), the set S (in black), the set R (in gray), and the set $T = G - R - S$ (in white).

▶ **Lemma 9.** *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 $\mathsf{girth}(\mathsf{reduce}(G-S)) > 6$, 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, 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 8 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 $\mathcal{O}(2^{|V|} \cdot \mathsf{poly}(|V|))$ and using polynomial space, which we present in Section 6.

▶ **Lemma 10.** *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.*[3]

Proof. We fix $g = \frac{48 \log k}{\log \log k} + 6$. Using Lemma 5, 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 5 and Corollary 7, we now have a set $S \subseteq V$ such that $|S| = \mathcal{O}(\frac{k \log k}{\log \log k})$, and $|\mathsf{reduce}(G - S)| = \mathcal{O}(k \log^{1.5} k)$.

Applying Lemma 9 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 $\mathcal{P}$ an 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

---

[3] In practice, to use polynomial space, we output the instances one-by-one.
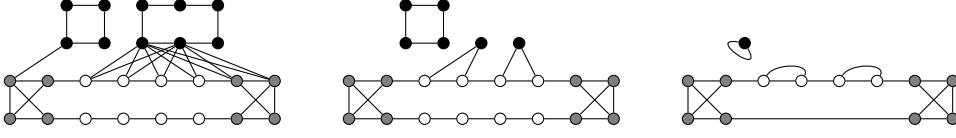
■ **Figure 2** [Left] a graph G (not all edges shown), the set S (in black), the set R (in gray), and the set $T = G - R - S$ (in white). [Center] the graph obtained after guessing vertices in S and their neighbors in a solution. [Right] example of a reduced instance.

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 1.** *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 7, 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 1 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 2.** *(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 $\{u_0, u_f\} \nsubseteq \widetilde{O}$ 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 1 and 2 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 [33]. Due to space constraints, the details are given in Appendix E.

▶ **Lemma 11.** *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.*

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 11. Moreover, Bodlaender et al. [6] 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 $2^{o(tw \log tw)} \cdot |V|^{\mathcal{O}(1)}$ [12].[4] 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.

---

[4] However, we do not rule out the existence of an algorithm solving CYCLE PACKING in time $2^{\mathcal{O}(fvs)} \cdot |V|^{\mathcal{O}(1)}$.

References

1 Alon, N., Hoory, S., Linial, N.: The Moore bound for irregular graphs. Graphs and Combinatorics 18(1), 53–57 (2002)

2 Bar-Yehuda, R., Geiger, D., Naor, J., Roth, R.M.: Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and bayesian inference. SIAM J. Comput. 27(4), 942–959 (1998)

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

4 Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Kernel bounds for path and cycle problems. Theor. Comput. Sci. 511, 117–136 (2013)

5 Bodlaender, H.L., Penninkx, E., Tan, R.B.: A linear kernel for the k-disjoint cycle problem on planar graphs. In: ISAAC. pp. 306–317 (2008)

6 Bodlaender, H.L., Thomasse, S., Yeo, A.: Kernel bounds for disjoint cycles and disjoint paths. Theor. Comput. Sci. 412, 4570–4578 (2011)

7 Chekuri, C., Chuzhoy, J.: Large-treewidth graph decompositions and applications. In: STOC. pp. 291–300 (2013)

8 Chekuri, C., Chuzhoy, J.: Polynomial bounds for the grid-minor theorem. In: STOC. pp. 60–69 (2014)

9 Chuzhoy, J.: Excluded grid theorem: Improved and simplified. In: STOC. pp. 645–654 (2015)

10 Chuzhoy, J.: Excluded grid theorem: Improved and simplified. In: SWAT (see also arXiv:1602.02629). pp. 31:1–31:1 (2016)

11 Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized algorithms. Springer (2015)

12 Cygan, M., Nederlof, J., Pilipczuk, M., Pilipczuk, M., Rooij, J M M Wojtaszczyk, J.O.: Solving connectivity problems parameterized by treewidth in single exponential time. In: FOCS. pp. 150–159 (2011)

