A Linear Kernel for Co-Path/Cycle Packing

Zhi-Zhong Chen[1], Michael Fellows[2], Bin Fu[3], Haitao Jiang[5], Yang Liu[3], Lusheng Wang[4], and Binhai Zhu[5]

[1] Department of Mathematical Sciences, Tokyo Denki University, Hatoyama, Saitama 350-0394, Japan
zzchen@mail.dendai.ac.jp
[2] The University of New Castle, Callaghan, NSW 2308, Australia
michael.fellows@newcastle.edu.au
[3] Department of Computer Science, University of Texas-American, Edinburg, TX 78739-2999, USA
binfu,yliu@panam.edu
[4] Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong
cswangl@cityu.edu.hk
[5] Department of Computer Science, Montana State University, Bozeman, MT 59717-3880, USA
htjiang,bhz@cs.montana.edu

Abstract. Bounded-Degree Vertex Deletion is a fundamental problem in graph theory that has new applications in computational biology. In this paper, we address a special case of Bounded-Degree Vertex Deletion, the Co-Path/Cycle Packing problem, which asks to delete as few vertices as possible such that the graph of the remaining (residual) vertices is composed of disjoint paths and simple cycles. The problem falls into the well-known class of 'node-deletion problems with hereditary properties', is hence NP-complete and unlikely to admit a polynomial time approximation algorithm with approximation factor smaller than 2. In the framework of parameterized complexity, we present a kernelization algorithm that produces a kernel with at most $37k$ vertices, improving on the super-linear kernel of Fellows *et al.*'s general theorem for Bounded-Degree Vertex Deletion. Using this kernel, and the method of bounded search trees, we devise an FPT algorithm that runs in time $O^*(3.24^k)$. On the negative side, we show that the problem is APX-hard and unlikely to have a kernel smaller than $2k$ by a reduction from Vertex Cover.

1 Introduction

In computational biology, a fundamental problem is to build up phylogenetic networks (trees) for various species, some of which are possibly extinct. A basic problem along this line is to construct ancestral genomes from the genomes of currently living species. Recently, Chauve and Tannier proposed the use of PQ-trees, where each leave represents a gene marker, to represent possible ancestral genomes [3]. This approach raises a natural question: given two PQ-trees over the same set of markers, how do we compare their similarity? For example, do they generate the same sequence?

In [9], the above problem is shown to be NP-complete. A natural extension of the problem is to delete the minimum number of common markers so that the two resulting PQ-trees can generate the same sequence. Modeling the markers in the PQ-trees as a hyper-graph, this is exactly the problem of deleting the minimum number of markers so that the resulting graph is composed of a set of paths; and, when circular genomes are allowed (as in [13]), a set of paths and cycles. We call this the Co-Path/Cycle Packing problem.

This problem belongs to the family of problems concerned with deleting a minimum number of vertices to obtain a graph belonging to a hereditary class of graphs (the well-known Vertex Cover problem is another example) [11]. A general polynomial time 2-approximation algorithm is known for this family of problems [5], and therefore Co-Path/Cycle Packing can be approximated within a factor of 2 in polynomial time. By a reduction from Vertex Cover, we show that an α-approximation algorithm for Co-Path/Cycle Packing yields an α-approximation algorithm for Vertex Cover. By the recent results of Khot and Regev [10], it follows that Co-Path/Cycle Packing does not admit a polynomial time approximation algorithm with performance factor $2 - \epsilon$, unless the Unique Games Conjecture fails.

In the parameterized framework, Fellows *et al.* recently considered the Bounded-Degree Vertex Deletion (d-BDD) problem, that of deleting the minimum number k of vertices (the parameter) so that the resulting graph has maximum degree d [6]. When $d = 0$, this is the Vertex Cover problem; 2-BDD is exactly our Co-Path/Cycle Packing problem. Fellows *et al.* presented a generalized Nemhauser-Trotter Theorem that implies that the d-BDD problem admits a kernel with a linear number of vertices for $d \leq 1$ and that d-BDD admits an $O(k^{1+\epsilon})$ kernel for $d \geq 2$ [6]. (We comment that in the conference version of their paper, the claimed linear kernel bound for all d was not correct; the result described is in the journal version.)

