Kernelization of Cycle Packing with Relaxed Disjointness Constraints

Akanksha Agrawal[1], Daniel Lokshtanov[2], Diptapriyo Majumdar[3], Amer E. Mouawad[4], and Saket Saurabh[5]

1 **University of Bergen, Bergen, Norway**
akanksha.agrawal@uib.no
2 **University of Bergen, Bergen, Norway**
daniel.lokshtanov@uib.no
3 **Institute of Mathematical Sciences, Chennai, India**
diptapriyom@imsc.res.in
4 **University of Bergen, Bergen, Norway**
a.mouawad@uib.no
5 **University of Bergen, Bergen, Norway, and**
Institute of Mathematical Sciences, Chennai, India
saket@imsc.res.in

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 NP $\subseteq$ 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 $0 \leq \epsilon < 1$.

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

Keywords and phrases parameterized complexity, cycle packing, kernelization, relaxation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.26

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

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 26; pp. 26:1–26:14

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

preprocessing routines. Progress in parameterized complexity [11] made such an analysis possible. In parameterized complexity, each problem instance is coupled with a parameter k 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 that a problem does not admit a polynomial kernel [3, 12, 16, 18], or a kernel of a specific size [8, 9, 19]. We refer the reader to the survey articles by Kratsch [20] and Lokshtanov et al. [21] 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 $C \in \mathcal{C}$ induces 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 [2, 4]. Bodlaender et al. [5] showed that, when parameterized by k, Disjoint Cycle Packing does not admit a polynomial kernel unless NP $\subseteq$ 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 $\lvert V(C_i) \cap V(C_j) \rvert \leq t$ 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 [24]. 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

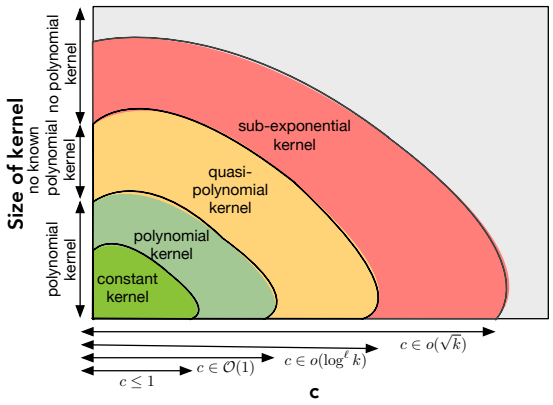

Figure 1 Spectrum of kernelization algorithms for Almost Disjoint Cycle Packing as c grows in the denominator of $t = \frac{k}{c}$.

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.

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 NP $\subseteq$ 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. Most of the technical details and proofs have been omitted from this extended abstract.

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 NP $\subseteq$ 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 bounds. The Disjoint Cycle Packing problem falls into categories (a) and (c) and is the main subject of this work. Before we state 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. [17], 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. [26, 27] and Fernau et al. [14] 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 [10] 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.

A *path* in a graph is a sequence of distinct vertices $v_0, v_1, \ldots, v_\ell$ such that (v_i, v_{i+1}) is an edge for all $0 \leq i < \ell$. A *cycle* in a graph is a sequence of distinct vertices $v_0, v_1, \ldots, v_\ell$ such that $(v_i, v_{(i+1) \mod \ell})$ 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 P, we use $V(P)$ to denote the set of vertices in the path P and the length of P is denoted by $|P|$ (i.e, $|P| = |V(P)|$). For a cycle C, we use $V(C)$ to denote the set of vertices in the cycle C and length of C, denoted by $|C|$, is $|V(C)|$. For a path or a cycle Q we use $N_G(Q)$ and $N_G[Q]$ to denote the set $N_G(V(Q))$ 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 $P = \{v_1, \ldots, v_\ell\}$ in G forms a *degree-two path* if P is a path and all vertices $\{v_1, \ldots, v_\ell\}$ 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 1** ([13]). *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 [11], Flum and Grohe [15], Niedermeier [23], and the more recent book by Cygan et al. [7].

▶ **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.

▶ **Definition 3.** Let $L, R \subseteq \Sigma \times \mathbb{N}$ be two parameterized problems. An algorithm $\mathcal{A}$ is called a *polynomial parameter transformation* from L to R if, given an instance (I, k) of problem :, $\mathcal{A}$ 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 4** ([7]). *Let $L, R \subseteq \Sigma \times \mathbb{N}$ be two parameterized problems and assume there exists a polynomial parameter transformation from L to R. Then, if R does not admit a polynomial compression, neither does L. In particular, if R does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly then the same holds for L.*

3 Almost Disjoint Cycle Packing

As previously noted, Bodlaender et al. [5] showed that Disjoint Cycle Packing admits no polynomial kernel unless NP $\subseteq$ coNP/poly. On the other hand, finding k distinct cycles in a graph is solvable in time polynomial in n and k [24]. 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 monotonically increasing computable function $f : \mathbb{N} \to \mathbb{R}_+$, 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 [5]. 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 NP $\subseteq$ 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. Armed with Theorem 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 [6] 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).