13 Diestel, R.: Graph Theory, 4th Edition, Graduate texts in mathematics, vol. 173. Springer (2012)

14 Dorn, F., Fomin, F.V., Thilikos, D.M.: Catalan structures and dynamic programming in h-minor-free graphs. Journal of Computer and System Sciences 78, 1606–1622 (2012)

15 Downey, R., Fellows, M.: Fundamentals of parameterized complexity. Springer (2013)

16 Erdős, P., Pósa, L.: On independent circuits contained in a graph. Canad. J. Math. 17, 347–352 (1965)

17 Fomin, F.V., Lokshtanov, D., Raman, V., Saurabh, S.: Bidimensionality and EPTAS. In: SODA. pp. 748–759 (2011)

18 Fomin, F.V., Lokshtanov, D., Saurabh, S., Thilikos, D.M.: Bidimensionality and kernels. In: SODA. pp. 503–510 (2010)

19 Friggstad, Z., Salavatipour, M.R.: Approximability of packing disjoint cycles. Algorithmica 60, 395–400 (2011)

20 Grohe, M., Gruber, M.: Parameterized approximability of the disjoint cycle problem. In: ICALP. pp. 363–374 (2007)

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

22 Itai, A., Rodeh, M.: Finding a minimum circuit in a graph. SIAM J. Comput. 7(4), 413–423 (1978)

23 Joos, F.: Parity linkage and the Erdős-Pósa property of odd cycles through prescribed vertices in highly connected graphs. In: WG. pp. 339–350 (2015)

24 Kakimura, N., Kawarabayashi, K.: Half-integral packing of odd cycles through prescribed vertices. Combinatorica 33(4), 549–572 (2013)

25   Kakimura, N., Kawarabayashi, K., Marx, D.: Packing cycles through prescribed vertices. J. Comb. Theory Series B 101, 378–381 (2011)

26   Kawarabayashi, K., Kobayashi, Y.: Fixed-parameter tractability for the subset feedback set problem and the S-cycle packing problem. J. Comb. Theory Series B 102(4), 1020–1034 (2012)

27   Kawarabayashi, K., Kral, D., Krcal, M., Kreutzer, S.: Packing directed cycles through a specified vertex set. In: SODA. pp. 365–377 (2013)

28   Kawarabayashi, K., Nakamoto, A.: The Erdős-Pósa property for vertex- and edge-disjoint odd cycles in graphs on orientable surfaces. Discrete Mathematics 307(6), 764–768 (2007)

29   Kawarabayashi, K., Reed, B.: Highly parity linked graphs. Combinatorica 29, 215–225 (2009)

30   Kawarabayashi, K., Wollan, P.: Non-zero disjoint cycles in highly connected group labelled graphs. J. Comb. Theory Series B 96, 296–301 (2006)

31   Krivelevich, M., Nutov, Z., Salavatipour, M.R., Yuster, J., Yuster, R.: Approximation algorithms and hardness results for cycle packing problems. ACM Transactions on Algorithms 3(4) (2007)

32   Lokshtanov, D., Panolan, F., Ramanujan, M.S., Saurabh, S.: Lossy kernelization. arXiv:1604.04111v1 (2016)

33   Nederlof, J.: Fast polynomial-space algorithms using inclusion-exclusion. Algorithmica 65(4), 868–884 (2013)

34   Pontecorvi, M., Wollan, P.: Disjoint cycles intersecting a set of vertices. J. Comb. Theory Series B 102(5), 1134–1141 (2012)

35   Raman, V., Saurabh, S., Subramanian, C.R.: Faster fixed parameter tractable algorithms for finding feedback vertex sets. ACM Transactions on Algorithms 2(3), 403–415 (2006)

36   Rautenbach, D., Reed, B.: The Erdős-Pósa property for odd cycles in highly connected graphs. Combinatorica 21, 267–278 (2001)

37   Reed, B.: Mangoes and blueberries. Combinatorica 19, 267–296 (1999)

38   Robertson, N., Seymour, P.D.: Graph minors. V. Excluding a planar graph. J. Comb. Theory Series B 41, 92–114 (1986)

39   Robertson, N., Seymour, P.D.: Graph minors. XIII. the disjoint paths problem. J. Comb. Theory Series B 63(1), 65–110 (1995)