Here we present a $37k$ kernel for Co-Path/Cycle Packing. This is the first vertex linear kernel 2-BDD problem. Our approach here is similar to that of Fellows *et al.* Roughly speaking, using the fact that a path/cycle packing cannot contain any 3-star, we compute a *proper* maximal 3-star packing in the input graph and use them to compute a *triple crown decomposition*, and subsequently obtain the linear kernel. Using the $37k$ vertex kernel, we describe an FPT algorithm that runs in time $O^*(3.24^k)$, based on a bounded search tree approach.

This paper is organized as follows. In Section 2, we give some definitions. In Section 3, we show the kernelization algorithm and prove the lower bound. In Section 4, we present the bounded search tree algorithm. In Section 5, we conclude the paper with several open questions.

2 Preliminaries

We begin this section with some basic definitions and notations of graph theory.

Given an undirected graph $G = (V, E)$, for a vertex subset $S \subseteq V$, let $G[S]$ be the subgraph induced by S and $G-S = G[V/S]$. Similarly, for an edge set $P \subseteq E$, $G-P = (V, E-P)$. The *neighborhood* of a vertex v and a vertex set S is denoted as $N(v) = \{u \in V|\{u,v\} \in E\}$ and $N(S) = \{u \in V - S|v \in S, \{u,v\} \in E\}$, respectively. $N[S] = N(S) \cup S$. $d(v) = |N(v)|$ is the *degree* of a vertex v. The graph $K_{1,s} = (\{u, v_1, \dots, v_s\}, \{(u, v_1), \dots, (u, v_s)\})$ is called an s-star. u is the *center* and v_i's are the leaves. An s-star packing is a collection of vertex-disjoint s-stars. A *path/cycle packing* for an undirected graph $G = (V, E)$ is a vertex set whose induced subgraph is composed of disjoint paths and simple cycles. (An isolated vertex is also considered as a path, of length zero.) For convenience, we also call the corresponding subgraph *path/cycle set.* A *co- path/cycle packing* for an undirected graph $G = (V, E)$ is a vertex set whose deletion results in a graph which is a path/cycle set. We now make the following formal definition for the problem involved in this paper.

Minimum Co-Path/Cycle Packing:
Input: An undirected graph $G = (V, E)$, integer k.
Question: Does there exist a vertex set $S \subseteq V$ of size at most k such that the induced subgraph $G - S$ is a path/cycle set?

We now present some definitions regarding FPT algorithms. Given a parameterized problem instance (I,k), an FPT (Fixed-Parameter Tractable) algorithm solves the problem in $O(f(k)n^c)$ time (often simplified as $O^*(f(k))$), where f is a function of k only, n is the input size and c is some fixed constant (i.e., not depending on k or n). A useful technique in parameterized algorithmics is to provide polynomial time executable data-reduction rules that lead to a *problem kernel.* A data-reduction rule replaces (I,k) by an instance (I',k') in polynomial time such that: (1) $|I'| \leq |I|$, $k' \leq k$, and (2) (I,k) is a Yes-instance if and only if (I',k') is a Yes-instance. A set of polynomial-time data-reduction rules for a problem are applied to an instance of the problem to achieve a *reduced* instance termed the *kernel.* A parameterized problem is FPT if and only if there is a polynomial time algorithm applying data-reduction rules that reduce any instance of the problem to a kernelized instance of size $g(k)$. More about parameterized complexity can be found in the monographs [4,7,12].

3 A Linear Kernel

In this section, we describe a polynomial time data-reduction rule and show that it yields a kernel having at most $37k$ vertices for the Co-Path/Cycle Packing problem, improving the super-linear kernel for 2-BDD problem by Fellows *et al.* [6].

As a common trick in data reduction, we remove vertices of high degree and remove useless vertices connecting vertices of degree at most two. The corresponding rules are summarized in the following lemmas.

