KERNELIZATION OF CYCLE PACKING WITH RELAXED DISJOINTNESS CONSTRAINTS*

AKANKSHA AGRAWAL†, DANIEL LOKSHTANOV‡, DIPTAPRIYO MAJUMDAR§, AMER E. MOUAWAD¶, AND SAKET SAURABH‖

Abstract. A key result in the field of kernelization, a subfield of parameterized complexity, states that the classic DISJOINT CYCLE PACKING problem, i.e. finding k vertex disjoint cycles in a given graph G, admits no polynomial kernel unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. However, very little is known about this problem beyond the aforementioned kernelization lower bound (within the parameterized complexity framework). In the hope of clarifying the picture and better understanding the types of "constraints" that separate "kernelizable" from "non-kernelizable" variants of DISJOINT CYCLE PACKING, we investigate two relaxations of the problem. The first variant, which we call ALMOST DISJOINT CYCLE PACKING, introduces a "global" relaxation parameter t. That is, given a graph G and integers k and t, the goal is to find at least k distinct cycles such that every vertex of G appears in at most t of the cycles. The second variant, PAIRWISE DISJOINT CYCLE PACKING, introduces a "local" relaxation parameter and we seek at least k distinct cycles such that every two cycles intersect in at most t vertices. While the PAIRWISE DISJOINT CYCLE PACKING problem admits a polynomial kernel for all $t \geq 1$, the kernelization complexity of ALMOST DISJOINT CYCLE PACKING reveals an interesting spectrum of upper and lower bounds. In particular, for $t = \frac{k}{c}$, where c could be a function of k, we obtain a kernel of size $\mathcal{O}(2^{c^2} k^{7+c} \log^3 k)$ whenever $c \in o(\sqrt{k})$. Thus the kernel size varies from being sub-exponential when $c \in o(\sqrt{k})$, to quasi-polynomial when $c \in o(\log^\ell k)$, $\ell \in \mathbb{R}_+$, and polynomial when $c \in \mathcal{O}(1)$. We complement these results for ALMOST DISJOINT CYCLE PACKING by showing that the problem does not admit a polynomial kernel whenever $t \in \mathcal{O}(k^\epsilon)$, for any $0 \leq \epsilon < 1$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

Key words. parameterized complexity, cycle packing, kernelization, lower bounds, relaxation

AMS subject classifications. 68Q25, 68Q15, 68Q17, 68R10

1. Introduction.

Polynomial-time preprocessing is one of the widely used methods to tackle NP-hard problems in practice, as it plays well with exact algorithms, heuristics, and approximation algorithms. Until recently, there was no robust mathematical framework to analyze the performance of preprocessing routines. Progress in parameterized complexity [12] made such an analysis possible. In parameterized complexity, each problem instance is coupled with an integer k, which is called as the parameter, and the parameterized problem is said to admit a *kernel* if there is a polynomial-time algorithm, called a *kernelization* algorithm, that reduces the input instance down to an instance whose size is bounded by a function $f(k)$ in k, while preserving the answer. Such an algorithm is called an $f(k)$-*kernel* for the problem. If $f(k)$ is a polynomial, quasi-polynomial, subexponential, or exponential function of k, we say that this is a polynomial, quasi-polynomial, subexponential, or exponential kernel, respectively. Over the last decade or so, kernelization has become a very active field of study, especially with the development of complexity-theoretic tools to show

*An extended abstract of this paper [2] has appeared in the proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).

Funding: The research leading to these results received funding from the BeHard grant under the recruitment programme of the of Bergen Research Foundation (D. Lokshtanov) and the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreements no. 306992 (S. Saurabh).

†Department of Informatics, University of Bergen, Norway (akanksha.agrawal@uib.no).

‡Department of Informatics, University of Bergen, Norway (daniello@ii.uib.no).

§Institute of Mathematical Sciences, Chennai, India (diptapriyom@imsc.res.in).

¶Department of Informatics, University of Bergen, Norway (a.mouawad@uib.no).

‖Institute of Mathematical Sciences, Chennai, India (saket@imsc.res.in).

that a problem does not admit a polynomial kernel [4, 13, 17, 20], or a kernel of a specific size [9, 10, 21]. We refer the reader to the survey articles by Kratsch [22] and Lokshtanov et al. [23] for recent developments.

One of the first and important problems to which the lower-bounds machinery was applied is the NP-complete Disjoint Cycle Packing problem. In the Disjoint Cycle Packing problem, we are given as input an n-vertex graph G and an integer k, and the task is to find a collection $\mathcal{C}$ of at least k pairwise disjoint vertex sets of G, such that every set $C \in \mathcal{C}$ is a cycle in G. The Disjoint Cycle Packing problem can be solved in $\mathcal{O}(k^{k \log k} n^{\mathcal{O}(1)})$ using dynamic programming over graphs of bounded treewidth [3, 5]. Bodlaender et al. [6] showed that, when parameterized by k, Disjoint Cycle Packing does not admit a polynomial kernel unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ (and the polynomial hierarchy collapses to its third level, which is considered very unlikely). Beyond the aforementioned negative result for polynomial kernels and the folklore $\mathcal{O}(k^{k \log k} n^{\mathcal{O}(1)})$-time algorithm, the Disjoint Cycle Packing problem has remained mostly unexplored from the viewpoint of parameterized complexity.

Our problems and results. In this paper we study two variants of Disjoint Cycle Packing, obtained by relaxing the disjointness constraint. In particular, we focus on the kernelization complexity of the Disjoint Cycle Packing problem by considering two relaxed versions of the problem, one with a "local" relaxation parameter and the other with a "global" relaxation parameter. In the locally relaxed variant, which we call Pairwise Disjoint Cycle Packing, the goal is to find at least k distinct cycles in a graph G such that they pairwise intersect in at most t vertices.

Pairwise Disjoint Cycle Packing **Parameter:** k
Input: An undirected (multi) graph G and integers k and t.
Question: Does G have at least k distinct cycles $C_1, \ldots, C_k$ such that $\lvert V(C_i) \cap V(C_j)\rvert \le t$ for all $i \neq j$?

We consider two cycles to be distinct whenever their edge sets differ by at least one element. Note that when $t = 0$, Pairwise Disjoint Cycle Packing corresponds to the original Disjoint Cycle Packing problem. However, when $t = |V(G)|$ the Pairwise Disjoint Cycle Packing problem is solvable in time polynomial in $|V(G)|$ and k since we can enumerate distinct cycles in a graph with polynomial delay [26]. In other words, any k distinct cycles in a graph will trivially pairwise intersect in at most $|V(G)|$ vertices. We show that Pairwise Disjoint Cycle Packing remains NP-complete when $t = 1$. Then, we complement this result by showing that the problem admits a polynomial kernel for $t = 1$ and a polynomial compression for $t \geq 2$. An interesting problem which remains unclear is to determine what value of t separates NP-hard instances from polynomial-time solvable ones.

The second relaxation we consider is Almost Disjoint Cycle Packing. The goal in Almost Disjoint Cycle Packing is to determine whether G contains at least k distinct cycles such that every vertex in $V(G)$ appears in at most t of them. As we shall see, the kernelization complexity landscape for Almost Disjoint Cycle Packing is much more diverse than that of Pairwise Disjoint Cycle Packing. In some sense, this suggests that the global relaxation parameter does a "better job" of capturing the "hardness" of the original problem.

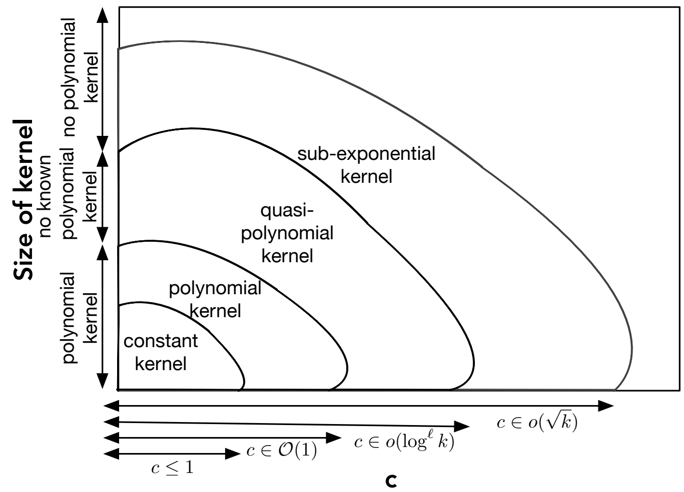

FIG. 1. *Spectrum of kernelization algorithms for* ALMOST DISJOINT CYCLE PACKING *as* c *grows in the denominator of* $t = \frac{k}{c}$.

ALMOST DISJOINT CYCLE PACKING **Parameter:** k
Input: An undirected (multi) graph G and integers k and t.
Question: Does G have at least k distinct cycles $C_1, \ldots, C_k$ such that every vertex in $V(G)$ appears in at most t of them?

Again, for $t = 1$, ALMOST DISJOINT CYCLE PACKING corresponds to DISJOINT CYCLE PACKING and when $t = k$ the problem is solvable in time polynomial in $|V(G)|$ and k by simply enumerating distinct cycles. However, and rather surprisingly, we show that t has to be "very close" to k for this relaxation to become "easier" than the original problem, at least in terms of kernelization. In fact, we show that as long as $t = \mathcal{O}(k^{1-\epsilon})$, where $0 < \epsilon \leq 1$, ALMOST DISJOINT CYCLE PACKING remains NP-complete and admits no polynomial kernel unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. We complement our hardness result by a spectrum of kernel upper bounds. To that end, we consider the case $t = \frac{k}{c}$, where c is a constant or a function of k. We show that we can (in polynomial time) compress an instance of ALMOST DISJOINT CYCLE PACKING into an equivalent instance with $\mathcal{O}(2^{c^2} k^{7+c} \log^3 k)$ vertices. This implies polynomial, quasi-polynomial, or subexponential size kernels for ALMOST DISJOINT CYCLE PACKING, depending on whether c is a constant, $c \in o(\log k)$, or $c \in o(\sqrt{k})$, respectively. It remains open whether the problem is in P or NP-hard for $t = \frac{k}{c}$, when c is a constant. A high level summary of our results for ALMOST DISJOINT CYCLE PACKING is given in Figure 1.

Related Results. Our results also fit into the relatively new direction of research that is concerned with the parameterized complexity of problems with relaxed packing/covering constraints. For several important problems (that we need to solve), there are settings in which we need not be very strict about constraints. This is particularly interesting for "strict" problems where, e.g., (a) it is known that no polynomial kernels are possible unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$, or where (b) the algorithm with the best running time matches the known lower bound, or where (c) no considerable improvements have been made either algorithmically or in terms of kernel upper/lower

	ALMOST DISJOINT CYCLE PACKING		PAIRWISE DISJOINT CYCLE PACKING	
	NP-complete	Poly. kernel	NP-complete	Poly. kernel
$t = 0$			Yes	No
$t = 1$	Yes	No	Yes	Yes
$t = \mathcal{O}(1)$	Yes	No	Open	Yes
$t = \mathcal{O}(k^{\epsilon})$	Yes	No	Open	Yes
$t = \frac{k}{c}$	Open	Yes	Open	Yes

TABLE 1
Summary of our results and some open problems.

bounds. The DISJOINT CYCLE PACKING problem, which is the main subject of this work, falls into categories (a) and (c). Before we delve into the technical details of our results, let us look at some examples where the introduction of relaxation parameters has been successful. Abasi et al. [1], followed by Gabizon et al. [18], studied a generalization of the k-PATH problem, namely r-SIMPLE k-PATH, where the task is to find a walk of length k that never visits any vertex more than r times. Here r is the relaxation parameter. By definition, the generalized problem is computationally harder than the original. However, observe that for $r = 1$ the problem is exactly the problem of finding a simple path of length k in G. On the other hand, for $r = k$ the problem is easily solvable in polynomial time, as any walk in G of length k will suffice. In some sense, the "further away" an instance of the generalized problem is from being an instance of the original, the easier the instance is. Put differently, gradually increasing r from 1 to k should make the problem computationally easier. This intuition was confirmed by the authors by providing, amongst other results, algorithms for the generalized problem whose worst-case running time matches the running time of the best algorithm for the original problem up to constants in the exponent, and improves significantly as the relaxation parameter increases. Also closely related is the work of Romero et. al. [28, 29] and Fernau et al. [15] who studied relaxations of graph packing problems allowing certain overlaps.

2. Preliminaries. We let $\mathbb{N}$ denote the set of natural numbers, $\mathbb{R}$ denote the set of real numbers, $\mathbb{R}_+$ denote the set of non-zero positive real numbers, and $\mathbb{R}_{\geq 1}$ denote the set of real numbers greater than or equal to one. For $r \in \mathbb{N}$, by $[r]$ we denote the set $\{1, 2, \ldots, r\}$.

Graphs. We use standard terminology from the book of Diestel [11] for those graph-related terms which are not explicitly defined here. We only consider finite graphs possibly having loops and multi-edges. For a graph G, $V(G)$ and $E(G)$ denote the vertex and edge sets of the graph G, respectively. For a vertex $v \in V(G)$, we use $d_G(v)$ 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 loop at a vertex v contributes two to its degree. For a vertex subset $S \subseteq V(G)$, $G[S]$ and $G - S$ are the graphs induced on S and $V(G) \setminus S$, respectively. For a vertex subset $S \subseteq V(G)$, we let $N_G(S)$ and $N_G[S]$ denote the open and closed neighborhood of S in G. That is, $N_G(S) = \{v \mid (u, v) \in E(G), u \in S\} \setminus S$ and $N_G[S] = N_G(S) \cup S$. For a graph G and an edge $e \in E(G)$, G/e denotes the graph obtained by contracting e in G (loops and multi-edges are preserved).