40   Slivkins, A.: Parameterized tractability of edge-disjoint paths on directed acyclic graphs. In: ESA. pp. 482–493 (2003)

41   Thomassen, C.: The Erdős-Pósa property for odd cycles in graphs of large connectivity. Combinatorica 21, 321–333 (2001)

A   Discarding Edges

In this appendix we describe the procedure, mentioned in Section 2 to justify Assumption 1, which given a graph $G = (V, E)$, discards at least $|E| - (2c'k \log k + 1) \cdot |V|$ edges in time $\mathcal{O}(k \log k \cdot |V|)$. We examine the vertices in V in some arbitrary order $\{v_1, v_2, \ldots, v_{|V|}\}$, and initialize a counter x to 0. For each vertex v_i, if $x < (2c'k \log k + 1) \cdot |V|$ then we iterate over the set of edges incident to v_i, and for each edge whose other endpoint is v_j for $j \geq i$, we increase x by 1. Let ℓ be the largest index for which we iterated over the set of edges incident to v_ℓ. We copy V, and initialize the adjacency lists to be empty. Then, we copy the adjacency lists of the vertices $v_1, v_2, \ldots, v_\ell$, where for each adjacency involving vertices v_i and v_j, where $i \leq \ell < j$, we update the adjacency list of v_j to include v_i.

B   Proof of Theorem 2

We fix c as the smallest integer such that $c \geq 150(\log_2 c)$. Let $G = (V, E)$ be a (multi) graph, and let k be a non-negative integer. The objective is to show that in time $k^{\mathcal{O}(1)} \cdot |V|$ we can either output k vertex-disjoint cycles or a feedback vertex set of size at most $ck \log k = r$. We remark that the first part of this proof, which ends at the statement of Lemma 4, follows the proof of the Erdős-Pósa theorem [16] given in the book [13].

We may assume that G contains at least one cycle, since this fact can clearly be checked in time $\mathcal{O}(|V| + |E|)$, and if it is not true, we output an empty set as a feedback vertex set. Now, we construct a maximal subgraph H of G such each vertex in H is of degree 2 or 3 (in H). This construction can be done in time $\mathcal{O}(|V| + |E|)$ (see [2]). Let V_2 and V_3 be the degree-2 and degree-3 vertices in H, respectively. We also compute (in time $\mathcal{O}(|V| + |E|)$) the set $\mathcal{S}$ of connected components of $G - V(H)$. Observe that for each connected component $S \in \mathcal{S}$, there is at most one vertex $v_S \in V_2$ such that there is at least one vertex in S adjacent to v_S, else we obtain a contradiction to the maximality of H as it could have been extended by adding a path from S. We compute (in time $\mathcal{O}(|V| + |E|)$) the vertices v_S, where for each component for which v_S is undefined (since it does not exist), we set $v_S = nil$. Let $V_2^\star \subseteq V_2$ be the set of vertices $v_S \neq nil$ such that v_S has at least two neighbors in S, which is easily found in time $\mathcal{O}(|V| + |E|)$. Observe that if $|V_2^\star| \geq k$, we can output k vertex-disjoint cycles in time $\mathcal{O}(|V| + |E|)$. Thus, we next assume that $|V_2^\star| < k$. Moreover, observe that $V_2^\star \cup V_3$ is a feedback vertex set. Thus, if $|V_2^\star \cup V_3| \leq ck \log k$, we are done. We next assume that $|V_2^\star \cup V_3| > ck \log k$. In particular, it holds that $|V_3| > ck \log k - k \geq (c-1)k \log k$.

Let H^* be the graph obtained from H by contracting, for each vertex in V_2, an edge incident to it. We remark that here we permit the multiplicity of edges to be 3. Then, H^* is a cubic graph whose vertex-set is V_3. To find k vertex-disjoint cycles in G in time $k^{\mathcal{O}(1)} \cdot |V|$, it is sufficient to find k vertex-disjoint cycles in H^* in time $k^{\mathcal{O}(1)} \cdot |V|$, since the cycles in H^* can be translated into cycles in G in time $\mathcal{O}(|V| + |E|)$. We need to rely on the following claim, whose proof is given in the book [13]. We remark that the original claim refers to graphs, but it also holds for multigraphs.