Lemma 1. *Let G be a graph such that there exists $v \in V$ with $d(v) > k+2$, then G has a k-co-path/cycle packing iff $G - v$ has a $(k-1)$-co-path/cycle packing.*

Lemma 2. *Let G be a graph such that there exists an edge $e = (u, v)$ and $d(u) = d(v) \leq 2$, then G has a minimum co-path/cycle packing that does not contain u and v.*

Proof. W.l.o.g., we only consider the case when u is in some minimum co-path/cycle packing. Assume that W is a minimum co-path/cycle packing with $u \in W$, then $G - W$ contains a path ending at v and a path containing x (the other neighbor of u). We can construct another co-path/cycle packing W' from W by replacing u with x. Symmetrically, we just add e at the end of some path in $G - W$. Therefore, W' is a minimum co-path/cycle packing. □

Lemma 3. *Let G be a graph such that there exists a path $P=\langle v_1, v_2, \ldots, v_t\rangle$ where $t \geq 3$, and $d(v_i) \leq 2$ for all $1 \leq i \leq t$. Let Z denote the vertex set in the middle of the path, i.e., $Z = \{v_2, v_3, ..., v_{t-1}\}$, and let G' be the graph constructed by adding a new edge $e = (v_1, v_t)$ to $G-M$, then G has a k-co-path/cycle packing iff G' has a k-co-path/cycle packing.*

Proof. The 'only if' part is trivially true, so we will focus on the 'if' part. Following Lemma 2, there is a k-co-path/cycle packing W in G' which does not contain v_1 and v_t. So we will not create any vertex with degree more than two in $G - W$ by inserting the vertices in Z between v_1 and v_t. □

Lemma 3 basically implies that we can contract a vertex of degree at most two to either one of its neighbors, as long as its neighbors also have degrees at most two. From now on, we assume that an input graph is already preprocessed by Lemma 3.

We next review a famous structure in parameterized complexity, the *crown decomposition*, which was used to obtain a small kernel for Vertex Cover [1,2].

Definition 1. A *crown decomposition* (H,C,R) in a graph $G = (V, E)$ is a partition of V into three sets H, C and R which have the following properties:

(1) H (the head) is a separator in G such that there are no edges between the vertices in C and the vertices in R.

(2) $C = C_u \cup C_m$ (the crown) is an independent set in G.

(3) $|C_m| = |H|$, and there is a perfect matching between C_m and H.

We modify and generalize the crown decomposition to handle our particular problem. The variation is called *triple crown decomposition*, where each vertex in H has three vertices in C matched to it. We elaborate the details as follows.

Definition 2. A *triple crown decomposition* (H,C,L,R) in a graph $G = (V, E)$ is a partition of the vertices in V into four sets H,C,L and R which have the following properties:

(1) H (the head) is a separator in G such that there are no edges between the vertices in C and the vertices in R.

(2) $L = N(C)/H$ (the neighbor), $G[L \cup C]$ is a path/cycle set, and $|N(l) \cap (R \cup C)| \leq 2$ for all $l \in L$.

(3) $C = C_u \cup C_{m_1} \cup C_{m_2} \cup C_{m_3}$ (the crown). $|C_{m_i}| = |H|$, and there is a perfect matching between every C_{m_i} and H, for all $i \in \{1, 2, 3\}$.

Based on the triple crown decomposition, we describe the critical data reduction rule in this paper through the following lemma.

Lemma 4. *A graph $G = (V, E)$ which admits a triple crown decomposition (H, C, L, R) has a k-co-path/cycle packing iff $G - H$ has a $(k - |H|)$-co-path/cycle packing.*

Proof. The 'if' part is easy to prove because any co-path/cycle packing of $G - H$ together with H is certainly a solution for G.