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 $v_0, v_1, \dots, v_\ell$ such that $(v_i, v_{(i+1) \mod \ell+1})$ is an edge for all $0 \leq i \leq \ell$. We note that both a double edge and a loop are cycles. If P is a path from a vertex u to a vertex v in 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 or a cycle Q, we use $V(Q)$ to denote the set of vertices in Q and the length of Q is denoted by $|Q|$ (i.e, $|Q| = |V(Q)|$). 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 sometimes refer to a path or a cycle Q as a $|Q|$-path or $|Q|$-cycle. Given a vertex $v \in V(G)$, a *v-flower* of order k is a set of k cycles in G whose pairwise intersection is exactly $\{v\}$. We say a set of distinct vertices $P = \{v_1, \dots, v_\ell\}$ in G forms a *degree-two path* if P is a path and all vertices $\{v_1, \dots, v_\ell\}$ have degree exactly two in G. We say P is a *maximal degree-two path* if no proper superset of P also forms a degree-two path. Finally, a *feedback vertex set* is a subset S of vertices such that $G - S$ is a forest.

Theorem 2.1 (see [14]). *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 $ck \log k$. Moreover, there is a polynomial-time algorithm that takes a graph G and an integer k as input, and outputs either k vertex disjoint cycles or a feedback vertex set of size at most $ck \log k$.*

Parameterized Complexity. We only state the basic definitions and general results needed for our purposes. For more details on parameterized complexity in general, and kernelization in particular, we refer the reader to the books of Downey and Fellows [12], Flum and Grohe [16], Niedermeier [25], and the more recent book by Cygan et al. [8].

Definition 1. *A reduction rule that replaces an instance (I, k) of a parameterized language L by a new instance (I', k') is said to be* sound *or* safe *if $(I, k) \in L$ if and only if $(I', k') \in L$.*

Definition 2. *A* polynomial compression *of a parameterized language $L \subseteq \Sigma^* \times \mathbb{N}$ into a language $R \subseteq \Sigma^*$ is an algorithm that takes as input an instance $(I, k) \in \Sigma^* \times \mathbb{N}$, works in time polynomial in $|I| + k$, and returns a string I' such that:*

- *$|I'| \leq p(k)$ for some polynomial $p(.)$, and*
- *$I' \in R$ if and only if $(I, k) \in L$.*

In case $|\Sigma| = 2$, the polynomial $p(.)$ is called the bitsize *of the compression.*

Note that polynomial compressions are a generalization of kernels and being able to rule out a compression algorithm automatically rules out a kernelization algorithm. Like in classical complexity, in the world of kernel lower bounds, it is often easier to "transfer" hardness from one problem to another. To be able to do so, we need an appropriate notion of reduction.

Definition 3. *Let $L, R \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. An algorithm $\mathcal{A}$ is called a* polynomial parameter transformation *(PPT, for short) from L to R if, given an instance (I, k) of problem L, $\mathcal{A}$ works in polynomial time and outputs an equivalent instance (I', k') of problem R, i.e., $(I, k) \in L$ if and only if $(I', k') \in R$, such that $k' \leq p(k)$ for some polynomial $p(.)$.*

Theorem 2.2 (see [8]). *Let $L, R \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems and assume that there exists a polynomial parameter transformation from L to R. Then, if L does not admit a polynomial compression (into any language), neither does R.*

3. Almost Disjoint Cycle Packing. As previously noted, Bodlaender et al. [6] showed that Disjoint Cycle Packing admits no polynomial kernel unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. On the other hand, finding k distinct cycles in a graph is solvable in time polynomial in $n = |V(G)|$ and k [26]. The intuition is that the more cycles we allow a vertex to belong to, the easier the problem of finding k distinct cycles should become. In this section, we study the spectrum of kernelization algorithms for Almost Disjoint Cycle Packing based on the "distance" between k and t. Recall that given an instance (G, k, t) of Almost Disjoint Cycle Packing, our goal is to find at least k distinct cycles such that each vertex appears in at most t of them. To formalize the notion of distance between k and t, we define the following class of problems.

Let $L = \{(G, k, t) \mid G$ has k cycles such that every vertex appears in at most t of them$\}$. Basically, L is the language Almost Disjoint Cycle Packing. For a non-decreasing and polynomial-time computable function $f : \mathbb{N} \to \mathbb{R}_+$ (polynomial in k), we define the following sub-language of L.

$$L_f = \{(G, k, t) \mid (G, k, t) \in L \text{ and } t = \lceil k/f(k) \rceil\}.$$

When f is the identity function, i.e. when $f(k) = k$, L_f is exactly the Disjoint Cycle Packing problem, which is known not to admit a polynomial kernel [6]. In Section 3.1, we show that even when $f(k) = k^\epsilon$, for any fixed $0 < \epsilon \leq 1$, L_f (or equivalently Almost Disjoint Cycle Packing with $t = k^{1-\epsilon}$) is NP-complete and does not admit a polynomial kernel unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. If $f = a$ (a constant function), where $a \leq 1$ and $a \in \mathbb{R}_+$, then L_f can be decided in polynomial time (as finding any k distinct cycles is enough). This implies that for $f = a$ we have a constant kernel. In Section 3.2, we obtain a polynomial kernel for $f = c$ (another constant function), where $c > 1$ and $c \in \mathbb{R}$. In fact, our result implies that for $f \in \mathcal{O}(1)$, $f \in o(\log^\ell k)$ $(\ell \in \mathbb{N})$, or $f \in o(\sqrt{k})$, we can (in polynomial time) compress an instance of Almost Disjoint Cycle Packing into an equivalent instance of polynomial, quasi-polynomial, or subexponential size, respectively (see Figure 1).

Before we consider the kernelization complexity of the Almost Disjoint Cycle Packing problem, we first show, using standard arguments, that the problem is fixed-parameter tractable when parameterized by k, i.e., the problem can be solved in $f(k)n^{\mathcal{O}(1)}$ time, where $n = |V(G)|$ and f is a computable function. Armed with Theorem 2.1, we can assume that, for an instance (G, k, t) of Almost Disjoint Cycle Packing, the treewidth of G is at most $\mathcal{O}(k \log k)$; as G has a feedback vertex set of size at most $\mathcal{O}(k \log k)$. Courcelle's Theorem [7] gives a powerful way of quickly showing that a problem is fixed-parameter tractable on bounded treewidth graphs. That is, it suffices to show that our problem can be expressed in monadic second-order logic (MSO_2). We only briefly review the syntax and semantics of MSO_2. The reader is referred to the excellent survey by Martin Grohe [19] for more details. Sentences in MSO_2 contain quantifiers, logical connectives ($\neg$, $\vee$, and $\wedge$), vertex variables, vertex set variables, edge set variables, binary relations $\in$ and $=$, and the atomic formula $E(u, v)$ expressing that u and v are adjacent. If a graph property can be described in this language, then this description can be made algorithmic:

Theorem 3.1 (see [7]). *If a graph property can be described as a formula ϕ in the monadic second-order logic of graphs, then it can be recognized in time $f(||\phi||, tw(G))(|E(G)| + |V(G)|)$ if a given graph G has this property, where f is a computable function, $||\phi||$ is the length of the encoding of ϕ as a string, and $tw(G)$ is the treewidth of G.*

Lemma 3.1. ALMOST DISJOINT CYCLE PACKING *can be solved in* $f(k)n^{\mathcal{O}(1)}$ *time, for some computable function* f*. In other words, the problem is fixed-parameter tractable when parameterized by* k*.*

Proof. Given an instance (G, k, t) of ALMOST DISJOINT CYCLE PACKING, we construct a formula ϕ such that $||\phi||$ is bounded by an exponential function in k and t. Given that $t \leq k$ and that the treewidth of G is at most $\mathcal{O}(k \log k)$, applying Theorem 3.1 completes the proof.

We set

$$\phi = \exists_{C_1} \dots \exists_{C_k} \Big(\forall_{v \in V(G)} \text{ cap}(v, C_1, \dots, C_k) \bigwedge_{1 \leq i \leq k} \text{cycle}(C_i) \bigwedge_{1 \leq i \neq j \leq k} \text{distinct}(C_i, C_j) \Big)$$

where $C_i \subseteq E(G)$, cycle(C_i) is true if and only if C_i is a cycle, distinct(C_i, C_j) is true if and only if C_i and C_j are distinct (as edge sets), and cap$(v, C_1, \dots, C_k)$ is true if and only if v appears in at most t cycles. Formally, we set

$$\text{cycle}(C_i) = \text{connected}(C_i) \wedge \text{not-empty}(C_i) \wedge (\forall_v \text{ degree-two}(v, C_i) \vee v \notin C_i)$$

$$\text{distinct}(C_i, C_j) = (\exists_{e \in C_i} \forall_{e' \in C_j} e \neq e') \vee (\exists_{e \in C_j} \forall_{e' \in C_i} e \neq e')$$

$$\text{cap}(v, C_1, \dots, C_k) = \bigwedge_{S = \{i_1, \dots, i_t\} \subseteq \binom{[k]}{t}} \text{appears-in}(v, S) \rightarrow \text{misses}(v, [k] \setminus S).$$

In order to guarantee that C_i is a cycle we make sure that it induces a non-empty (not-empty(C_i)) connected graph (connected(C_i)) and that every vertex v is either incident to exactly two edges of C_i (degree-two(v, C_i)) or not in C_i. The formula distinct(C_i, C_j) is true if and only if the symmetric difference of C_i and C_j contains at least one edge. For a set $S = \{i_1, \dots, i_t\} \subseteq \binom{[k]}{t}$, appears-in$(v, S)$ is true if and only if vertex v appears in all cycles $C_{i_1}, \dots, C_{i_t}$. The formula misses$(v, [k] \setminus S)$ is true if and only v does not belong to any of the cycles in $\{C_1, \dots, C_k\} \setminus \{C_{i_1}, \dots, C_{i_t}\}$. It is not hard to see that $G \models \phi$ if and only if (G, k, t) is a yes-instance. Furthermore, note that $||\phi||$ depends only on k and $t \leq k$. ∎

3.1. Refuting polynomial kernels for $t = \mathcal{O}(k^{1-\epsilon})$. We now show that ALMOST DISJOINT CYCLE PACKING restricted to L_f, where $f(k) = k^{\epsilon}$, does not admit a polynomial kernel, for any $0 < \epsilon \leq 1$, unless NP ⊆ coNP/poly. Here k is the number of required cycles and $t = \frac{k}{f(k)} = k^{1-\epsilon}$ is the maximum number of cycles a vertex can belong to. Below we define the DISJOINT FACTORS problem [6] which is known to admit no polynomial compression unless NP ⊆ coNP/poly.

Let Σ_q be an alphabet set of q elements. By Σ_q^* we denote the set of all strings over Σ_q. A *factor* of a string $\bar{y} = y_1 y_2 \dots y_n \in \Sigma_q^*$ is a pair (s, e), where $s, e \in [n]$ and $s < e$, such that $y_s y_{s+1} \dots y_e$ is a substring of $\bar{y}$ and $y_s = y_e$. Two factors (s, e) and (s', e') of $\bar{y}$ are said to be disjoint if $\{s, s+1, \dots, e\} \cap \{s', s'+1, \dots, e'\} = \emptyset$. The string $\bar{y}$ is said to have disjoint factors over Σ_q if for all $x \in \Sigma_q$ there is a factor (s_x, e_x) such that $y_{s_x} = y_{e_x} = x$, and for all distinct $x, \hat{x} \in \Sigma_q$, (s_x, e_x) and $(s_{\hat{x}}, e_{\hat{x}})$ are disjoint factors.

DISJOINT FACTORS	**Parameter:** q
Input: Alphabet set Σ_q, string $\bar{y} \in \Sigma_q^*$.	
Question: Does $\bar{y}$ have disjoint factors over Σ_q?	

Construction. We give a polynomial parameter transformation from an instance $(\Sigma_q, \bar{y})$ of Disjoint Factors to an instance (G, k, t) of Almost Disjoint Cycle Packing. For technical reasons, we will assume that $t - 1 = 2^l$, for some $l \in \mathbb{N}$. Note that this can be achieved by at most doubling the value of t while keeping t in $\mathcal{O}(k^{1-\epsilon})$. We let $l = \log_2(t-1)$. The end goal will be to construct a graph in which we have to find k cycles such that every vertex appears in at most $t = \mathcal{O}(k^{1-\epsilon})$ of them.

The reduction is as follows. Let $\Sigma_q = \{x_1, x_2, \dots x_q\}$. We create a vertex $\hat{x}_i \in V(G)$ corresponding to each element x_i, where $i \in [q]$. For $\bar{y} = y_1 y_2 \dots y_n \in \Sigma_q^*$ we create a path $P_y = (u, \hat{y}_1, \hat{y}_2, \dots \hat{y}_n, u')$ by adding two new vertices u and u'. We add an edge between $\hat{x}_i$ and $\hat{y}_j$, for $i \in [q]$ and $j \in [n]$, if and only if $x_i = y_j$. We also add four more vertices u_1, u_2, u_1', and u_2' to $V(G)$ and add edges (u_1, u_2), (u_2, u), (u, u_1), (u_1', u_2'), (u_2', u'), and (u', u_1') to $E(G)$ (see Figure 2). For each $x_i \in \Sigma_q$, we attach $t-1$ triangles to $\hat{x}_i$, i.e. we add edges $\{(z_i^1, \tilde{z}_i^1), (z_i^2, \tilde{z}_i^2), \dots, (z_i^{t-1}, \tilde{z}_i^{t-1})\}$ and $(z_i^j, \hat{x}_i), (\hat{x}_i, \tilde{z}_i^j)$, for $j \in [t-1]$. Next, we create a path $P_w = (w_1, w_1', w_2, w_2', \dots, w_l, w_l')$ in G. We add a set $R = \{r_i \mid i \in [l]\}$ of l independent vertices and for $i \in [l]$, we add the edges (w_i, r_i) and (w_i', r_i) to $E(G)$. Finally, we add edges (u, w_1) and (w_l', u') (see Figure 2). We set $k = tq + t + l + 1$. This completes the construction. In what follows, we let (G, k, t) denote an instance of Almost Disjoint Cycle Packing given by the above construction for an instance $(\Sigma_q, \bar{y})$ of Disjoint Factors. The next proposition follows by construction.