▶ **Proposition 4** ([13]). *If a cubic (multi) graph contains at least $q = 4k(\log k + \log \log k + 4)$ vertices, then it contains k vertex-disjoint cycles.*

Thus, we know that H^* contains k vertex-disjoint cycles, and it remains to find them in time $k^{\mathcal{O}(1)} \cdot |V|$. We now modify H^* to obtain a cubic graph H' on at least q vertices but at most $\mathcal{O}(k \cdot \log k)$ vertices, such that given k vertex-disjoint cycles in H', we can translate them into k vertex-disjoint cycles in H^* in time $\mathcal{O}(|V|)$, which will complete the proof. To

this end, we initially let H' be a copy of H^*. Now, as long as $|V(H')| > (c-1)k \log k + 2$, we perform the following procedure:

1. Choose arbitrarily a vertex $v \in V(H')$.
2. If v has exactly one neighbor u — that is, $\{v, u\}$ is an edge of multiplicity 3 — remove v and u from the graph.
3. Else if v has a neighbor u such that u, in turn, has a neighbor w (which might be v) such that the edge $\{u, w\}$ is of multiplicity 2, then remove u and w from H' and connect the remaining neighbor of u to the remaining neighbor of w by a new edge (which might be a self-loop).
4. Else, let x, y, z be the three distinct neighbors of v. Then, remove v and add an edge between x and y. Now, each vertex is of degree 3, except for z, which is of degree 2, and has two distinct neighbors. Remove z, and connected its two neighbors by an edge.

Since this procedure runs in time $\mathcal{O}(1)$ and each call decreases the number of vertices in the graph, the entire process runs in time $\mathcal{O}(|V|)$. It is also clear that the procedure outputs a cubic graph, and at its end, $(c-1)k \log k \leq |V(H')| \leq (c-1)k \log k + 2$. Thus, to prove the correctness of the process, it is now sufficient to consider graphs H_1 and H_2, where H_2 is obtain from H_1 by applying the procedure once, and show that given a set $\mathcal{C}_2$ of k vertex-disjoint cycles in H_2, we can modify them to obtain a set $\mathcal{C}_1$ of k vertex-disjoint cycles in H_1. Let v be the vertex chosen in the first step. If the condition in the second step was true, we simply let $\mathcal{C}_1 = \mathcal{C}_2$. In the second case, we examine whether the newly added edge belongs to a cycle in the solution in time $\mathcal{O}(1)$ (as we assume that each element in the graph, if it belongs to the solution, has a pointer to its location in the solution), and if it is true, we replace it by the path between its endpoints whose only internal vertices are u and w. Finally, suppose the procedure reached the last case. Then, if the first newly added edge is used, replace it by the path between its endpoints, x and y, whose only internal vertex is v, and if the second newly added edge is used, replace it by the path between its endpoints whose only internal vertex is z.

We are now left with the task of finding k vertex-disjoint cycles in H'. We initialize a set $\mathcal{C}$ of vertex-disjoint cycles to be empty. As long as $|\mathcal{C}| < k$, we find a shortest cycle in H' in time $\mathcal{O}(|V(H')| \cdot |E(H')|) = k^{\mathcal{O}(1)}$ (see [22]), insert it into $\mathcal{C}$ and remove all of the edges incident to its vertices from H'. Thus, to conclude the proof, it remains to show that for each $i \in \{0, 1, \ldots, k-1\}$, after we remove the edges incident to the ith cycle from H', it still contains a cycle.