We now prove the other direction. Let G have a co-path/cycle packing of size k. First, we can see that any co-path/cycle packing for $G[H \cup C]$ contains at least $|H|$ vertices. Since (H, C, L, R) is a *triple crown decomposition*, there are $|H|$ vertex-disjoint 3-stars in $G[H \cup C]$, at least one vertex of every star should be deleted in order to obtain a path/cycle set. Moreover, every vertex belonging to $C \cup L$ has degree at most two and every vertex belonging to C has no neighbor outside of L in $G - H$. There is no minimum co-path/cycle packing W for $G - H$ such that $v \in W \cap C$, otherwise v connects at most two paths in $G - H - W$, and $W - v$ is hence a smaller co-path/cycle packing. So the size of the minimum co-path/cycle packing for G is at least $|W| + |H|$, which means $|W| \leq k - |H|$. Hence the 'only if' part holds. □

Now, our main idea of the kernelization algorithm is to search for *triple crown decomposition* in the graph iteratively. When we cannot find that structure at all, we can conclude that the graph has bounded size. At the beginning, the algorithm computes a maximal 3-star packing in a greedy fashion. Note that the maximal 3-star packing thus found is a co-path/cycle packing, i.e., if we delete all these 3-stars the resulting graph has no vertex of degree greater than two. Then we refine the maximal 3-star packing such that the following lemma is satisfied. After that, the algorithm tries to search a *triple crown decomposition* with H being a subset of the star centers and C belonging to the path/cycle set.

We first summarize the method to obtain/refine a proper maximal 3-star packing in the following lemma.

Lemma 5. *Given a graph $G = (V, E)$, we can produce a maximal 3-star packing W such that every 3-star $P \in W$ falls into one of the three cases,*

1. *if the center u of P has at least 4 neighbors in $G - W$, then every leaf of P has at most two neighbors in $G - W$, all of them are of degree one in $G - W$.*
2. *if the center u of P has one to three neighbors in $G - W$, then any 3-star Q composed of one leaf v of P and three other vertices in $G - W$ contains all neighbors of u in $G - W$.*
3. *if the center u has no neighbor in $G - W$, then each leaf of P has at most two distinct neighbors, for a total of at most 6, in $G - W$.*

Proof. We prove the three cases respectively.

1. Suppose on the contrary that there exists a leaf x of P which is incident to some vertex y of degree-2 in $G - W$. Then we have a 3-star P' centered at y, with x being one of its leaves. Therefore, we can obtain one extra 3-star besides P': just modify P by replacing x with v as a new leaf. (v must exist due to that u has at least 4 neighbors in $G - W$.) This contradicts the optimality of W.
2. Otherwise, we can swap v with some neighbor x of u in $G - W$, with $x \notin Q$. We thus obtain one more 3-star. Again, this contradicts the optimality of W.
3. Let the leaves of P be $\{v_1, v_2, v_3\}$. Assume on the contrary that there exists a leaf v_1 of P which has more than three neighbors different from the neighbors of v_2 and v_3 in $G - W$. We can replace P with a new 3-star composed of v and its three neighbors. □

The algorithm *Proper Maximal 3-Star Packing* follows directly from Lemma 5. In the following box, we describe the corresponding algorithm with pseudo-code.

```
Algorithm Proper Maximal 3-Star Packing
Input: Graph G=(V,E)
Output: A proper maximal 3-star packing W
 1    Compute a maximal 3-star packing $W$ greedily.
 2    For every $P = \langle u, \{x, y, z\}\rangle \in W$, check the following properties iteratively
until $W$ fulfills Lemma 5
   2.1   if there are two or three disjoint 3-stars $Q$,$R$ (or $S$)
each contains only one leaf of $P$,
           then $W = W - P + Q + R$ $(+S)$.
   2.2   if there is a 3-star $Q$ which contains only one leaf $x$ and
$u$ has a neighbor $v$ distinct from $Q$ in $G - W$,
           then replace $x$ with $v$ in $P$, $W = W + Q$.
   2.2   if there is a 3-star $Q$ which contains only one leaf $x$, and
$y$ and $z$ have no neighbor in $Q$,
           then $W = W - P + Q$.
```

Since the maximum 3-star packing in a graph is bounded by $O(n)$, any violation of the conditions (1) and (2) leads to a larger 3-star packing and each 3-star can be computed in $O(n)$ time. So we can find such a proper maximal 3-star packing fulfilling Lemma 5 in $O(n^2)$ time.