Proposition 1. *Let $P = (s, a_1, a_1', a_2, a_2', \dots, a_n, a_n', s')$ be a path and $B = \{b_i \mid i \in [n]\}$ be a set of independent vertices. Let H be the graph consisting of path P, the set B, and, for $i \in [n]$, the edges (a_i, b_i) and (a_i', b_i). Then, for each $B' \subseteq B$, there exists a unique path $P_{B'}$ between s and s' such that $V(P_{B'}) \cap B = B'$. Moreover, the set $\mathcal{B} = \{P_{B'} \mid B' \subseteq B\}$ is the set of all possible paths between s and s' in H.*

Applying Proposition 1 to G, for each $R' \subseteq R$, we have a (unique) cycle $C_{R'}$ which contains all the vertices in $V(P_y)$, all the vertices in P_w, and exactly the vertices of the set R' from R. We define a family of cycles $\mathcal{R} = \{C_{R'} \mid R' \subseteq R\} \cup \{(w_i, w_i', r_i) \mid i \in [l]\}$. Note that $|\mathcal{R}| = 2^l + l = t + l - 1$ and each $C \in \mathcal{R}$ is a cycle in G. The intuition of having the set of cycles $\{C_{R'} \mid R' \subseteq R\}$ in G is that each vertex in the path P_y appears in $t-1$ of these cycles, and can therefore participate in at most one additional cycle (which contains vertices in $V(P_y)$). Our end goal is to associate this extra cycle with a factor. We let $\mathcal{U} = \{(u, u_1, u_2), (u', u_1', u_2')\}$ and $\mathcal{Z} = \{(z_i^j, \tilde{z}_i^j, \hat{x}_i) \mid i \in [q], j \in [t-1]\}$. Note that each $C \in \mathcal{U} \cup \mathcal{Z}$ forms a cycle in G.

Lemma 3.2. *If $(G, k = tq + t + l + 1, t)$ is a yes-instance of* Almost Disjoint Cycle Packing *then there is a solution containing all cycles in $\mathcal{Z} \cup \mathcal{U}$.*

Proof. Let $\mathcal{S}$ be the set of $\hat{k} \geq k$ cycles in G such that every vertex belongs to at most t cycles in $\mathcal{S}$. We create another solution $\mathcal{S}'$ with k' cycles such that $k' \geq \hat{k}$ and $\mathcal{Z} \cup \mathcal{U} \subseteq \mathcal{S}'$. Initially, we have $\mathcal{S}' = \mathcal{S}$. Suppose for some $i \in [q]$ and $j \in [t-1]$, cycle $(z_i^j, \tilde{z}_i^j, \hat{x}_i) \notin \mathcal{S}$. If $\hat{x}_i$ belongs to less than t cycles in $\mathcal{S}$, then we can add $(z_i^j, \tilde{z}_i^j, \hat{x}_i)$ to $\mathcal{S}'$ and obtain a larger solution. Otherwise, let $\mathcal{C}_i$ be the set of cycles in $\mathcal{S} \setminus (\mathcal{Z} \cup \mathcal{U})$ in which $\hat{x}_i$ is present. Pick any cycle $C \in \mathcal{C}_i$ and replace it by $(z_i^j, \tilde{z}_i^j, \hat{x}_i)$ in $\mathcal{S}'$. Observe that $\hat{x}_i$ separates z_i^j and $\tilde{z}_i^j$ from the rest of the graph. Therefore, there is a unique (simple) cycle in G containing z_i^j and $\tilde{z}_i^j$. Also, we can do the above replacement at most $t-1$ times. This implies that, even after the replacement, every vertex appears in at most t cycles in $\mathcal{S}'$. A similar argument can be given for cycles in $\mathcal{U}$. Therefore, we can obtain a solution $\mathcal{S}'$ consisting of k' cycles, where $k' \geq \hat{k}$, $\mathcal{Z} \cup \mathcal{U} \subseteq \mathcal{S}'$, and

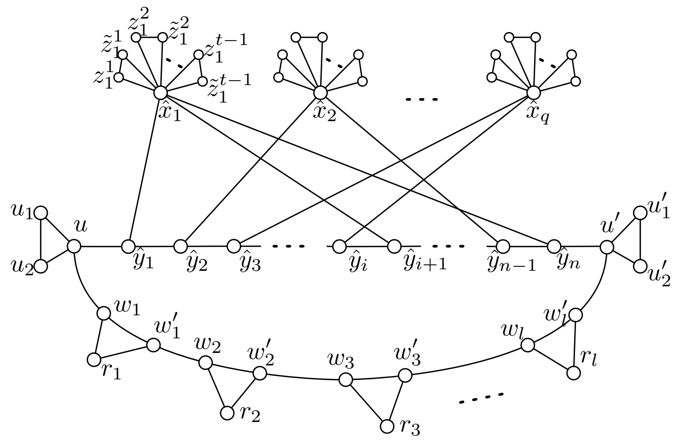

FIG. 2. *An instance* $(G, k = tq + t + l + 1, t)$ *of* ALMOST DISJOINT CYCLE PACKING *from an instance* $(\Sigma_q, \bar{y})$ *of* DISJOINT FACTORS.

every vertex appears in at most t of the cycles. ∎

Lemma 3.3. *If* $(G, k = tq + t + l + 1, t)$ *is a yes-instance of* ALMOST DISJOINT CYCLE PACKING *and* $\mathcal{S}$ *is a set of* k *cycles such that every vertex appears in at most* t *of the cycles then* $\mathcal{S}$ *contains all the cycles in* $\mathcal{R}$*.*

Proof. Let $\mathcal{S}$ be a set of k cycles in G such that every vertex $v \in V(G)$ belongs to at most t cycles in $\mathcal{S}$. Observe that, for $i \in [q]$, $\hat{x}_i$ can appear in at most t cycles in $\mathcal{S}$. Therefore, the number of cycles $C \in \mathcal{S}$ such that $V(C) \cap \{\hat{x}_i \mid i \in [q]\} \neq \emptyset$ is at most tq.

Since u is a cut vertex separating u_1 and u_2 from the rest of the graph, the only cycle containing both u_1 and u_2 is (u, u_1, u_2). Similarly, the only cycle containing both u'_1 and u'_2 is (u', u'_1, u'_2). Therefore, the remaining cycles in $\mathcal{S}$ (not considered so far) are cycles in $G' = G[V']$ as well, where $V' = R \cup V(P_w) \cup V(P_y)$.

By construction P_w and P_y are induced paths in G' (and in G). Moreover, vertices in $V(P_y)$ are degree-two vertices in G'. Therefore, a cycle in G' either contains all the vertices from P_y or none of the vertices in P_y. By Proposition 1, the number of distinct paths (excluding P_y) between u and u' (i.e. the start and end vertices of P_y) is $2^l = t - 1$. Observe that each of these paths forms a cycle C in G' along with the path P_y and $C \in \mathcal{R}$. This implies that the number of cycles containing vertices from $V(P_y)$ is $t - 1$. The cycles in G' which do not contain vertices from path P_y are the cycles in $G'[P_w \cup R]$. Given that P_w is an induced path in $G'[P_w \cup R]$, the only cycles that $G'[P_w \cup R]$ contains are the vertex disjoint cycles formed by w_i, w'_i, r_i, for $i \in [l]$. Also, for each $i \in [l]$, $(w_i, w'_i, r_i) \in \mathcal{R}$. Note that the vertices in $V(P_w) \cup R$ belong to exactly t cycles in $\mathcal{R}$. Consequently, if $\mathcal{S}$ does not contain all cycles in $\mathcal{R}$ then $|\mathcal{S}| < tq + 2 + t - 1 + l = tq + t + l + 1$; a contradiction. ∎

Lemma 3.4. *If* $(G, k = tq + t + l + 1, t)$ *is a yes-instance of* ALMOST DISJOINT CYCLE PACKING *then there is a set* $\mathcal{S}$ *of* k *cycles such that every vertex appears in at most* t *of the cycles in* $\mathcal{S}$ *and, for all* $C \in \mathcal{S}$*,* $V(C) \cap \{\hat{x}_i \mid i \in [q]\} \leq 1$*.*

Proof. Let $\mathcal{S}$ be a set of k cycles in G such that every vertex appears in at most t of the cycles in $\mathcal{S}$. By Lemmas 3.2 and 3.3, we can assume that $\mathcal{Z} \cup \mathcal{U} \cup \mathcal{R} \subseteq \mathcal{S}$.

Suppose that there is a cycle $C \in \mathcal{S}$ such that C contains at least two vertices from $\{\hat{x}_i \mid i \in [q]\}$. Let $\hat{x}_i$ and $\hat{x}_j$ be two such vertices. By Lemma 3.2, we know that, for each $p \in [q]$, $\hat{x_p}$ can belong to at most one more cycle in $\mathcal{S} \setminus \mathcal{Z}$. Since $C \in \mathcal{S}$, the number of cycles in $\mathcal{S}$ can be at most $tq + t + l$, contradicting the fact that $\mathcal{S}$ is a solution of size $tq + t + l + 1$. ◻

Lemma 3.5. *Let $(\Sigma_q, \bar{y})$ be an instance of* Disjoint Factors *and $(G, k = tq + t + l + 1, t)$ be the corresponding instance of* Almost Disjoint Cycle Packing. *Then, $(\Sigma_q, \bar{y})$ is a yes-instance of* Disjoint Factors *if and only if (G, k, t) is a yes-instance of* Almost Disjoint Cycle Packing.

Proof. In the forward direction let (s_i, e_i) be a factor for x_i, $i \in [q]$. We construct a solution $\mathcal{S}$ for (G, k, t) as follows. We include all the cycles in $\mathcal{Z} \cup \mathcal{U} \cup \mathcal{R}$ to $\mathcal{S}$. For $i \in [q]$, we add the cycle $C_i = (\hat{x}_i, \hat{y}_{s_i}, \hat{y}_{s_i+1}, \ldots, \hat{y}_{e_i})$ to $\mathcal{S}$. Note that $s_i, e_i \in [n]$, $s_i < e_i$, and, for distinct $i, j \in [q]$, the sets $\{s_i, s_{i+1}, \ldots, e_i\}$ and $\{s_j, s_{j+1}, \ldots, e_j\}$ are disjoint sets. Therefore, for C_i and C_j, $i \neq j$ and $i, j \in [q]$, we have $V(C_i) \cap V(C_j) = \emptyset$. Observe that, for $i \in [q]$, $\hat{x}_i$ appears in $t-1$ cycles in $\mathcal{Z} \cup \mathcal{U} \cup \mathcal{R}$ and in the cycle C_i. Therefore, $\hat{x}_i$ belongs to at most t cycles in $\mathcal{S}$. Also, vertices in path P_y belong to $t-1$ cycles in $\mathcal{Z} \cup \mathcal{U} \cup \mathcal{R}$ and at most one of the cycles in $\{C_i \mid i \in [y]\}$. Therefore, every vertex appears in at most t of the cycles in $\mathcal{S}$ and $|\mathcal{S}| = |\mathcal{Z} \cup \mathcal{U} \cup \mathcal{R}| + |\Sigma_q| = |\mathcal{Z}| + |\mathcal{U}| + |\mathcal{R}| + |\Sigma_q| = (t-1)q + 2 + t - 1 + l + q = tq + t + l + 1 = k$, as needed.

In the reverse direction, consider a set of k cycles $\mathcal{S}$ in G such that every vertex appears in at most t of the cycles. By Lemmas 3.2 and 3.3, we can assume that $\mathcal{C} = \mathcal{Z} \cup \mathcal{U} \cup \mathcal{R} \subseteq \mathcal{S}$. Furthermore, $C \in \mathcal{S} \setminus \mathcal{C}$ cannot contain any vertex from $V(P_w) \cup \{u, u'\}$, since these vertices already belong to t cycles in $\mathcal{U} \cup \mathcal{R}$. Also, C cannot contain any vertices from $\{z_i^j, \tilde{z}_i^j \mid i \in [q], t \in [t-1]\}$, as there is a unique cycle containing them which is present in $\mathcal{Z}$. Therefore, C contains vertices only from $\{\hat{x}_i \mid i \in [q]\} \cup V(P_y)$. Moreover, vertices in $V(P_y)$ belong to $t-1$ cycles in $\mathcal{R}$. Therefore, each vertex in $V(P_y)$ can belong to at most one cycle $C \in \mathcal{S} \setminus \mathcal{C}$. By Lemma 3.4, we know that, for each $C \in \mathcal{S} \setminus \mathcal{C}$, C contains at most one vertex from $\{\hat{x}_i, i \in [q]\}$. Also, all the cycles in $\mathcal{S} \setminus \mathcal{C}$ must contain a vertex from $\{\hat{x}_i, i \in [q]\}$. Therefore, cycle C contains a vertex from $\{\hat{x}_i, i \in [q]\}$ and some vertices from $V(P_y)$. Observe that C must contain consecutive vertices from P_y. For a cycle C which contains $\hat{x}_i$, for some $i \in [q]$, and vertices $\hat{y}_{s_i}, \hat{y}_{s_{i+1}}, \ldots, \hat{y}_{e_i}$, we return a factor (s_i, e_i), where $s_i < e_i$. Note that for $i, j \in q$ and $i \neq j$, $\{s_i, s_{i+1}, \ldots, e_i\} \cap \{s_j, s_{j+1}, \ldots, e_j\} = \emptyset$. Therefore, we have a factor for each x_i, $i \in q$. This concludes the proof. ◻

We can now state the main theorem of this section.