By using induction on i, we show that after removing the edges incident to the ith cycle from H', the number of edges in H' is at least $p(i) = \frac{3}{2}(c-1)k \log_2 k - 12 \cdot i \cdot \log_2(ck \log_2 k)$. This would imply that the average degree of a vertex of H' is at least $\frac{2p(i)}{|V(H')|} \geq \frac{2p(i)}{(c-1)k \log_2 k + 2} \geq 2$ (we later also explicitly show that $2p(i) \geq (\sqrt{2} + 1)ck \log_2 k$), and therefore it contains a cycle (since the average degree of a forest is smaller than 2). Initially, H' is a cubic graph, and therefore $|E(H')| = \frac{3}{2}|V(H')| \geq \frac{3}{2}(c-1)k \log_2 k$, and the claim is true. Now, suppose that it is true for some $i \in \{0, 1, \ldots, k-2\}$, and let us prove that it is true for $i+1$. By Proposition 2, a shortest cycle in H' is of length at most $2 \log_{d-1} |V(H')| + 2 \leq 3 \log_{d-1}(ck \log_2 k)$, where $d = \frac{2p(i)}{(c-1)k \log_2 k + 2} \geq \frac{2p(i)}{ck \log_2 k}$. Such a cycle is incident to at most $6 \log_{d-1}(ck \log_2 k)$ edges. Therefore, after removing from H' the edges incident to a shortest cycle in it, it contains at least $p(i) - 6 \log_{d-1}(ck \log_2 k) \geq p(i) - 6 \frac{\log_2(ck \log_2 k)}{\log_2(d-1)} = p(i) - 6 \frac{\log_2(ck \log_2 k)}{\log_2(\frac{2p(i)}{ck \log_2 k} - 1)}$ edges. Thus, by the induction hypo-

thesis, it remains to prove that $\log_2(\frac{2p(i)}{ck\log_2 k} - 1) \geq 1/2$, to which end we need to show

that $\frac{2p(i)}{ck\log_2 k} - 1 \geq \sqrt{2}$, that is, $2p(i) \geq (\sqrt{2} + 1)ck\log_2 k$. For this purpose, it is sufficient to show that $4p(i) \geq 5ck\log_2 k$. By the induction hypothesis and since $i \leq k - 1$, $4p(i) \geq 6(c-1)k\log_2 k - 48k\log_2(ck\log_2 k) = 5ck\log_2 k + (ck\log_2 k - 6k\log_2 k - 48k\log_2 k - 48k\log_2 c - 48k\log_2\log_2 k) \geq 5ck\log_2 k + (ck\log_2 k - 150(\log_2 c)k\log_2 k)$. Thus, we need to show that $c \geq 150(\log_2 c)$, which holds by our choice of c. This concludes the proof. ◄

C   Proof of Lemma 3

We can clearly detect self-loops and edges of multiplicity 2 in time $\mathcal{O}(|V| + |E|)$, and return a cycle of length 1 or 2 accordingly, and therefore we next assume that G is a simple graph. Since F is a feedback vertex set, to prove the lemma it is sufficient to present a procedure that given a vertex $v \in F$, finds in time $\mathcal{O}(|V| + |E|)$ a cycle that is at least as short as the shortest cycle in G that contains v. Indeed, then we can iterate over F and invoke this procedure, returning the shortest cycle among those returned by the procedure. Thus, we next fix some vertex $v \in F$. Let H be the connected component of G containing v.

From the vertex v, we run a breadth first search (BFS). Thus, we obtain a BFS tree T rooted at v, and each vertex in V gets a level i, indicating the distance between this vertex and v (the level of v is 0). By iterating over the neighborhood of each vertex, we identify the smallest index i_1 such that there exists an edge with both endpoints, u_1 and v_1, at level i_1 (if such an index exists), and the smallest index i_2 such that there exists a vertex w_2 at level i_2 adjacent to two vertices, u_2 and v_2, at level $i_2 - 1$ (if such an index exists). For i_1, the edge $\{u_1, v_1\}$ and the paths between v_1 and u_1 and their lowest common ancestor result in a cycle of length at most $2i_1 + 1$. For i_2, the edges $\{w_2, u_2\}$ and $\{w_2, v_2\}$ and the paths between u_2 and v_2 and their lowest common ancestor result in a cycle of length at most $2i_2$. We return the shorter cycle among the two (if such a cycle exists).