In order to describe the main algorithm concisely, we make use of some notations for some vertex sets in the graph. Let W be the maximal 3-star packing we obtain (after running Lemma 5). L_W and C_W denotes all the leaves and centers in W respectively. Then, $D_W^L = N(L_W) \cap (G - W)$, $D_W^C = (N(C_W) - D_W^L) \cap (G - W)$, $F(L_W) = N[D_W^L] \cap (G - W)$, and for every $v \in C_W$, $F(v) = N[N(v)] \cap (G - W - F(L_W))$. $S \subseteq C_W$, $F(S) = \bigcup_{v \in S} F(v)$.

The next procedure computes a *triple crown decomposition*, if it exists.

```
Algorithm Triple Crown Decomposition
Input: Graph G=(V,E), proper maximal 3-star packing W
Output: Triple crown decomposition (H,C,L,R)
 1   $X$ is the case-1 centers in $C_W$, $Y = G - W - F(L_W) - F(C_W - X)$.
 2   Construct a bipartite graph $J = (X, Y, T)$,
where $T = \{(u, v) \in E | u \in X, v \in Y\}$.
 3   Replicate every vertex $u \in X$ twice such that $u$ and its two copies have
the same neighbor in $J$.
     Let the new graph be $J'$ and let $u^*$ denote the original vertex in $X$
for both of $u$'s copies.
 4   Compute the maximum matching $M'$ in $J'$ and for every edge
$e = (u, v) \in M'$, construct $(u^*, v) \in M \subseteq T$          .
 5   Repeat the following two steps with $C_0 = Y - M$ until $C_i = C_{i+1}$.
     5.1      $H_i = N_J(C_i)$;
     5.2      $C_{i+1} = N_M(H_i) \cup C_i$;
 6   If $H_i = X$, then we get a triple crown decomposition $(H, C, L, R)$
     where $H = H_i$, $C = C_i$, $L = N_G(C) - H$, $R = G - H - C - L$.
 7   Else $F_X = X - H_i$, $F_Y = F(X - H_i)$, $X = X - F_X$, $Y = Y - F_Y$.
 8   If $H_i \neq \phi$, goto step 2; else exit.
```

We have the following lemmas regarding the above algorithm.

Lemma 6. *$(H \cup C) \cap M$ is a 3-star packing such that all centers are in H and all leaves are in C.*

Proof. If there exists a vertex $v \in (H \cap M)$ which has fewer than three neighbors in $(C \cap M)$. From the way we get v, there is an M-augmenting path from some vertex not in M to v. (An M-augmenting path is a path where the edges in M and edges not in M alternate.) Then M cannot be maximum since both the start and end edges of the M-augmenting path are not in M. □

Lemma 7. *If v connects a vertex $h \notin H$, then v is matched to some vertex not in H.*

Proof. If v is not in M, then $v \in C$ and $N_J(v) \subseteq H$, which contradicts the condition in the lemma. Therefore $v \in M$. If v is matched to some vertex in H, then $v \in C$ and $h \in H$, also contradicts the condition in the lemma. □

Note that every vertex in Y either belongs to C or belongs to $N(F_X)$.

Lemma 8. *If the procedure does not find a triple crown decomposition, then $|F_Y| \leq 9|F_X|$.*

Proof. From Lemma 7, every vertex in F_Y is either matched to some vertex in F_X or is a neighbor of a matched vertex in F_Y. Since every vertex in F_X has at most three neighbors in M, $G - W$ is a path/cycle set. Therefore, $|F_Y| \leq 3 \cdot (2 + 1)|F_X| = 9|F_X|$. □

If the procedure cannot find a triple crown decomposition, which means $F_X = X$, we then refer T to the vertices in $Y - F_Y$. The next lemma shows properties of vertices in T.