Theorem 3.2. *Let $f : \mathbb{N} \to \mathbb{R}_{\geq 1}$ be a non-decreasing computable function such that $f(k) \in \mathcal{O}(k^\epsilon)$, where $0 < \epsilon \leq 1$. Then,* Almost Disjoint Cycle Packing *admits no polynomial kernel restricted to L_f unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.*

Proof. We refute polynomial kernels for Almost Disjoint Cycle Packing restricted to L_f. Since $f(k) \in \mathcal{O}(k^\epsilon)$, we have that $t = \mathcal{O}(k^{1-\epsilon}) = \mathcal{O}(k^{\epsilon'})$. We start with an instance $(\Sigma_q, \bar{y})$ of Disjoint Factors and create an instance (G, k, t) of Almost Disjoint Cycle Packing by applying the reduction as described. Note that the parameter for Disjoint Factors is q. Moreover, $k = \mathcal{O}(q^{\frac{1}{\epsilon}})$ whenever $t = q^{\frac{\epsilon'}{1-\epsilon'}}$, $k = tq + t + l + 1$, and $l = \log_2(t-1)$. Replacing q by $t^{\frac{1-\epsilon'}{\epsilon'}}$ for k, we get $t^{\frac{1}{\epsilon'}} < k < 2t^{\frac{1}{\epsilon'}}$ and hence $t = \mathcal{O}(k^{\epsilon'})$. By Lemma 3.5, this polynomial time reduction is a polynomial parameter transformation from Disjoint Factors to Almost Disjoint Cycle Packing. Therefore, assuming we have a polynomial kernel for Almost Disjoint Cycle Packing, where $t = \mathcal{O}(k^{\epsilon'})$ and $0 < \epsilon' < 1$, implies a polynomial compression for Disjoint Factors, contradicting Theorem 2.2. So, Almost Disjoint Cycle Packing restricted to L_f has no polynomial kernel unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. ∎

3.2. A kernel for Almost Disjoint Cycle Packing. Let $f : \mathbb{N} \to \mathbb{R}_{\geq 1}$ be a non-decreasing computable function such that $f(k) \in o(\sqrt{k})$. In this section, we consider the Almost Disjoint Cycle Packing problem restricted to L_f. The kernelization algorithm presented here is inspired from the *lossy kernel* for the Cycle Packing problem (Section 5, [24]). To simplify notation, we let $c = f(k)$ and use c instead of $f(k)$ throughout the section, which implies that $t = \lceil \frac{k}{c} \rceil$. As we shall see, the assumption $c \in o(\sqrt{k})$ is required to guarantee that our kernelization algorithm does in fact run in time polynomial in the input size. We show that, as long as $c \in o(\sqrt{k})$, we can in polynomial time reduce an instance to at most $\mathcal{O}(2^{\lceil c \rceil^2} k^{7+\lceil c \rceil} \log^3 k)$ vertices. Our kernelization algorithm can be more or less divided into three stages. We start by computing (using Theorem 2.1) a feedback vertex set of size at most $\mathcal{O}(k \log k)$ and denote this set by F (assuming no k vertex disjoint cycles were found). We let $T = G - F$ and 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 greater than two in T, respectively. Moreover, we let $\mathcal{P}$ denote the set of all maximal degree-two paths in $G[T]$. Next, we bound the size of $T_{\leq 1}$. We know that T is a forest. By a property of forests, we know that $|T_{\geq 3}| \leq |T_{\leq 1}|$ and $|\mathcal{P}| \leq |T_{\geq 3}| + |T_{\leq 1}|$. So, an upper bound on $|T_{\leq 1}|$ provides an upper bound on $|T_{\geq 3}|$ and $|\mathcal{P}|$. In the second stage, we show that (roughly speaking) the graph can have at most $\lceil c \rceil - 1$ vertices of high degree. Using this fact, the last stage consists of bounding the size of T_2. Note that bounding the sizes of $T_{\leq 1}$, T_2, $T_{\geq 3}$, and $\mathcal{P}$ implies a bound on the size of T. Combining this bound with the fact that F is of size at most $\mathcal{O}(k \log k)$, we get the claimed kernel.

Bounding the size of $T_{\leq 1}$. First, we get rid of vertices of degree one and two in the graph G using Reduction Rules A1 and A2. Observe that we can safely delete vertices of degree zero or one (in G) as they do not participate in any cycle.

Reduction Rule A1. *Delete vertices of degree zero or one in G.*

Reduction Rule A2. *If there is a vertex v of degree exactly two in G then delete v and connect its two neighbors by a new edge.*

Lemma 3.6. *Reduction Rule A2 is safe.*

Proof. Let u be a vertex of degree two in G and let $N_G(u) = \{v, w\}$. Let G' be the graph obtained after contracting edge (u, v) onto vertex v.

Consider a set $\mathcal{C} = \{C_1, \ldots, C_k\}$ of cycles such that every vertex in $V(G)$ participates in at most t of them. There can be at most t cycles in $\mathcal{C}$ to which u belongs. Moreover, both v and w (and hence the edge (u, v)) must be present in all those cycles. Now, after contracting the edge (u, v) onto v, we can see that v is present in exactly those cycles where u was also present. Therefore, if (G, k, t) is a yes-instance then so is (G', k, t).

Let (G', k, t) be a yes-instance such that $\mathcal{C}' = \{C'_1, \ldots, C'_k\}$ is a solution for (G', k, t). Consider those cycles in $\mathcal{C}'$ containing the edge (v, w). There can be at most t such cycles. Now, when we translate back to the graph G, the edge (v, w) corresponds to a path of length three. Therefore, v, u, and w, all participate in at most t cycles, as needed. ∎

Reduction Rule A3. *If there exists an edge $(u, v) \in E(G)$ of multiplicity more than $2t$ then reduce its multiplicity to $2t \leq 2k$.*

The safeness of Reduction Rule A3 follows from the fact that any pair of vertices can belong to at most t cycles. The fact that we can assume $2t \leq 2k$ follows from the observation that when $t = k$ the problem becomes solvable in time polynomial in n and k. Once Reduction Rules A1, A2, and A3 are no longer applicable, the minimum degree of the graph G is three and the multiplicity of every edge is at most $2t$. Note that every vertex in $T_{\leq 1}$ is either a leaf or an isolated vertex in T. Therefore, every vertex of $T_{\leq 1}$ has at least two neighbours in F. For $(u, v) \in F \times F$, let $L(u, v)$ be the set of vertices of degree at most one in $T = G - F$ 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 $T = G - F$ and at least two edges to u). For each pair $(u, v) \in F \times F$, we mark $|F|\lceil \frac{k}{c} \rceil + 2k + 1$ vertices from $L(u, v)$ if $L(u, v) > |F|\lceil \frac{k}{c} \rceil + 2k + 1$ and mark all vertices in $L(u, v)$ if $L(u, v) \leq |F|\lceil \frac{k}{c} \rceil + 2k + 1$.

Reduction Rule A4. *If $|T_{\leq 1}| \geq |F|^2(|F|\lceil \frac{k}{c} \rceil + 2k + 1) + 1$ then there exists an unmarked vertex $v \in T_{\leq 1}$.*

- *If $d_{G-F}(v) = 0$ then delete v.*
- *If $d_{G-F}(v) = 1$ contract the unique edge in $G - F$ which is incident to v. We let e denote this unique edge and we let w denote the other endpoint onto which we contract e.*

Reduction Rule A4 is also available as Lemma 5.7 in [24].

Lemma 3.7. *Reduction Rule A4 is safe.*

Proof. Since we marked at most $|F|\lceil \frac{k}{c} \rceil + 2k + 1$ vertices for each pair $(u, v) \in F \times F$, there can be at most $|F|^2(|F|\lceil \frac{k}{c} \rceil + 2k + 1)$ marked vertices in $T_{\leq 1}$. Let v be an unmarked vertex. We only consider the case where $d_{G-F}(v) = 1$, as the other case can be proved analogously.

Let $\mathcal{C}$ be a maximum packing in G such that every vertex in $V(G)$ appears in at most $t = \lceil \frac{k}{c} \rceil$ cycles of $\mathcal{C}$. Observe that if $\mathcal{C}$ does not contain any cycles intersecting $\{v\}$ then contracting e will keep all the cycles in $\mathcal{C}$ present in $G' = G/e$. Consider those cycles in $\mathcal{C}$ containing vertex v. Such cycles either contain both v and (its unique neighbor in T) w or contain v and two of its neighbors in F. Note that cycles containing both v and w are also present in G' as w is connected to all neighbors of v. Hence, we only need to show that cycles containing v and two of its neighbors in F can be reconstructed in G'. Fix such a cycle C and let x and y be the neighbors of v in F (x and y are not necessarily distinct). Since $v \in L(x, y)$ and it is unmarked, there are $|F|\lceil \frac{k}{c} \rceil + 2k + 1$ vertices in $L(x, y)$ which are already marked by the marking procedure. Furthermore, since G can have at most $|F|\lceil \frac{k}{c} \rceil$ cycles such that every vertex appears in at most $\lceil \frac{k}{c} \rceil$ of them, at least one of these marked vertices, call it v', is not present in any of the cycles in $\mathcal{C}$; this is true since, for any cycle $C \in \mathcal{C}$, $|V(C) \cap F| \geq V(C) \cap T_{\leq 1}$, which implies that at most $|F|\lceil \frac{k}{c} \rceil$ marked vertices can belong to cycles in $\mathcal{C}$. Therefore we can route the cycle C through v' instead of v. Since v can appear in at most $\lceil \frac{k}{c} \rceil$ cycles and we have marked $|F|\lceil \frac{k}{c} \rceil + 2k + 1 > |F|\lceil \frac{k}{c} \rceil + 2\lceil \frac{k}{c} \rceil + 1$ vertices for each pair in F, we can repeat the same procedure for each cycle in $\mathcal{C}$ containing v to obtain a packing $\mathcal{C}'$ in G' whose size is at least $|\mathcal{C}|$.

For the reverse direction, let $\mathcal{C}'$ be a maximum packing in G' such that every vertex in $V(G')$ appears in at most $t = \lceil \frac{k}{c} \rceil$ cycles of $\mathcal{C}'$. The only cycles in G' which do not correspond to cycles in G are those cycles containing an edge (w, z), where $z \in N_{G'}(w)$ but $z \notin N_G(w)$. However, we can simply replace such edges by a path on three vertices in G, namely w, v, and z. It is not hard to see that v appears in at most as many cycles as w. Hence, we can construct, from $\mathcal{C}'$, a packing $\mathcal{C}$ in G whose size is at least $|\mathcal{C}'|$. This completes the proof. ∎

Bounding the number of high-degree vertices. When none of the aforementioned reduction rules are applicable, the size of $T_{\leq 1}$, $T_{\geq 3}$, and $\mathcal{P}$, is at most $|F|^2(|F|\lceil \frac{k}{c} \rceil + 2k+1) = \mathcal{O}(k^4 \log^3 k)$. Consider $\mathcal{P}$, i.e. the collection of maximal degree-two paths in T_2, and assume that there exists a set $F_{\lceil c \rceil} = \{x_1, \ldots, x_{\lceil c \rceil}\} \subseteq F$ (of size $\lceil c \rceil$) such that for every vertex $x \in F_{\lceil c \rceil}$ there exists a path $P \in \mathcal{P}$ such that x has at least $4k\lceil c \rceil$ neighbours in P. Our goal is to show that if $F_{\lceil c \rceil}$ exists then we have a yes-instance. Before we do so, we need to prove the following lemma.

Lemma 3.8. *If $\lceil c \rceil \in o(\sqrt{k})$ and $\lceil c \rceil > \lceil \frac{k}{c} \rceil$ then* Almost Disjoint Cycle Packing *can be solved in time polynomial in n.*

Proof. When $\lceil c \rceil > \lceil \frac{k}{c} \rceil$, $k < \lceil c \rceil^2$. Moreover, observe that if $\lceil c \rceil \in o(\sqrt{k})$ then k is a constant. Therefore, we can simply apply the algorithm of Lemma 3.1 which runs in time polynomial in n when k is a constant. ∎

Reduction Rule A5. *If there exists a set of $\lceil c \rceil$ vertices $F_{\lceil c \rceil} = \{x_1, \ldots, x_{\lceil c \rceil}\} \subseteq F$ such that for all x_i, $1 \leq i \leq \lceil c \rceil$, $|N_G(x_i) \cap V(\mathcal{P})| > |F|^2(|F|\lceil \frac{k}{c} \rceil + 2k+1)4k\lceil c \rceil$, then return a trivial yes-instance.*

Lemma 3.9. *Reduction Rule A5 is safe.*

Proof. For each x_i, we mark a path $P_i \in \mathcal{P}$ satisfying the condition $|N_G(x_i) \cap V(P_i)| \geq 4k\lceil c \rceil$. Since $|\mathcal{P}| \leq |F|^2(|F|\lceil \frac{k}{c} \rceil + 2k+1)$ and $|N_G(x_i) \cap V(\mathcal{P})| > |F|^2(|F|\lceil \frac{k}{c} \rceil + 2k+1)4k\lceil c \rceil$ such a path must exist. Next, we construct a set of cycles $\mathcal{C}_i$, for each x_i, as follows. Given x_i and P_i, we pick (any) $2\lceil \frac{k}{c} \rceil$ neighbors of x_i to form $\lceil \frac{k}{c} \rceil$ cycles pairwise intersecting only in x_i. Note that every vertex in $V(P_i)$ appears at most once in $\mathcal{C}_i$. We claim that $\mathcal{C} = \mathcal{C}_1 \cup \ldots \cup \mathcal{C}_c$ is in fact the desired solution. Clearly, $|\mathcal{C}| = \lceil c \rceil \lceil \frac{k}{c} \rceil \geq k$. Every vertex in $F_{\lceil c \rceil}$ appears in exactly $\lceil \frac{k}{c} \rceil$ cycles and every other vertex appears in at most $\lceil c \rceil \leq \lceil \frac{k}{c} \rceil$ cycles (assuming $\lceil c \rceil \in o(\sqrt{k})$ and applying Lemma 3.8 otherwise), as needed. ∎

After applying Reduction Rule A5, there can be at most $\lceil c \rceil - 1$ vertices in F having more than $|F|^2(|F|\lceil \frac{k}{c} \rceil + 2k+1)4k\lceil c \rceil = \mathcal{O}(k^5 \log^3 k)$ neighbors in T_2. We let $F_{\lceil c \rceil - 1} \subseteq F$ denote the maximum sized such subset and we let $F^\star = F \setminus F_{\lceil c \rceil - 1}$. For any vertex $x \in F^\star$, $|N_G(x) \cap V(\mathcal{P})| \leq |F|^2(|F|\lceil \frac{k}{c} \rceil + 2k+1)4k\lceil c \rceil$ and, consequently, $|N_G(F^\star) \cap V(\mathcal{P})| \leq |F|^2(|F|\lceil \frac{k}{c} \rceil + 2k+1)4k\lceil c \rceil |F^\star| \leq |F|^3(|F|\lceil \frac{k}{c} \rceil + 2k+1)4k\lceil c \rceil = \mathcal{O}(k^6 \log^3 k)$.