Suppose that there exists a cycle containing v, and let C be a shortest such cycle. We need to show that above procedure returns a cycle at least as short as C. Every edge of H either connects two vertices of the same level, or a vertex of level $i - 1$ with a vertex of level i. Thus, if there does not exist an index i_1' such that there exists an edge in $E(C)$ with both endpoints, u_1' and v_1', at level i_1', there must exist an index i_2' such that there exists a vertex w_2' at level i_2' adjacent to two vertices, u_2' and v_2', at level $i_2' - 1$, and the edges $\{w_2', u_2'\}$ and $\{w_2', v_2'\}$ belong to $E(C)$. First, suppose that the first case is true. Then, the procedure returns a cycle of length at most $2i_1' + 1$. The length of C cannot be shorter than $2i_1' + 1$, since it consists of a path from v to u_1' (whose length is at least i_1' since u_1' belongs to level i_1'), a path from v to v_1' whose only common vertex with the previous path is v (whose length is at least i_1' since v_1' belongs to level i_1'), and the edge $\{u_1', v_1'\}$. Now, suppose that the second case is true. Then, the procedure returns a cycle of length at most $2i_2'$. The length of C cannot be shorter than $2i_2'$, since it consists of two internally vertex-disjoint paths from v to w_2' (each of length at least i_2' since w_2' belongs to level i_2'). ◄

D   Proof of Lemma 8

For $(u, v) \in X \times X$, let $L(u, v)$ be the set of vertices of degree at most one in $G - X$ such that each $x \in L(u, v)$ is adjacent to both u and v (if $u = v$, then $L(u, u)$ is the set of vertices which have degree at most one in $G - X$ and an edge of multiplicity two to u). For each pair $(u, v) \in X \times X$, we arbitrarily mark $2|X| + 1$ vertices from $L(u, v)$ if $|L(u, v)| > 2|X| + 1$,

and we mark all vertices in $L(u,v)$ if $|L(u,v)| \leq 2|X| + 1$. We can execute this process as follows. First, in time $\mathcal{O}(|X| \cdot |V|)$, for each vertex in X we compute the set of its neighbors of degree at most one in $G - X$. Then, in time $\mathcal{O}(|X|^3)$, for each pair $(u,v) \in X \times X$ we mark at most $2|X| + 1$ vertices as required.

Since we mark at most $2|X| + 1$ vertices for each pair $(u,v) \in X \times X$, there can be at most $|X|^2(2|X| + 1)$ marked vertices in $G - X$. Let w be an unmarked vertex of degree at most one in $G - X$. We only consider the case where $\deg_{G-X}(w) = 1$, as the other case can be proved analogously. Let e be the unique edge in $G - X$ which is incident to w and let z be the other endpoint of this edge. Let $\mathcal{C}$ be a set of maximum size of vertex-disjoint cycles in G. Observe that if $\mathcal{C}$ does not contain a pair of cycles such that each of them intersects a different endpoint of e, then contracting e keeps the resulting cycles vertex disjoint in G/e. Therefore, we may assume that $\mathcal{C}$ contains two cycles C_w and C_z where C_w contains w and C_z contains z. The neighbor(s) of w in C_w must lie in X. Let these neighbors be x and y (again, x and y are not necessarily distinct). Since $w \in L(x,y)$ and it is unmarked, there are $2|X| + 1$ other vertices in $L(x,y)$ which were marked by the marking procedure. Moreover, each degree-1 vertex in $G - X$ that belongs to a cycle in $\mathcal{C}$ is either the predecessor or the successor of a vertex in X in such a cycle. Therefore, at most $2|X|$ of the marked vertices can participate in cycles in $\mathcal{C}$. Hence, there exists a vertex in $L(x,y)$, call it w', which is unused by $\mathcal{C}$. Consequently, we can route the cycle C_w through w' instead of w, which gives us a set of $|\mathcal{C}|$ vertex disjoint cycles in G/e.