▶ **Theorem 5** ([6]). *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 6.** 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.*

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 $\subseteq$ 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 [5] which is known to admit no polynomial compression unless NP $\subseteq$ coNP/poly.

Let Σ_q be an alphabet set of q elements. By $\Sigma_q^\star$ we denote the set of all strings over Σ_q. A factor of a string $\bar{y} = y_1 y_2 \dots y_n \in \Sigma_q^\star$ 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 a disjoint factor 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 $x, \hat{x} \in \Sigma_q$, (s_x, e_x) and $(s_{\hat{x}}, e_{\hat{x}})$ are disjoint factors.

Disjoint Factors	**Parameter:** q
Input: Alphabet set Σ_q, string $\bar{y} \in \Sigma_q^\star$	
Question: Does $\bar{y}$ have a disjoint factor?	

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 $\bar{y} = y_1 y_2 \dots y_n \in \Sigma_q^\star$ we create a path $P_y = (u, \hat{y}_1, \hat{y}_2, \dots \hat{y}_n, u')$. 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 $E(G)$. For each $x_i \in \Sigma$, 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 of 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'). We set $k = tq + t + l + 1$, which completes the construction.

▶ **Proposition 7.** *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 is a path $P_{B'}$ 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, s' in H.*

Applying Proposition 7 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 R' vertices 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 path P_y must be used $t-1$ times and can therefore participate in one additional cycle (which contains vertices in $V(P_y)$). We associate each such extra cycle with a factor.

▶ **Theorem 8.** *Let $f : \mathbb{N} \to \mathbb{R}_{\geq 1}$ be a computable monotonically increasing function such that $f(k) \in \mathcal{O}(k^\epsilon)$, where $0 < \epsilon \leq 1$. Then,* Almost Disjoint Cycle Packing *admits no polynomial kernel over L_f unless NP $\subseteq$ coNP/poly.*

3.2 A kernel for Almost Disjoint Cycle Packing

Let $f : \mathbb{N} \to \mathbb{R}_{\geq 1}$ be a computable monotonically increasing 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 below is inspired by the *lossy kernel* for the Cycle Packing problem given in [22]. 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 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 $G[T_2]$. Next, we bound the size of $T_{\leq 1}$, which implies a bound on the size of $T_{\geq 3}$ 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.

Bounding the size of $T_{\leq 1}$. First, we get rid of vertices of degree one and two using Reduction Rules A1 and A2.

▶ **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.*

▶ **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 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 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 [22].** *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.*

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. We show that if $F_{\lceil c \rceil}$ exists then we have a yes-instance.

▶ **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.*

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. 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 9.** *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 $\mathcal{C}$.*

Using Lemma 9, 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.*

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 10.** *Let $f : \mathbb{N} \to \mathbb{R}_{\geq 1}$ be a computable monotonically increasing 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 10 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. 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 NP-complete by Ratner and Warrnuth [25].

▶ **Theorem 11.** Pairwise Disjoint Cycle Packing *is NP-complete for* $t = 1$.

4.1 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, not all vertices of degree two can be bypassed 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.

The first step in our kernelization algorithm is to run the algorithm of Theorem 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.*

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 12** ([7]). *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 (i) every vertex of A is incident with exactly q edges of M and (ii) M saturates exactly $q|A|$ vertices in B, i.e. there is a set of $q|A|$ vertices in B which are incident to edges in M.

▶ **Lemma 13** (See [7, 28]). *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 12. 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. 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 14.** *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 $\mathcal{D}$.*

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 13 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 13 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.*

We now have all the required ingredients to bound the size of our kernel. From Theorem 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 16 follows from combining these facts with Lemma 15.

▶ **Lemma 15** ([7]). *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 16.** *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.*

4.2 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 $P = \{v_1, \ldots, v_{t+2}\} \subseteq V(G)$ such that $G[P]$ is a path, $d_G(v_i) = 2$, $2 \leq i \leq t+1$, and $|P| \geq t+2$, then contract the edge v_1v_2.*

As before, for every vertex $v \in V(G)$, we apply the algorithm of Lemma 12. 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 $\mathcal{D} = \{D_1, D_2, \ldots, D_q\}$ 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.*

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 17.** *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.