Bounding the size of T_2. We start by marking all vertices in F, $T_{\leq 1}$, $T_{\geq 3}$, and $N_G(F^\star) \cap V(\mathcal{P})$. The total number of marked vertices is therefore in $\mathcal{O}(k^6 \log^3 k)$. Moreover, all the unmarked vertices must be in T_2 and form degree-two paths. As minimum degree of G is at least three, each unmarked vertex must have at least one neighbor in $F_{\lceil c \rceil - 1}$ and cannot have neighbors in $F^\star$. We call a set of unmarked vertices a *region* if they form a maximal path in $G[T_2]$. At this point, the total number of regions is in $\mathcal{O}(k^6 \log^3 k)$, as the number of marked vertices is in $\mathcal{O}(k^6 \log^3 k)$. Therefore, our last step is to bound the size of each region. To do so, we first recursively further subdivide each region as follows. Fix a region R and check for each vertex $x_i \in F_{\lceil c \rceil - 1}$, the value of $|N_G(x_i) \cap R|$. If $|N_G(x_i) \cap R| < 4k\lceil c \rceil 2^{\lceil c \rceil}$, then we again mark the vertices in $N_G(x_i) \cap R$, increasing the number of regions by a multiplicative factor of at most $4k\lceil c \rceil 2^{\lceil c \rceil}$. We repeat this process as long as there exists a region R and a vertex $x_i \in F_{\lceil c \rceil - 1}$ satisfying $|N_G(x_i) \cap R| < 4k\lceil c \rceil 2^{\lceil c \rceil}$. Since $|F_{\lceil c \rceil - 1}| < \lceil c \rceil$, repeating this procedure for every region and every vertex in $F_{\lceil c \rceil - 1}$ increases the number of regions to at most $\mathcal{O}(2^{\lceil c \rceil^2} k^{6+\lceil c \rceil} \log^3 k)$; each of the initial $\mathcal{O}(k^6 \log^3 k)$ regions can be subdivided into at most $(4k\lceil c \rceil 2^{\lceil c \rceil})^{\lceil c \rceil}$ subregions.

Lemma 3.10. *Let H be a graph consisting of a path P and an independent set $X = \{x_1, \ldots, x_{\lceil c \rceil}\}$ of size $\lceil c \rceil \geq 1$. Let $k \geq \lceil c \rceil^2$ be an integer. If $\forall x \in X$ we have $|N_H(x)| \geq 4k\lceil c \rceil 2^{\lceil c \rceil}$ and $\forall p \in V(P)$ we have $|N_H(p) \cap X| > 0$, then we can construct a set of distinct cycles $\mathcal{C} = \mathcal{C}_1 \cup \ldots \cup \mathcal{C}_{\lceil c \rceil}$ such that (a) $|\mathcal{C}_i| = \lceil \frac{k}{c} \rceil$, (b) all cycles in $\mathcal{C}_i$ pairwise intersect in x_i, and (c) every vertex in P appears in at most one cycle in $\mathcal{C}$.*

Proof. We prove the lemma by induction on the number of vertices in X. Let $P = \{p_1, \ldots, p_{|P|}\}$. For the base case, we have $\lceil c \rceil = 1$ and $X = \{x_1\}$. Since every vertex on the path is connected to x_1 and x_1 has at least $8k$ neighbors, we know that $|V(P)| \geq 8k$. Therefore, taking the first $2k$ vertices on the path we can easily construct k cycles pairwise intersecting only at $\{x\}$.

Suppose the statement holds for all $\lceil c \rceil$, where $1 < \lceil c \rceil \leq \lceil q \rceil - 1$, and consider the case $\lceil c \rceil = \lceil q \rceil$. We claim that there exists a vertex x in X such that we can pack $\lceil \frac{k}{q} \rceil$ cycles pairwise intersecting only at $\{x\}$ using only the first $4k(\lceil q \rceil - 1) + 1$ vertices on the path, i.e. $\{p_1, \ldots, p_{4k(\lceil q \rceil - 1)+1}\}$. In fact, it is enough to show that at least one vertex $x \in X$ has at least $2k$ neighbours in $\{p_1, \ldots, p_{4k(\lceil q \rceil - 1)+1}\}$. If no such vertex exists then $|N_H(X) \cap \{p_1, \ldots, p_{4k(\lceil q \rceil - 1)}\}| < 2k\lceil q \rceil$. But since $|\{p_1, \ldots, p_{4k(\lceil q \rceil - 1)+1}\}| = 4k(\lceil q \rceil - 1) + 1 > 2k\lceil q \rceil$ (for $\lceil q \rceil \geq 2$) this contradicts the fact that every vertex in $\{p_1, \ldots, p_{4k(\lceil q \rceil - 1)+1}\}$ must have at least one neighbor in X. Now delete vertex x from X and vertices $\{p_1, \ldots, p_{4k(\lceil q \rceil - 1)+1}\}$ from P. Moreover, if after deleting x some vertices in $P' = P \setminus \{p_1, \ldots, p_{4k(\lceil q \rceil - 1)+1}\}$ no longer have neighbors in $X' = X \setminus \{x\}$ simply delete those vertices and add an edge connecting their two unique neighbors in P. Call this new graph H'. Observe that for all $x \in X'$, we have $|N_{H'}(x)| > 4k\lceil q \rceil 2^{\lceil q \rceil} - 4k(\lceil q \rceil - 1) - 1 = 4k\lceil q \rceil (2^{\lceil q \rceil} - 1) + 4k - 1 \geq 4k(\lceil q \rceil - 1) 2^{\lceil q \rceil - 1}$, when $\lceil q \rceil \geq 2$. Applying the induction hypothesis to X' and P', we know that we can pack $\lceil \frac{k}{q-1} \rceil \geq \lceil \frac{k}{q} \rceil$ cycles for each vertex $x \in X'$, as needed. ∎

Using Lemma 3.10, we can get an upper bound on the size of a region R by applying the following reduction rule. Recall that by construction (and after subdividing regions), vertices of a region have neighbours only in $F_{\lceil c \rceil - 1}$, where $F_{\lceil c \rceil - 1}$ is a set of at most $\lceil c \rceil - 1$ vertices. In fact, for each region R, there exists a set $F_R \subseteq F_{\lceil c \rceil - 1}$ such that each vertex in R has at least one neighbor in F_R and each vertex in F_R has at least $4k\lceil c \rceil 2^{\lceil c \rceil}$ neighbors in R.

Reduction Rule A6. *Let R be a region such that $|R| > 4k\lceil c \rceil 4^{\lceil c \rceil}$. Let $\mathcal{Q} = \{Q_1, Q_2, \ldots\}$ be a family of sets which partitions R such that for any two vertices $u, v \in R$, we have $u, v \in Q_i$ if and only if $N_G(u) \cap F_R = N_G(v) \cap F_R$. In other words, two vertices belong to the same set in $\mathcal{Q}$ if and only if they share the same neighborhood in F_R. Since $|R| > 4k\lceil c \rceil 4^{\lceil c \rceil}$ and $|\mathcal{Q}| \leq 2^{\lceil c \rceil}$, there exists a set $Q \in \mathcal{Q}$ such that $|Q| > 4k\lceil c \rceil 2^{\lceil c \rceil}$. Let v be a vertex in Q and let w be a neighbor of v in R (v can have at most two neighbors in R). Contract the edge (v, w) onto w. Note that since $|Q| > 4k\lceil c \rceil 2^{\lceil c \rceil}$, each vertex in F_R has at least $4k\lceil c \rceil 2^{\lceil c \rceil}$ neighbors in R even after the contraction.*

Lemma 3.11. *Reduction Rule A6 is safe.*

Proof. Let $\mathcal{C}$ be a maximum packing in G and $\mathcal{C}'$ be a maximum packing in G' such that every vertex in $V(G)$ and $V(G')$ appears in at most $t = \frac{k}{c}$ cycles of $\mathcal{C}$ and $\mathcal{C}'$, respectively.

Since $G' = G/e$ is a minor of G, we have $|\mathcal{C}| \geq |\mathcal{C}|'$. We now show that $|\mathcal{C}'| \geq |\mathcal{C}|$.

Let $\mathcal{C}_R$ denote the cycles in $\mathcal{C}$ which intersect with both R and F_R. Observe that all cycles in $\mathcal{C} \setminus \mathcal{C}_R$ are still present in G' (possibly of shorter length). Moreover, in $\mathcal{C} \setminus \mathcal{C}_R$, all the vertices of R appear in the same number of cycles, as any such cycle must cross all of the region. Consider the at most $|F_R|\lceil \frac{k}{c} \rceil$ cycles in $\mathcal{C}_R$. By applying Lemma 3.10, we can find at least as many cycles in $G'[R \cup F_R]$. Every vertex in F_R appears in at most $\lceil \frac{k}{c} \rceil$ of them and every vertex in R appears in at most one of them. Therefore no vertex is ever used more than $\lceil \frac{k}{c} \rceil$ times, as needed. ∎

Since the number of regions is in $\mathcal{O}(2^{\lceil c \rceil^2} k^{6+\lceil c \rceil} \log^3 k)$ and the size of a region is at most $4kc4^c$, the theorem follows.

Theorem 3.3. *Let $f : \mathbb{N} \to \mathbb{R}_{\geq 1}$ be a non-decreasing computable function such that $f(k) \in o(\sqrt{k})$. For $c = f(k)$,* Almost Disjoint Cycle Packing *admits a kernel consisting of at most $\mathcal{O}(2^{c^2} k^{7+c} \log^3 k)$ vertices over L_f.*

Theorem 3.3 implies that when $c \in o(\sqrt{k})$ the Almost Disjoint Cycle Packing problem admits a subexponential kernel. When $c \in o(\log^\ell k)$, $\ell \in \mathbb{N}$, the problem admits a quasi-polynomial kernel. Finally, when $c \in \mathcal{O}(1)$ the problem admits a polynomial kernel.

4. Pairwise Disjoint Cycle Packing.

Recall that in the Pairwise Disjoint Cycle Packing problem, given a graph G and integers k and t, the goal is to find at least k cycles such that every pair of cycles intersects in at most t vertices.

4.1. NP-completeness for $t = 1$.

To show NP-completeness of Pairwise Disjoint Cycle Packing, for $t = 1$, we give a reduction from a variant of SAT called 2/2/4-SAT defined as follows: Each clause contains four literals, each variable appears four times in the formula, twice negated and twice not negated, and the question is whether there is a truth assignment of the variables such that in each clause there are exactly two true literals. This variant was shown to be NP-complete by Ratner and Warrnuth [27]. We let ϕ denote the formula, $U = \{u_1, \ldots, u_{|U|}\}$ denote the set of variables, and $W = \{w_1, \ldots, w_{|W|}\}$ denote the set of clauses.

Variable gadget. For each variable $u \in U$, we construct a graph G_u, which we call a necklace graph, as follows. G_u consists of 32 vertices. The first set of 16 vertices form a cycle $C_u^{in} = \{v_1^1, \ldots, v_{16}^1\}$ and the second set of 16 vertices form cycle $C_u^{out} = \{v_1^2, \ldots, v_{16}^2\}$. We add an edge $v_i^1 v_i^2$ for $1 \leq i \leq 16$. Informally, G_u consists of 16 4-cycles where every two consecutive cycles share an edge (see Figure 3). Cycle C_u^{in} is the inner cycle, C_u^{out} is the outer cycle, and we number all 4-cycles from 1 to 16 in a clockwise order, i.e. we denote the cycles by $\{C_u^1, \ldots, C_u^{16}\}$. It is not hard to see that the maximum size of a packing of distinct cycles, pairwise intersecting in at most one vertex, is 8. Such a packing consists of picking either odd-numbered or even-numbered cycles. We adopt the convention that picking odd-numbered cycles corresponds to setting the variable to true and picking even-numbered cycles corresponds to setting the variable to false. Since each variable appears in exactly four clauses, we mark two consecutive 4-cycles for each clause as follows. Assume variable u appears in w_1, w_2, w_3, and w_4. Then cycles numbered 1 and 2 are reserved for the clause gadget of w_1, cycles numbered 5 and 6 are reserved for the clause gadget of w_2, cycles numbered 9 and 10 are reserved for the clause gadget of w_3, and finally cycles numbered 13 and 14 are reserved for the clause gadget of w_4. Note that every pair of marked cycles will be separated by at least two consecutive 4-cycles. For a cycle C_u^i, $1 \leq i \leq 16$, we let e_u^i denote the edge of C_u^i which lies on the outer cycle C_u^{out}. These outer edges will be used to connect variable gadgets to clause gadgets.