Lemma 9. *For every $u \in T$, u fulfills the following two properties:*

1. *u has no neighbor in X.*
2. *There exists a vertex $v \in N(u)$, v is included in $F_Y \cup F(L_W) \cup F(C_W - X)$ and v is connected to another vertex $w \in N(W)$.*

Proof. From the procedure, we can see that any vertex in $Y = G - W - F(L_W) - F(C_W - X)$ which has some neighbor in X or has some neighbor in $N(F_X)$ is included in F_Y. Therefore, if $u \notin F_Y$, both of u's neighbors either belong to T or belong to $N(N(W))$. From Lemma 3, there are at most two consecutive vertices whose degrees are at most two. So at least one of u's neighbors belongs to $N(N(W))$. □

It is easy to verify that any single vertex or two consecutive vertices in T lie between vertices in $N(N(W))$. The following lemma bounds the cardinality of T.

Lemma 10. $|T| \leq 12|C_W|$.

Proof. Assume that there are r_i 3-stars of case-i where $i \in \{1, 2, 3\}$, we define an *interval* as a maximal path whose endpoints are in $N(N(W))$ and other intermediate vertices are not in $N(N(W))$. The proof of this lemma is based on computing the number of intervals between the vertices in $N(N(W))$. From Lemma 8, there are at most $3r_1$ distinct vertices in $N(F_X)$. From Lemma 5, every leaf of a case-1 3-star connects at most two vertices in $N(W)$, both of which are of degree one in $G - W$. Also, there are at most $3r_2$ distinct vertices which are the neighbors of the centers of case-2 3-stars. We conclude that after deleting neighbors of its corresponding center, every leaf of a case-2 3-star connects at most two vertices in $N(W)$, both of which are of degree one. Besides, there are at most $6r_3$ distinct vertices who are neighbors of the leaves of case-3 3-stars. For any vertex u in $N(W)$, when we contract the vertices in $(N[N(u)])$ in $G - W$ to a single vertex, there are at most $6r_1 + 6r_2 + 6r_3 - m/2$ intervals between those resulting vertices, where m is the number of tails (endpoints) of paths that are not in $N(W)$. Since there are at most two consecutive vertices in T lying in each interval and at most one vertex lies in each of the m tails, $|T| \leq 2 * (6r_1 + 6r_2 + 6r_3 - m/2) + m = 12|C_W|$. □

Theorem 1. *The Co-Path/Cycle Packing problem has a linear kernel of size $37k$.*

Proof. First of all, note that $V = W \cup F(L_W) \cup F(C_W - X) \cup F_Y \cup T$. Assume that there are r_i 3-stars of case-i where $i \in \{1, 2, 3\}$, from Lemma 5, every case-3 3-star corresponds to at most 18 vertices in $F(L_W) \cup F(C_W - X)$, every case-2 3-star corresponds to at most 21 vertices in $F(L_W) \cup F(C_W - X)$ and every leaf of case-1 3-star corresponds to at most 12 vertices in $F(L_W)$. Therefore, $|F(L_W) \cup F(C_W - X)| \leq (12r_1 + 21r_2 + 18r_3)$. From Lemma 8, every center of case-1 3-star corresponds to at most 9 vertices in F_Y. Then, $|F_Y| \leq 9r_1$. Following Lemma 10, $|T| \leq 12|C_W| = 12r_1 + 12r_2 + 12r_3$. In addition, $|W| = 4(r_1 + r_2 + r_3)$. Consequently, $|V| = |W| + |F(L_W) \cup F(C_W - X)| + |F_Y| + |T| \leq (37r_1 + 37r_2 + 34r_3) \leq 37(r_1 + r_2 + r_3) \leq 37k$, since we have $r_1 + r_2 + r_3 \leq k$. □

For completeness, we show below that the Co-Path/Cycle Packing problem is at least as hard as Vertex Cover, in terms of designing both approximation and FPT algorithms.

Theorem 2. *The Co-Path/Cycle Packing problem is APX-hard and cannot have a linear kernel of size smaller than* $2k$.