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

▶ **Theorem 19.** *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.*

References

1. Hasan Abasi, Nader H. Bshouty, Ariel Gabizon, and Elad Haramaty. On r-simple k-path. In *Mathematical Foundations of Computer Science 2014 – 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, pages 1–12, 2014. doi:10.1007/978-3-662-44465-8_1.
2. Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, December 1996.
3. Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009. doi:10.1016/j.jcss.2009.04.001.
4. Hans L. Bodlaender and Arie M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *Comput. J.*, 51(3):255–269, May 2008.
5. Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theor. Comput. Sci.*, 412(35):4570–4578, 2011. doi:10.1016/j.tcs.2011.04.039.
6. Bruno Courcelle. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.
7. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
8. Holger Dell and Dániel Marx. Kernelization of packing problems. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 68–81, 2012. URL: http://portal.acm.org/citation.cfm?id=2095122&CFID=63838676&CFTOKEN=79617016.
9. Holger Dell and Dieter van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. *J. ACM*, 61(4):23:1–23:27, 2014. doi:10.1145/2629620.
10. Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.
11. Rod G. Downey and Michael R. Fellows. *Parameterized complexity*. Springer-Verlag, 1997.
12. Andrew Drucker. New limits to classical and quantum instance compression. *SIAM J. Comput.*, 44(5):1443–1479, 2015. doi:10.1137/130927115.
13. Paul Erdős and Lajos Pósa. On independent circuits contained in a graph. *Canad. Journ. Math*, 17(0):347–352, 1965.
14. Henning Fernau, Alejandro López-Ortiz, and Jazmín Romero. Kernelization algorithms for packing problems allowing overlaps. In *Theory and Applications of Models of Computation – 12th Annual Conference, TAMC 2015, Singapore, May 18-20, 2015, Proceedings*, pages 415–427, 2015. doi:10.1007/978-3-319-17142-5_35.
15. J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

16 Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *J. Comput. Syst. Sci.*, 77(1):91–106, 2011. doi:10.1016/j.jcss.2010.06.007.

17 Ariel Gabizon, Daniel Lokshtanov, and Michal Pilipczuk. Fast algorithms for parameterized problems with relaxed disjointness constraints. In *Algorithms – ESA 2015 – 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 545–556, 2015. doi:10.1007/978-3-662-48350-3_46.

18 Danny Hermelin, Stefan Kratsch, Karolina Soltys, Magnus Wahlström, and Xi Wu. A completeness theory for polynomial (turing) kernelization. *Algorithmica*, 71(3):702–730, 2015. doi:10.1007/s00453-014-9910-8.

19 Danny Hermelin and Xi Wu. Weak compositions and their applications to polynomial lower bounds for kernelization. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 104–113, 2012. URL: http://portal.acm.org/citation.cfm?id=2095125&CFID=63838676&CFTOKEN=79617016.

20 Stefan Kratsch. Recent developments in kernelization: A survey. *Bulletin of the EATCS*, 113, 2014. URL: http://eatcs.org/beatcs/index.php/beatcs/article/view/285.

21 Daniel Lokshtanov, Neeldhara Misra, and Saket 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*, pages 129–161, 2012. doi:10.1007/978-3-642-30891-8_10.

22 Daniel Lokshtanov, Fahad Panolan, M.S. Ramanujan, and Saket Saurabh. Lossy kernelization. *arXiv:1604.04111*, 2016.

23 Rolf Niedermeier. *Invitation to fixed-parameter algorithms*. Oxford University Press, Oxford, 2006.

24 Jan Ramon and Siegfried Nijssen. Polynomial-delay enumeration of monotonic graph classes. *The Journal of Machine Learning Research*, 10:907–929, 2009.

25 Daniel Ratner and Manfred K. Warmuth. NxN puzzle and related relocation problem. *J. Symb. Comput.*, 10(2):111–138, 1990. doi:10.1016/S0747-7171(08)80001-6.

26 Jazmín Romero and Alejandro López-Ortiz. The $\mathcal{G}$-packing with t-overlap problem. In *Algorithms and Computation – 8th International Workshop, WALCOM 2014, Chennai, India, February 13-15, 2014, Proceedings*, pages 114–124, 2014. doi:10.1007/978-3-319-04657-0_13.

27 Jazmín Romero and Alejandro López-Ortiz. A parameterized algorithm for packing overlapping subgraphs. In *Computer Science – Theory and Applications – 9th International Computer Science Symposium in Russia, CSR 2014, Moscow, Russia, June 7-11, 2014. Proceedings*, pages 325–336, 2014. doi:10.1007/978-3-319-06686-8_25.

28 Stéphan Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Trans. Algorithms*, 6(2):32:1–32:8, April 2010. doi:10.1145/1721837.1721848.