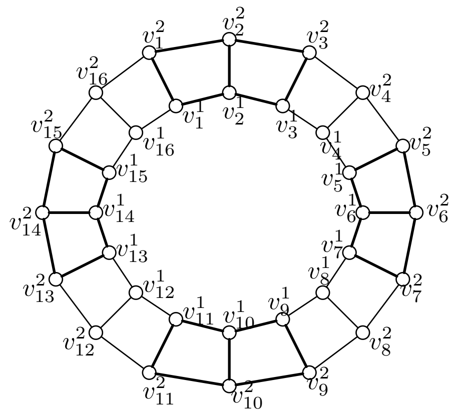

FIG. 3. *Variable gadgets*

Clause gadget. Let $w \in W$ be a clause in ϕ and let u_1, u_2, u_3, and u_4 be the variables appearing in w. We construct the clause gadget for w as follows (Figure 4). First, we add two pairs of vertices, a red pair and a blue pair, denoted by $\mathcal{P}_w = \{\{r_w^1, r_w^2\}, \{b_w^1, b_w^2\}\}$. Let G_{u_i} be the graph constructed as variable gadget for variable u_i, $i \in \{1, 2, 3, 4\}$, and assume, without loss of generality, that cycles $C_{u_i}^1$ and $C_{u_i}^2$ in G_{u_i} are marked for clause w. If u_i appears positively in w, we add an edge from r_w^1 to one endpoint of the outer edge $e_{u_i}^1$ and another edge from r_w^2 to the other endpoint of $e_{u_i}^1$. We say $\{r_w^1, r_w^2\}$ is linked to $e_{u_i}^1$. If u_i appears negatively in w, we add an edge from r_w^1 to one endpoint of the outer edge $e_{u_i}^2$ and another edge from r_w^2 to the other endpoint of $e_{u_i}^2$. We do the reverse construction for $\{b_w^1, b_w^2\}$. That is, if u_i appears positively in w we add an edge from b_w^1 to one endpoint of the outer edge $e_{u_i}^2$ and another edge from b_w^2 to the other endpoint of $e_{u_i}^2$. If u_i appears negatively in w we add an edge from b_w^1 to one endpoint of the outer edge $e_{u_i}^1$ and another edge from b_w^2 to the other endpoint of $e_{u_i}^1$. The process is repeated for every variable appearing in the clause. Since each clause consists of four variables, every vertex in a clause gadget will have exactly four neighbors in (different) variable gadgets.

The construction. Given an instance ϕ of 2/2/4-SAT, we first construct all variable gadgets followed by all clause gadgets. To complete the construction, we add $\binom{4|W|}{2} - 2|W|$ cycles of length four, which we call auxiliary cycles, as follows. Recall that for each clause $w \in W$ we create two pairs of vertices $\mathcal{P}_w = \{\{r_w^1, r_w^2\}, \{b_w^1, b_w^2\}\}$. We add internally vertex disjoint 4-cycles between r_w^i and b_w^j, $i, j \in \{1, 2\}$ (Figure 4), i.e., 4-cycles whose only common vertices are r_w^i and b_w^j. Finally, for every two clauses $w, w' \in W$ we add internally vertex disjoint 4-cycles between r_w^i and $r_{w'}^j$, b_w^i and $b_{w'}^j$, and r_w^i and $b_{w'}^j$, $i, j \in \{1, 2\}$. Since every pair of vertices in clause gadgets are connected by a cycle except for $2|W|$ pairs, namely $\{r_w^1, r_w^2\}$ and $\{b_w^1, b_w^2\}$ for each $w \in W$, the total number of added cycles follows. We let G be the resulting graph and $(G, k = 8|U| + \binom{4|W|}{2}, t = 1)$ denotes the resulting PAIRWISE DISJOINT CYCLE PACKING instance.

Lemma 4.1. *Let G be a graph constructed from a given* 2/2/4-SAT *formula as described above. Then, any packing of distinct cycles pairwise intersecting in at most one vertex has size at most $8|U| + \binom{4|W|}{2}$.*

Proof. Consider any cycle C which is not fully contained inside a variable gadget

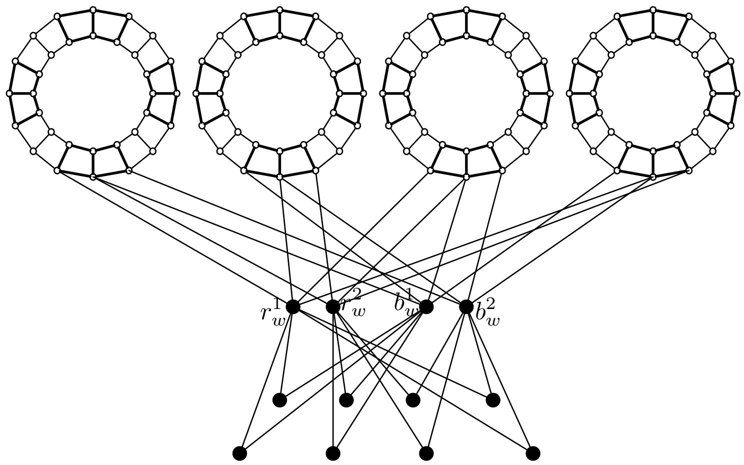

FIG. 4. *Clause gadget and its corresponding auxiliary cycles*

(i.e. a necklace graph). We claim that such a cycle must contain at least two vertices from clause gadgets (not necessarily the same clause gadget). To see why, it is enough to note that C must contain at least one such vertex, say v (recall that all vertices in auxiliary cycles are either in clause gadgets or have degree exactly two). However, v has exactly one neighbor in any variable gadget and all neighbors of v not in clause gadgets have degree exactly two (and connect two different vertices from clause gadgets).

Since any cycle not fully contained inside a variable gadget must use at least two vertices from clause gadgets and no two cycles can share more than a single vertex, we know that the total number of such cycles is at most $\binom{4|W|}{2}$. To conclude the proof, note that any variable gadget can contribute at most 8 cycles that pairwise intersect in at most one vertex (in this case the cycles are in fact vertex disjoint). ∎

Lemma 4.2. *If* ϕ *is a yes-instance of* 2/2/4-SAT *then* $(G, k = 8|U| + \binom{4|W|}{2}, t = 1)$ *is a yes-instance of* PAIRWISE DISJOINT CYCLE PACKING.

Proof. Consider a satisfying assignment of the variables such that in each clause there are exactly two true literals. If a variable is set to false we pack all even-numbered cycles in its corresponding gadget. Similarly, if a variable is set to true we pack all odd-numbered cycles. The total number of such cycles is $8|U|$ and all cycles are vertex disjoint. Next, we pack all $\binom{4|W|}{2} - 2|W|$ auxiliary cycles. These cycles pairwise intersect in at most one vertex by construction. Hence, we still need to pack exactly $2|W|$ cycles. Let $w \in W$ be a clause in ϕ, $\mathcal{P}_w = \{\{r_w^1, r_w^2\}, \{b_w^1, b_w^2\}\}$, and let u_1, u_2, u_3, and u_4 be the variables appearing in w. Note that the vertices in $\{r_w^1, r_w^2\}$ do not share an auxiliary cycle nor do the vertices in $\{b_w^1, b_w^2\}$. We show that for each clause we can pack two cycles using each of its pairs exactly once.

Let G_{u_i} be the variable gadget constructed for variable u_i, $i \in \{1, 2, 3, 4\}$, and assume, without loss of generality, that cycles $C_{u_i}^1$ and $C_{u_i}^2$ in G_{u_i} are marked for clause w. Out of the eight edges, $\{e_{u_1}^1, e_{u_1}^2, \ldots, e_{u_4}^1, e_{u_4}^2\}$, we know that exactly four belong to some cycle that was already packed (based on the truth value of each variable). Hence, we need to show that, out of the remaining four free edges, $\{r_w^1, r_w^2\}$ is linked to two of them and $\{b_w^1, b_w^2\}$ is linked to the other two. If so, then we can pack two additional cycles without violating the pairwise disjointness constraint. By construction, we known that (a) if u_i appears positively in w then $\{r_w^1, r_w^2\}$ is linked to $e_{u_i}^1$ and $\{b_w^1, b_w^2\}$ is linked to $e_{u_i}^2$ and (b) if u_i appears negatively in w then $\{r_w^1, r_w^2\}$ is linked to $e_{u_i}^2$ and $\{b_w^1, b_w^2\}$ is linked to $e_{u_i}^1$. However, we know that in each clause there are exactly two true literals (and hence two false literals). If both false literals are negated variables, say u_1 and u_2, then both variables must be true and therefore $\{r_w^1, r_w^2\}$ is linked to both $e_{u_1}^2$ and $e_{u_2}^2$ (which are free). If both false literals are positive variables, say u_1 and u_2, then both variables must be false and therefore $\{r_w^1, r_w^2\}$ is linked to both $e_{u_1}^1$ and $e_{u_2}^1$ (which are free). If u_1 is negative and u_2 is positive (in w) then both u_1 must be true and u_2 must be false and therefore $\{r_w^1, r_w^2\}$ is linked to both $e_{u_1}^2$ and $e_{u_2}^1$ (which are free). Using similar arguments for positive literals we can show that $\{b_w^1, b_w^2\}$ must be linked to the remaining two free edges, which completes the proof. ∎

Lemma 4.3. *If* $(G, k = 8|U| + \binom{4|W|}{2}, t = 1)$ *is a yes-instance of* Pairwise Disjoint Cycle Packing *then* ϕ *is a yes-instance of* 2/2/4-SAT.

Proof. Let $\mathcal{C}$ be a packing of distinct cycles of size $8|U| + \binom{4|W|}{2}$ such that all cycles pairwise intersect in at most one vertex. By Lemma 4.1, we know that such a packing is maximum. Moreover, any cycle not fully contained in a variable gadget must use at least two vertices from clause gadgets and the maximum number of such cycles is $\binom{4|W|}{2}$. Therefore, we can safely assume that $\mathcal{C}$ contains all $\binom{4|W|}{2} - 2|W|$ auxiliary cycles; if an auxiliary cycle is not in $\mathcal{C}$ then the corresponding pair of vertices from clause gadgets must belong to some other cycle in $\mathcal{C}$ (since $\mathcal{C}$ is maximum). Therefore we can replace that cycle with the auxiliary cycle. Clearly, each variable gadget can contribute at most eight cycles. Assume some gadget contributes less. Then, the maximum size of $\mathcal{C}$ would be $8|U| + \binom{4|W|}{2} - 1$, a contradiction. It follows that for each clause w, each pair in $\mathcal{P}_w = \{\{r_w^1, r_w^2\}, \{b_w^1, b_w^2\}\}$ must use exactly two external edges belonging to variable gadgets to form a cycle and these four edges must all belong to different variable gadgets; it is easy to check that using more than one external edge or any non-external edge from a variable gadget would reduce the number of cycles that can be packed within the gadget by at least one.

Assume that for some clause w the assignment implied by the packing does not result in exactly two true literals and two false literals. Then, we claim that one of the pairs in $\mathcal{P}_w$ cannot form a cycle. Consider the case where three literals are false (the other cases can be handled similarly). If all three false literals are negated variables, say u_1, u_2, and u_3, then all three variables must be true and therefore $\{r_w^1, r_w^2\}$ is linked to $e_{u_1}^2$, $e_{u_2}^2$, and $e_{u_3}^2$, which are free, but $\{b_w^1, b_w^2\}$ is linked to $e_{u_1}^1$, $e_{u_2}^1$, and $e_{u_3}^1$, which are not free. ∎

The next theorem follows from combining the previous two lemmas with the fact that 2/2/4-SAT is NP-hard.

Theorem 4.1. Pairwise Disjoint Cycle Packing *is NP-complete for* $t = 1$.

4.2. A polynomial kernel for $t = 1$. There are many similarities but also some subtle differences when dealing with the cases $t = 1$ and $t \geq 2$. For instance, for any value of $t \geq 1$, finding a flower of order k in the graph is sufficient to solve the problem. On the other hand, we can not apply Reduction Rule A2 (which is the same as Reduction Rule B2) for all vertices of degree two when $t \geq 2$. More importantly, finding two vertices in G with more than $2k$ common neighbors is enough to solve the problem for $t \geq 2$ but not for $t = 1$. As we shall see, this seemingly small difference requires major changes when dealing with the case $t = 1$. We start with some classical results and reduction rules which will be used throughout. Whenever some reduction rule applies, we apply the lowest-numbered applicable rule. For clarity, we will always denote a reduced instance by (G, k, t) (the one where reduction rules do not apply).

The first step in our kernelization algorithm is to run the algorithm of Theorem 2.1 and either output a trivial yes-instance (if k vertex disjoint cycles are found) or mark the vertices of the feedback vertex set and denote this set by F. We proceed with the following simple reduction rules to handle low-degree vertices and self-loops in the graph.

Reduction Rule B1. *Delete vertices of degree zero or one in G.*

Reduction Rule B2. *If there is a vertex v of degree exactly two in G then delete v and connect its two neighbors by a new edge.*

Reduction Rule B3. *If there exists a vertex $v \in V(G)$ with a self-loop then delete the loop (not the vertex) and decrease the parameter k by one.*

Reduction Rule B4. *If there is a pair of vertices u and v in $V(G)$ such that there are more than two parallel edges between them then reduce the multiplicity of the edge to two.*

Lemma 4.4. *Reduction Rule B2 is safe.*

Proof. Let (G, k, t) denote the original instance and let (G', k, t) denote the instance obtained after applying Reduction Rule B2, i.e. after deleting vertex v and adding an edge between its two neighbors u and w.