Proof. We prove the theorem by a simple reduction from *Vertex Cover*, which is APX-hard. Moreover, there is a famous conjecture that Vertex Cover cannot be approximated within a factor smaller than 2 [10]; consequently, no kernel smaller than $2k$ exists (unless the conjecture is disproved). The detailed reduction is as follows. Given an instance of *Vertex Cover*, $G = (V, E)$, we construct an instance of Co-Path/Cycle Packing $G^{'} = (V^{'}, E^{'})$. Let V_1 and V_2 be two vertex set each has $|V|$ isolated vertices. For each $v_i \in V$, add an edge between v_i and an isolated vertices in V_1 and V_2 respectively, so the set of new edges is defined as $E^* = \{(v_i, x_i)|v_i \in V, x_i \in V_1\} \cup \{(v_i, y_i)|v_i \in V, y_i \in V_2\}$. Then, $V^{'} = V \cup V_1 \cup V_2$, $E^{'} = E \cup E^*$. It is easily seen that any Vertex Cover in G is also a co-path/cycle packing in $G^{'}$, and any co-path/cycle packing corresponds to a Vertex Cover with at most the same size (since we can replace the vertices in the solution from $V_1 \cup V_2$ with their neighbors in V). Then the two instances could have the same optimal solution. Hence, the Co-Path/Cycle Packing problem cannot achieve a smaller approximation factor than that for Vertex Cover, and is unlikely to have a kernel of size smaller than $2k$, unless the conjecture by Khot and Regev is disproved. □

4 The FPT Algorithm

In this section, we elaborate the FPT algorithm which runs in $O^*(3.24^k)$ time. The key idea of the algorithm is bounded search tree. (For a survey, please refer to the paper by Fernau and Raible [8].) We implement it by some involved analysis. The following lemmas are critical while applying the bounded search tree method.

Lemma 11. *Given a graph $G = (V, E)$, if there exists a vertex v with $d(v) \geq 3$, then any minimum co-path/cycle packing for G either contains v or contains all but at most two of its neighbors.*

Lemma 12. *Given a graph $G = (V, E)$, if there exists an edge (u, v) with $d(u) = d(v) = 3$ and $N(u) \cap N(v) = \phi$, then any minimum co-path/cycle packing for G either contains one of u and v or contains at least one neighbor of u and one neighbor of v.*

Lemma 13. *Given a graph $G = (V, E)$, if there exists three edges (u, v), (u, w) and (v, w) with $d(u) = d(v) = 3$ and $N(\{u, v, w\}) = \{x, y, z\}$, then any a minimum co-path/cycle packing either contains one of u, v, w or contains all of x, y, z.*

The above three lemmas are easy to check, since otherwise there will be some vertex with degree greater than two left.

Lemma 14. *Given a graph $G = (V, E)$, if there exists an edge (u, v) with $d(u) = d(v) = 3$, $x \in (N(u) \cap N(v))$, and $d(x) = 2$, then there exists a minimum co-path/cycle packing which does not contain x.*

Proof. When a minimum co-path/cycle packing W contains x, $W - x + u$ remains a minimum co-path/cycle packing. □

Lemma 15. *Given a graph $G = (V, E)$, if there exists an edge (u, v) with $d(u) = d(v) = 3$ and $N(u) \cap N(v) = \{x, y\}$, then there exists a minimum co-path/cycle packing which does not contain v.*

Proof. When a minimum co-path/cycle packing W contains v, $W - v + u$ remains a minimum co-path/cycle packing. □

Lemma 16. *Given a graph $G = (V, E)$, if there exists a 3-star with the center being of degree three and every leaf of degree at most two in G, then there is a minimum co-path/cycle packing for G which only contains one leaf in the star.*

Proof. Any minimum co-path/cycle packing should contain at least one vertex from any 3-star. From Lemma 2 and Lemma 3, the deletion of a leaf results in the reservation of the other three vertices (in an optimal solution). So we just prove the existence of a minimum co-path/cycle packing which does not contain the center. Suppose on the contrary that the center is in a minimum co-path/cycle packing W, then we can modify W by replacing the center with any leaf. Obviously, only two paths are connected together in $G - W$ as a result of the replacement. Hence, the lemma holds. □

The detailed algorithm based on the above lemmas is presented at the end of the paper.