The first phase of the claimed $\mathcal{O}(|X|^2 \cdot k \log k \cdot |V|)$-time algorithm performs the above marking procedure, and then proceeds as follows. First it deletes every unmarked isolated vertex in $G - X$. Then, it contracts every edge in $G - X$ incident to at least one unmarked vertex of degree one in $G - X$. After these operations, new vertices in $G - X$ of degree at most one in $G - X$ might have been created. These vertices were either the unique neighbors in $G - X$ of deleted vertices or vertices incident to contracted edges. Thus, in case new vertices in $G - X$ of degree at most one in $G - X$ have been created, the algorithm performs another phase. Here, the algorithm iterates over the set of new vertices in $G - X$ of degree at most one in $G - X$, and for each such vertex, if it is a neighbor of two vertices in X for which we have not yet marked $2|X| + 1$ vertices, the algorithm marks it. Then, the algorithm deletes vertices and contracts edges as it did in the first phase. The running time of such a phase is bounded by $\mathcal{O}(|X|^2 \cdot \rho)$, where ρ is the total number of vertices deleted and edges contracted in the previous phase. As long as new degree-one vertices are created, the execution of the algorithm continues. Since each vertex can be deleted only once, and each edge can be contracted only once, the overall running time is bounded by $\mathcal{O}(|X|(|X|^2 + |V|) + |X|^2 \cdot (|V| + |E|)) = \mathcal{O}(|X|^2 \cdot k \log k \cdot |V|)$ (since $|E| = \mathcal{O}(k \log k \cdot |V|)$). It also holds that when the algorithm terminates, $G - X$ contains at most $|X|^2(2|X| + 1)$ vertices whose degree in $G - X$ is at most one. This completes the proof of the lemma. ◄

E   Proof of Lemma 11

First, we recall the principle of inclusion-exclusion.

▶ **Proposition 5** (Folklore, [33]). *Let U and R be sets, and for every $v \in R$ let P_v be a subset of U. Use $\bar{P}_v$ to denote $U \setminus P_v$. With the convention $\bigcap_{v \in \emptyset} \bar{P}_v = U$, the following holds:*

$$|\bigcap_{v \in R} P_v| = \sum_{F \subseteq R} (-1)^{|F|} |\bigcap_{v \in F} \bar{P}_v|.$$

We now proceed with the proof of Lemma 11. In the context of Proposition 5, define the universe U as the set of all tuples $(C_1, \ldots, C_k, w_1^1, \ldots, w_k^1, w_1^2, \ldots, w_k^2, L)$ such that each C_i is a closed walk in G of length at least three, w_i^1 and w_i^2 are consecutive occurrences of vertices in C_i, $L \subseteq V$ and $(\sum_{i=1}^k |V(C_i)|) + |L| = |V|$. Here, by $|V(C_i)|$ we refer to a multiset – that is, if C_i contains x occurrences of some vertex v, then $V(C_i)$ contains x occurrences of v as well. We define the requirement space $R = V$, and for each $v \in V$, we let P_v be the set of all tuples $(C_1, \ldots, C_k, w_1^1, \ldots, w_k^1, w_1^2, \ldots, w_k^2, L) \in U$ such that $v \in (\bigcup_{i=1}^k V(C_i)) \cup L$. On the one hand, if G contains k vertex-disjoint cycles $C_1, \ldots, C_k$, then for any choice of edges $\{w_1^1, w_1^2\} \in E(C_1), \ldots, \{w_k^1, w_k^2\} \in E(C_k)$, we have that $(C_1, \ldots, C_k, w_1^1, \ldots, w_k^1, w_1^2, \ldots, w_k^2, V \setminus (\bigcup_{i=1}^k V(C_i))) \in \bigcap_{v \in V} P_v$. On the other hand, if there exists $(C_1, \ldots, C_k, w_1^1, \ldots, w_k^1, w_1^2, \ldots, w_k^2, L) \in \bigcap_{v \in V} P_v$, then since $(\sum_{i=1}^k |V(C_i)|) + |L| = |V|$, each vertex $v \in V$ occurs exactly once in either exactly one of the closed walks C_i or in the set L. In this case, we conclude that $C_1, \ldots, C_k$ are vertex-disjoint cycles. Therefore, we need to accept the input instance if and only if $|\bigcap_{v \in V} P_v| > 0$.