Assume (G', k, t) is a yes-instance and let $\mathcal{C}' = \{C'_1, \dots, C'_k\}$ denote the set of k distinct cycles satisfying $|V(C'_i) \cap V(C'_j)| \leq 1$, for all $1 \leq i, j \leq k$ and $i \neq j$. Consider a cycle $C' \in \mathcal{C}'$. If only one of u or w is in C' then C' is also a cycle in G. If both u and w are in C' then every other cycle in $\mathcal{C}'$ contains at most one of the two. Hence, if such a cycle exists we can obtain a corresponding cycle in G by simply replacing the edge (u, w) by the path formed by u, v, and w.

For the other direction, let (G, k, t) be a yes-instance and let $\mathcal{C} = \{C_1, \dots, C_k\}$ denote the corresponding solution. Assume, without loss of generality, that there exists a cycle $C \in \mathcal{C}$ such that $v \in V(C)$; otherwise $\mathcal{C}$ is also a solution for G'. Since v has degree two in G, both u and w must also belong to C. Let C' denote the cycle in G' obtained by deleting v and connecting u and w by an edge. We claim that $\mathcal{C}' = (\mathcal{C} \setminus \{C\}) \cup C'$ is a solution in G'. To see why, it is enough to note there can be at most one cycle in $\mathcal{C}$ containing v; otherwise at least one pair of cycles in $\mathcal{C}$ violates the disjointness constraint $|V(C_i) \cap V(C_j)| \leq 1$, $1 \leq i, j \leq k$ and $i \neq j$. ∎

Lemma 4.5. *Reduction Rule B3 is safe.*

Proof. Let (G, k, t) denote the original instance and let $(G', k-1, t)$ denote the instance obtained after applying Reduction Rule B3, i.e. after deleting the loop at vertex v.

Assume $(G', k-1, t)$ is a yes-instance and let $\mathcal{C}' = \{C'_1, \dots, C'_{k-1}\}$ denote the set of $k-1$ distinct cycles satisfying $|V(C'_i) \cap V(C'_j)| \leq 1$, for all $1 \leq i, j \leq k-1$ and $i \neq j$. Any cycle in $\mathcal{C}'$ can intersect with $\{v\}$ in at most one vertex. Therefore, adding the cycle corresponding to the loop at v we obtain a solution of size k for G.

For the other direction, let (G, k, t) be a yes-instance and let $\mathcal{C} = \{C_1, \dots, C_k\}$ denote the corresponding solution. Even though v could have multiple self-loops, each such loop corresponds to at most one cycle in $\mathcal{C}$. Therefore, $(G', k-1, t)$ is also a yes-instance. ∎

Lemma 4.6. *Reduction Rule B4 is safe.*

Proof. Assume u and v are connected by more than two parallel edges in G. Since $t = 1$, u and v can appear together in at most one cycle. Either this cycle includes other vertices, in which case at most one (u, v) edge is used, or the cycle consists of only u and v, in which case exactly two (u, v) edges are required. Therefore, reducing the multiplicity of any edge to two is safe. ∎

Once none of the above reduction rules are applicable, our next goal is to bound the maximum degree in the graph. To do so, we make use of the following.

Lemma 4.7 (see [8]). *Given a (multi) graph G, an integer k, and a vertex $v \in V(G)$, there is a polynomial-time algorithm that either finds a v-flower of order k or finds a set Z_v such that $Z_v \subseteq V(G) \setminus \{v\}$ intersects all cycles passing through v, $|Z_v| \leq 2k$, and there are at most $2k$ edges incident to v and with second endpoint in Z_v.*

A *q-star*, $q \geq 1$, is a graph with $q+1$ vertices, one vertex of degree q and all other vertices of degree 1. Let G be a bipartite graph with vertex bipartition (A, B). A set of edges $M \subseteq E(G)$ is called a *q-expansion* of A into B if

- Every vertex of A is incident with exactly q edges of M
- M saturates exactly $q|A|$ vertices in B, i.e. there is a set of $q|A|$ vertices in B that are incident to edges in M.

Lemma 4.8 (see [8, 30]). *Let q be a positive integer and G be a bipartite graph with vertex bipartition (A, B) such that $|B| \geq q|A|$ and there are no isolated vertices in B. Then, there exist nonempty vertex sets $X \subseteq A$ and $Y \subseteq B$ such that:*

- *X has a q-expansion into Y and*
- *no vertex in Y has a neighbour outside X, i.e. $N(Y) \subseteq X$.*

Furthermore, the sets X and Y can be found in time polynomial in the size of G.

For every vertex $v \in V(G)$ of high degree (which will be specified later), we apply the algorithm of Lemma 4.7. If the algorithm finds a v-flower of order k, the following reduction rule allows us to deal with it.

Reduction Rule B5. *If G has a vertex v such that there is a v-flower of order at least k then return a trivial yes-instance.*

Hence, in what follows we assume that no such flower was found but instead we have a set Z_v of size at most $2k$ such that $Z_v \subseteq V(G)$ intersects all cycles passing through v. Consider the connected components of the graph $G[V(G) \setminus (Z_v \cup \{v\})]$. At most $k-1$ of those components can contain a cycle, as otherwise we again have a trivial yes-instance consisting of k vertex disjoint cycles.

Reduction Rule B6. *If there are k or more components in $G \setminus (\{v\} \cup Z_v)$ containing a cycle then return a trivial yes-instance.*

Moreover, for every component D of $G[V(G) \setminus (Z_v \cup \{v\})]$, we have $|N_G(v) \cap V(D)| \leq 1$. In other words, v has at most one neighbor in any component and out of those components at most $k-1$ are not trees (see Figure 5). Let $\mathcal{D} = \{D_1, D_2, \ldots, D_q\}$ denote those trees in which v has a neighbor. Since the minimum degree of the graph is three, every leaf of a tree in $\mathcal{D}$ must have at least one neighbor in Z_v.

Lemma 4.9. *Let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be a solution in G and let C be a cycle in $\mathcal{C}$ such that $V(C) \cap (Z_v \cup \{v\}) \neq \emptyset$. Then, C can intersect with at most $2k+1$ components in $\mathcal{D}$ and therefore the solution $\mathcal{C}$ can intersect with at most $2k^2 + k$ components in $\mathcal{D}$.*

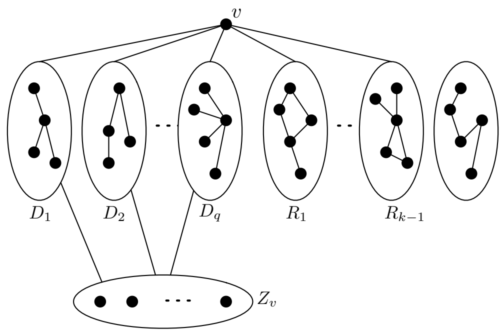

FIG. 5. *A vertex $v \in V(G)$, its corresponding set Z_v, and the set $\mathcal{D} = \{D_1, D_2, \ldots, D_q\}$*

Proof. Consider any cycle $C \in \mathcal{C}$ that intersects $Z_v \cup \{v\}$. We contract all edges of C that are not incident to any vertex in $Z_v \cup \{v\}$ and denote this new cycle by C'. Between any two consecutive vertices in $C' \cap (Z_v \cup \{v\})$, there is either an edge from $E(G)$ or a path passing through a vertex $z \notin Z_v \cup \{v\}$, where z corresponds to a contracted path from some component in $G \setminus (Z_v \cup \{v\})$. Since $|Z_v \cup \{v\}| \leq 2k+1$, there can be at most $2k+1$ such vertices. Therefore, any cycle $C \in \mathcal{C}$ can intersect with at most $2k+1$ components from $G \setminus (Z_v \cup \{v\})$. Summing up for the k cycles in $\mathcal{C}$, we get the desired bound. ∎

We now construct a bipartite graph $\mathcal{H}$ with bipartition $(A = Z_v, B = \mathcal{D})$. We slightly abuse notation and assume that every component in $\mathcal{D}$ corresponds to a vertex in B and every vertex in Z_v corresponds to a vertex in A. For every $D_i \in \mathcal{D}$ and for every $z \in Z_v$, $(D_i, z) \in E(\mathcal{H})$ if and only if there exists $u \in V(D_i)$ such that $(u, z) \in E(G)$. After exhaustive application of Reduction Rule B4, every pair of vertices in G can have at most two edges between them. In particular, there can be at most two edges between any $z \in Z_v$ and v. Therefore, if the degree of v in G is more than $(2k^2+k+2)2k+3k-1$ then the number of components $|\mathcal{D}|$ is at least $(2k^2+k+2)2k$ (taking into account the at most $k-1$ neighbors of v in components containing a cycle as well as the at most $2k$ edges incident to v and some vertex in Z_v). Consequently, $|\mathcal{D}| \geq (2k^2+k+2)|Z_v|$. We are now ready to state our main reduction rule.

REDUCTION RULE B7. *If there exists a vertex $v \in V(G)$ such that $d_G(v) > (2k^2+k+2)2k+3k-1$ then apply Lemma 4.8 with $q = 2k^2+k+2$ in the bipartite graph $\mathcal{H}$.*
- *Let $\mathcal{D}' \subseteq \mathcal{D}$ and $Z'_v \subseteq Z_v$ be the sets obtained after applying Lemma 4.8 with $q = 2k^2+k+2$, $A = Z_v$, and $B = \mathcal{D}$, such that Z'_v has a $(2k^2+k+2)$-expansion into $\mathcal{D}'$ in $\mathcal{H}$.*
- *Delete all the edges of the form $(u, v) \in E(G)$ such that $u \in D_i$ and $D_i \in \mathcal{D}'$.*
- *Add two parallel edges between v and every vertex in Z'_v.*

Lemma 4.10. *Reduction Rule B7 is safe.*

Proof. Let (G', k, t) be the instance obtained after applying Reduction Rule B7, let (G, k, t) be the original instance, and let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be the cycles in G satisfying the pairwise intersection constraint. We let $\mathcal{C}_v \subseteq \mathcal{C}$ be the set of cycles containing the high degree vertex v. Note that any such cycle must also contain at least one vertex from Z_v. From Lemma 4.8 and Reduction Rule B7, we know that $N_G(\mathcal{D}') \subseteq Z'_v$. Hence, any cycle $C \in \mathcal{C}_v$ which contains a vertex from $\mathcal{D}'$ must also contain a vertex from Z'_v. In other words, whenever a cycle passes through $\mathcal{D}'$ it must also pass through Z'_v. We let $\mathcal{C}'_v \subseteq \mathcal{C}_v$ denote all these cycles. Note that any cycle in $\mathcal{C} \setminus \mathcal{C}'_v$ is not modified in G' and hence such cycles can still be packed in G'. Moreover, for any two cycles C_1 and C_2 in $\mathcal{C}'_v$, we have $(V(C_1) \cap Z'_v) \cap (V(C_2) \cap Z'_v) = \emptyset$, as both C_1 and C_2 contain v. Now, let $V(C) \cap Z'_v$ denote the set of vertices in cycle $C \in \mathcal{C}'_v$. We can pick any vertex $z \in V(C) \cap Z'_v$ and replace the cycle C with the cycle consisting of only z and v (as we added two edges between them). Consequently, for any packing $\mathcal{C}$ of size k in G we can find a corresponding packing $\mathcal{C}'$ of size k in G', as needed.

Assume (G', k, t) is a yes-instance and let $\mathcal{C}' = \{C'_1, \dots, C'_k\}$ be a collection of k cycles pairwise intersecting in at most one vertex. Consider those cycles in $\mathcal{C}'$ which contain an edge $(v, z) \notin E(G)$ $(z \in Z'_v)$. Such cycles can be of two types. Either they contain a single edge $(v, z) \notin E(G)$ or they contain two edges $(v, z) \notin E(G)$ and $(v, z') \notin E(G)$, with z' possibly equal to z. Therefore, for every vertex $z \in Z'_v$, we need to have two components whose intersection with $\mathcal{C}$ is empty. However, we know that, for every $z \in Z'_v$, z is connected to at least $q = 2k^2 + k + 2$ distinct components in $\mathcal{D}'$. By Lemma 4.9, $\mathcal{C}$ intersects at most $2k^2 + k$ components in $\mathcal{D}'$. In other words, for every vertex $z \in Z'_v$ there are at least two components in $\mathcal{D}'$, say D_1 and D_2, such that $V(D_1) \cap V(\mathcal{C}) = V(D_2) \cap V(\mathcal{C}) = \emptyset$. Consequently, we can find a solution in G by replacing any edge of the form $(v, z) \notin E(G)$ by a path that starts from z, goes through D_1 (or D_2), and finally reaches v. ∎

We now have all the required ingredients to bound the size of our kernel. From Theorem 2.1, we know that the graph has a feedback vertex set F of size at most $\mathcal{O}(k \log k)$. The degree of any vertex in the graph is at least three (Reduction Rule B2) and at most in $\mathcal{O}(k^3)$ (Reduction Rule B7). Theorem 4.2 follows from combining these facts with Lemma 4.11.

Lemma 4.11 (see [8]). *Let $G = (V, E)$ be an undirected (multi) graph having minimum degree at least three, maximum degree at most d, and a feedback vertex set of size at most r. Then, $|V(G)| < (d + 1)r$ and $|E(G)| < 2dr$.*

Theorem 4.2. *For $t = 1$,* Pairwise Disjoint Cycle Packing *admits a kernel with $\mathcal{O}(k^4 \log k)$ vertices and $\mathcal{O}(k^4 \log k)$ edges.*

4.3. A polynomial compression for $t \geq 2$ (independent of t). When $t \geq 2$, finding two vertices in G with $2k$ internally vertex-disjoint paths connecting them is enough to pack k cycles pairwise intersecting in at most 2 vertices. Hence, bounding the maximum degree is relatively easy. We first mark the feedback vertex set F and exhaustively apply Reduction Rule B1 and the following modified variant of Reduction Rule B2.