By Proposition 5, to decide whether $|\bigcap_{v \in V} P_v| > 0$ in time $\mathcal{O}(2^{|V|} \cdot \text{poly}(|V|))$ and polynomial space, it is sufficient to show that for each subset $F \subseteq V$, $|\bigcap_{v \in F} \bar{P}_v|$ can be computed in polynomial time. To this end, we fix a subset $F \subseteq V$. Note that $\bigcap_{v \in F} \bar{P}_v$ is the set of all tuples $(C_1, \ldots, C_k, w_1^1, \ldots, w_k^1, w_1^2, \ldots, w_k^2, L) \in U$ such that $(\bigcup_{i=1}^k V(C_i)) \cup L \subseteq V \setminus F$. Now, given an integer $\ell \in \{2k, \ldots, |V|\}$,[5] let Q_ℓ denote the set of all tuples $(C_1, \ldots, C_k, w_1^1, \ldots, w_k^1, w_1^2, \ldots, w_k^2)$ such that each C_i is a closed walk in $G - F$ of length at least three, w_i^1 and w_i^2 are consecutive occurrences of vertices in C_i, and $(\sum_{i=1}^k |V(C_i)|) = \ell$. Then, $|\bigcap_{v \in F} \bar{P}_v| = \sum_{\ell=2k}^{|V|}(|Q_\ell| \cdot \binom{|V \setminus F|}{|V|-\ell})$, where if $|V| - \ell < |V \setminus F|$, we let $\binom{|V \setminus F|}{|V|-\ell} = 0$. Thus, it remains to show that each $|Q_\ell|$ can be computed in polynomial time. To this end, fix an integer $\ell \in \{2k, \ldots, |V|\}$.

Next, we will compute $|Q_\ell|$ by simply employing the method of dynamic programming. We use a matrix M that has an entry $[i, j, v, u]$ for all $i \in \{1, \ldots, k\}$, $j \in \{1, \ldots, \ell\}$ and $v, u \in V \setminus F$. Given $i \in \{1, \ldots, k\}$, $j \in \{1, \ldots, \ell\}$ and $v, u \in V \setminus F$, let $S(i, j, v, u)$ be the set of all tuples $(C_1, \ldots, C_i, w_1^1, \ldots, w_i^1, w_1^2, \ldots, w_i^2)$ such that for all $t \in \{1, \ldots, i-1\}$, C_t is a closed walk of length at least three and w_t^1 and w_t^2 are consecutive occurrences of vertices in this walk, C_i is a walk from v to u, $w_i^1 = v$ and $\sum_{t=1}^i |V(C_t)| = j$. The entry M$[i, j, v, u]$ will be used to store $|S(i, j, v, u)|$. Observe that

$$|Q_\ell| = \sum_{v \in V \setminus F} \sum_{u \in N(v) \setminus F} \sum_{w \in N(u) \setminus F} |S(k, \ell-1, v, w)|.$$

Thus, it remains to show that the entries of M can be calculated in polynomial time.

In the basis, we have the following calculations, relating to the case where $j = 1$:

- If $j = 1$ and ($i \geq 2$ or $v \neq u$): M$[i, j, v, u] = 0$.
- Else if $j = 1$: M$[i, j, v, u] = 1$.

Now, consider only entries where $j \geq 2$, which have not already been calculated in the basis. Then, we have the following calculations:

- If $i \geq 2, j \geq 3$ and $v = u$:
$$\text{M}[i, j, v, u] = \sum_{w \in N(u) \setminus F} \text{M}[i, j-1, v, w] + \sum_{p \in V \setminus F} \sum_{q \in N(p) \setminus F} \sum_{w \in N(q) \setminus F} \text{M}[i-1, j-2, p, w].$$
- Else: M$[i, j, v, u] = \sum_{w \in N(u) \setminus F} \text{M}[i, j-1, v, w]$.

---

[5] We do not consider the case where $\ell < 2k$ since k closed walks must overall contain at least $2k$ vertices.

It is straightforward to verify that the calculations are correct. The order of the computation is an ascending order with respect to j, which ensures that when an entry is calculated, the entries on which it relies have already been calculated. To output a solution, we apply a simple self-reduction from the decision to the search variant of the problem. In particular, we repeatedly remove edges until no more edges can be removed from the graph while preserving a yes-instance. ◄