Reduction Rule B8. *If there exists a set of vertices $P = \{v_1, \dots, v_{t+2}\} \subseteq V(G)$ such that $G[P]$ is a path and $d_G(v_i) = 2$, $2 \leq i \leq t + 1$, then contract the edge $v_1 v_2$.*

As before, for every vertex $v \in V(G)$, we apply the algorithm of Lemma 4.7. If the algorithm finds a v-flower of order k, we apply Reduction Rule B5. Otherwise, consider the connected components of the graph $G[V(G) \setminus (Z_v \cup \{v\})]$. We ignore the at most $k-1$ components that can contain a cycle and focus on the set $\mathcal{D} = \{D_1, D_2, \dots, D_q\}$ of trees in which v has a neighbor (recall that $|N_G(v) \cap V(D)| \leq 1$ for all $D \in \mathcal{D}$ and each component D must have a neighbor in Z_v).

Reduction Rule B9. *If $|\mathcal{D}| > 4k - 2$ (or equivalently if $d_G(v) > 7k - 3$) return a trivial yes-instance.*

Lemma 4.12. *Reduction Rule B9 is safe.*

Proof. Let v be a vertex in $V(G)$, Z_v be the set given by Lemma 4.7, and $\mathcal{D} = \{D_1, D_2, \ldots, D_q\}$ be the set of trees in which v has a neighbor. Observe that each $D \in \mathcal{D}$ contains at least one vertex which is adjacent to some vertex in Z_v. Let $Z_v = \{z_1, z_2, \ldots, z_l\}$, where $l \leq 2k$. For $i = 1$ to l (in increasing order), we let $\mathcal{D}_i = \{D \mid D \in \mathcal{D} \wedge z_i \in N_G(D) \cap Z_v \wedge \forall_{i'<i} D \notin \mathcal{D}_{i'}\}$. In other words, $\mathcal{D}_i$ contains a component $D \in \mathcal{D}$ whenever D contains a vertex which is adjacent to z_i and D does not belong to $\mathcal{D}_{i'}$, for all $i' < i$.

Once we have constructed the set $\mathcal{D}_i$, for all $i \in [l]$, we arbitrarily pair the components in $\mathcal{D}_i$ (all pairs being disjoint); there can be at most one component in $\mathcal{D}_i$ which is left unpaired. If we can find k pairs in $\cup_{i\in[l]}\mathcal{D}_i$, then for each pair $(D_1, D_2) \in \mathcal{D}_i$ we can pack a cycle formed by vertices in $V(D_1) \cup V(D_2) \cup \{v, z_i\}$. Every pair of such cycles intersects in at most two vertices, namely $\{v, z_i\}$, and we have a total of at least k cycles, as needed. Otherwise, $|\mathcal{D}| \leq 2(k-1) + l \leq 4k - 2$. Since v can have at most $k-1$ additional neighbors in $G[V(G) \setminus (Z_v \cup \{v\})]$ and there are at most $2k$ edges incident to v with second endpoint in Z_v, the bound on $d_G(v)$ follows. ◻

Having bounded the maximum degree of any vertex by $\mathcal{O}(k)$, we immediately obtain a bound of $\mathcal{O}(k^2 \log k)$ on $|T_{\leq 1}|$, $|T_{\geq 3}|$, and the number of maximal degree-two paths in T_2. Recall that $T_{\leq 1}$, T_2, and $T_{\geq 3}$, are the sets of vertices in $T = G[V(G) \setminus F]$ having degree at most one in T, degree exactly two in T, and degree greater than two in T, respectively. To bound the size of T_2, note that if we mark all vertices in $F \cup N_G(F)$ we would have marked a total of $\mathcal{O}(k^2 \log k)$ vertices and the only unmarked vertices form (not necessarily maximal) degree-two paths in T_2 (and G), which we call segments. However, we know from Reduction Rule B8 that the size of any segment is at most $t+1$. Moreover, the total number of such segments is at most $\mathcal{O}(k^2 \log k)$. Putting it all together, we now have a kernel with $\mathcal{O}(tk^2 \log k)$ vertices.

Lemma 4.13. *For any $t \geq 2$,* Pairwise Disjoint Cycle Packing *admits a kernel with $\mathcal{O}(tk^2 \log k)$ vertices.*

More work is needed to get rid of the dependence on t. The first step is to show that we can solve Pairwise Disjoint Cycle Packing in $c^{p(k)} n^{\mathcal{O}(1)}$ time, where c is a fixed constant and $p(.)$ is a polynomial function in k. In the second step, we introduce a "succinct" version of Pairwise Disjoint Cycle Packing, namely Succinct Pairwise Disjoint Cycle Packing, and show that we can reduce Pairwise Disjoint Cycle Packing to an instance of Succinct Pairwise Disjoint Cycle Packing where all the information can be encoded using a number of bits polynomially bounded in k alone. As is usually the case, we assume that the weight of a set of vertices/edges is equal to the sum of the weights of the individual vertices/edges.

Succinct Pairwise Disjoint Cycle Packing **Parameter:** k
Input: An undirected (multi) graph G, integers k and t, a weight function $\alpha : V(G) \to \mathbb{N}$, and a weight function $\beta : E(G) \to \mathbb{N}$.
Question: Does G have at least k distinct cycles $C_1, \ldots, C_k$ such that $\alpha(V(C_i) \cap V(C_j)) \leq t$ and $\beta(E(C_i) \cap E(C_j)) \leq t$ for all $i \neq j$?

Lemma 4.14. *For any $t \geq 2$,* Pairwise Disjoint Cycle Packing *can be solved in $2^{k^3 \log k} n^{\mathcal{O}(1)}$ time.*

Proof. We first obtain the kernel guaranteed by Lemma 4.13. Note that both the number of vertices having degree three or more and the number of segments in the reduced instance is bounded by $\mathcal{O}(k^2 \log k)$. We assume, without loss of generality, that any cycle in the solution must contain at least one degree-three vertex (if some components of G consist of degree-two cycles we can greedily pack those cycles). Hence, we can guess, for each cycle, which of those $\mathcal{O}(k^2 \log k)$ vertices and segments will be included in $\mathcal{O}(2^{k^2 \log k})$ time. Repeating this process for each of the k cycles and checking that they satisfy the pairwise intersection constraint can therefore be accomplished in $\mathcal{O}(2^{k^3 \log k})$ time. ◻

Theorem 4.3. *For any $t \geq 2$, we can compress an instance of* Pairwise Disjoint Cycle Packing *to an equivalent instance of* Succinct Pairwise Disjoint Cycle Packing *using at most $\mathcal{O}(k^5 \log^2 k)$ bits. In other words,* Pairwise Disjoint Cycle Packing *admits a polynomial compression.*

Proof. Given an instance of Pairwise Disjoint Cycle Packing we apply the kernelization algorithm to obtain an equivalent instance on at most $\mathcal{O}(tk^2 \log k)$ vertices. Then, we create an equivalent instance of Succinct Pairwise Disjoint Cycle Packing, where each vertex is assigned weight 1 and each edge is assigned weight 0. Note that in this new instance we still have a total number of at most $\mathcal{O}(k^2 \log k)$ segments each of size at most $t + 1$. We replace each such segment by an edge whose weight is equal to the number of vertices on the segment, which requires $\log t \leq \log n$ bits at most. However, if $\log n > k^3 \log k$, by Lemma 4.14, we can solve the corresponding Pairwise Disjoint Cycle Packing instance in time polynomial in n (and obtain a polynomial kernel). Hence, the number of bits required to encode the weight of each such edge is at most $k^3 \log k$. Multiplying by the total number of segments we obtain the claimed bound. ◻

5. Conclusion.

To summarize, we have showed that when relaxing the Disjoint Cycle Packing problem by allowing pairwise overlapping cycles (i.e. Pairwise Disjoint Cycle Packing) then polynomial kernels are relatively easy to obtain, even when cycles can share at most one vertex. On the other hand, relaxing the Disjoint Cycle Packing problem by limiting the number of cycles each vertex can appear in has much more diverse consequences on the kernelization complexity. However, even though we obtain a polynomial kernel for Almost Disjoint Cycle Packing with $t = \frac{k}{c}$, where c is a constant, it is not clear whether the problem is even NP-complete in this case. It would be very interesting to settle this question (probably more interesting to settle it negatively). Finally, it would also be interesting to consider relaxed variants of more problems known to admit no polynomial kernels and determine whether (for any of them) there exists a "smooth" relationship between relaxation parameters and kernelization complexity, i.e. whether kernel bounds improve as the relaxation parameter increases.

REFERENCES

[1] H. Abasi, N. H. Bshouty, A. Gabizon, and E. Haramaty, *On r-simple k-path*, in Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS, 2014, pp. 1–12.
[2] A. Agrawal, D. Lokshtanov, D. Majumdar, A. E. Mouawad, and S. Saurabh, *Kernelization of cycle packing with relaxed disjointness constraints*, in 43rd International Colloquium on Automata, Languages, and Programming, ICALP, 2016, pp. 26:1 – 26:14.
[3] H. L. Bodlaender, *A linear-time algorithm for finding tree-decompositions of small treewidth*, SIAM Journal on Computing, 25 (1996), pp. 1305–1317.

[4] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin, *On problems without polynomial kernels*, Journal of Computer and System Sciences, 75 (2009), pp. 423–434.

[5] H. L. Bodlaender and A. M. C. A. Koster, *Combinatorial optimization on graphs of bounded treewidth*, The Computer Journal, 51 (2008), pp. 255–269.

[6] H. L. Bodlaender, S. Thomassé, and A. Yeo, *Kernel bounds for disjoint cycles and disjoint paths*, Theoretical Computer Science, 412 (2011), pp. 4570–4578.

[7] B. Courcelle, *The monadic second-order logic of graphs. I. recognizable sets of finite graphs*, Information and Computation, 85 (1990), pp. 12 – 75.

[8] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized Algorithms*, Springer, 2015.

[9] H. Dell and D. Marx, *Kernelization of packing problems*, in Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, 2012, pp. 68–81.

[10] H. Dell and D. van Melkebeek, *Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses*, Journal of the ACM, 61 (2014), pp. 23:1–23:27.

[11] R. Diestel, *Graph Theory, 4th Edition*, vol. 173 of Graduate texts in mathematics, Springer, 2012.

[12] R. G. Downey and M. R. Fellows, *Parameterized complexity*, Springer-Verlag, 1997.

[13] A. Drucker, *New limits to classical and quantum instance compression*, SIAM Journal on Computing, 44 (2015), pp. 1443–1479.

[14] P. Erdős and L. Pósa, *On independent circuits contained in a graph*, Canadian Journal of Mathematics, 17 (1965), pp. 347–352.

[15] H. Fernau, A. López-Ortiz, and J. Romero, *Kernelization algorithms for packing problems allowing overlaps*, in Theory and Applications of Models of Computation - 12th Annual Conference, TAMC, 2015, pp. 415–427.

[16] J. Flum and M. Grohe, *Parameterized Complexity Theory*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[17] L. Fortnow and R. Santhanam, *Infeasibility of instance compression and succinct PCPs for NP*, Journal of Computer and System Sciences, 77 (2011), pp. 91–106.

[18] A. Gabizon, D. Lokshtanov, and M. Pilipczuk, *Fast algorithms for parameterized problems with relaxed disjointness constraints*, in Algorithms - 23rd Annual European Symposium, ESA, 2015, pp. 545–556.

[19] M. Grohe, *Logic, graphs, and algorithms.*, Electronic Colloquium on Computational Complexity (ECCC), 14 (2007).

[20] D. Hermelin, S. Kratsch, K. Soltys, M. Wahlström, and X. Wu, *A completeness theory for polynomial (turing) kernelization*, Algorithmica, 71 (2015), pp. 702–730.

[21] D. Hermelin and X. Wu, *Weak compositions and their applications to polynomial lower bounds for kernelization*, in Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, 2012, pp. 104–113.

[22] S. Kratsch, *Recent developments in kernelization: A survey*, Bulletin of the EATCS, 113 (2014).

[23] D. Lokshtanov, N. Misra, and S. Saurabh, *Kernelization - preprocessing with a guarantee*, in The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday, 2012, pp. 129–161.

[24] D. Lokshtanov, F. Panolan, M. S. Ramanujan, and S. Saurabh, *Lossy kernelization*, CoRR, abs/1604.04111 (2016).

[25] R. Niedermeier, *Invitation to fixed-parameter algorithms*, Oxford University Press, Oxford, 2006.

[26] J. Ramon and S. Nijssen, *Polynomial-delay enumeration of monotonic graph classes*, The Journal of Machine Learning Research, 10 (2009), pp. 907–929.

[27] D. Ratner and M. K. Warmuth, *NxN puzzle and related relocation problem*, Journal of Symbolic Computation, 10 (1990), pp. 111–138.

[28] J. Romero and A. López-Ortiz, *The $\mathcal{G}$-packing with t-overlap problem*, in Algorithms and Computation - 8th International Workshop, WALCOM, 2014, pp. 114–124.

[29] J. Romero and A. López-Ortiz, *A parameterized algorithm for packing overlapping subgraphs*, in Computer Science - Theory and Applications - 9th International Computer Science Symposium in Russia, CSR, 2014, pp. 325–336.

[30] S. Thomassé, *A $4k^2$ kernel for feedback vertex set*, ACM Transactions on Algorithms, 6 (2010), pp. 32:1–32:8.