Theorem 3. *Algorithm $CPCP(G, k)$ solves the Co-Path/Cycle Packing problem correctly in $O^*(3.24^k)$ time.*

Proof. Step 1 deals with the boundary cases for the k Co-Path/Cycle Packing problem: if $k < 0$, then no co-path/cycle packing of size at most k can be found, thus the algorithm returns 'NO'. If $k \geq 0$ and G consists of only paths and cycles, then there is no need to remove any vertex from G, thus ϕ can be returned safely.

Step 2 considers the case when there is a vertex v of degree greater than 3. Following Lemma 11, it returns a k co-path/cycle packing correctly.

Step 3 deals with the case when there is an edge (u, v) such that u and v are of degree-3. From Lemma 12, Step 3.1 returns a k co-path/cycle packing correctly if it exists. From Lemma 14, Step 3.2.1 returns a k co-path/cycle packing correctly if it exists. From Lemma 13, Step 3.2.2 returns a k co-path/cycle packing correctly if it exists. From Lemma 15, Step 3.3 returns a k co-path/cycle packing correctly if it exists.

After Step 1, we have that $k \geq 0$ and G cannot consist of paths and cycles only, which implies that there is at least a vertex of degree greater than 2. After Step 2, all the vertices have degree less than or equal to 3 in G. After Step

3, no two degree-3 vertices are adjacent. It follows that all three neighbors of any degree-3 vertex v must be of degree less than 2. From Lemma 16, Step 4 returns a k co-path/cycle packing correctly if it exists. After that, every vertex has degree at most two. This completes the correctness proof of the algorithm.

Now we analyze the running time. As a convention, we take the notation $O^*(g(k))$ to refer to $O(g(k)n^c)$, where c is a constant independent of k and n. Step 1 takes time of $O(|G|)$. Let $f(k)$ be the time complexity for the k Co-Path/Cycle Packing problem. Step 2 has recurrence

$$f(k) = f(k-1) + \binom{i}{2} f(k-(i-2)), \quad \text{where } i \geq 4.$$

Step 3 has recurrence

$$f(k) \leq \begin{cases} 2f(k-1) + 4f(k-2), \\ 2f(k-1) + 2f(k-2), \\ 3f(k-1) + f(k-3), \\ 3f(k-1). \end{cases}$$

Step 4 has recurrence

$$f(k) = 3f(k-1).$$

We can verify that $f(k) \leq 3.24^k$, which comes from $f(k) = 2f(k-1) + 4f(k-2)$ at Step 3. Actually, we can show that $f(k) \leq 3^k$ by induction for the recurrence at Step 2. This concludes the proof. □

5 Concluding Remarks

In this paper, we present a $37k$ kernel for the minimum Co-Path/Cycle Packing problem. With the bounded search technique, we obtain an FPT algorithm which runs in $O^*(3.24^k)$ time. This problem is a special case of the Bounded-Degree Vertex Deletion (BDD) problem (when $d = 2$). So our result is the first linear kernel for BDD when $d = 2$. Previously, a linear kernel is only known for BDD when $d = 0$ (Vertex Cover) or $d = 1$ [6]. An interesting question is whether the bounds can be further improved. Another interesting question is to disallow cycles when some vertices are deleted (in the ancestral genome reconstruction problem, that means that no circular genome is allowed). In this latter case, we only have an $O(k^2)$ kernel.

Acknowledgments

This research is partially supported by NSF Career Award 084537, NSF grant DMS-0918034, and by NSF of China under grant 60928006. LW is fully supported by a grant from the Research Grants Council of the Hong Kong SAR, China [Project No. CityU 121207].

```
Algorithm Co-Path/Cycle Packing (CPCP)
Input: a graph $G$ and an integer $k$
Output: a co-path/cycle packing $S$ for $G$ such that $|S| \le k$, or report 'NO'
 1   if $k < 0$, return 'NO'
     if $G$ consists of only paths and simple cycles, return $\phi$
 2   pick a vertex $v$ of degree greater than 3
   2.1   $S \leftarrow CPCP(G - v, k - 1)$
         if $S$ is not 'NO', return $S + v$
   2.2   for every two neighbors $w, z$ of $v$, $G'$ is the graph after removing all
neighbors of $v$ except $w, z$ from $G$.
         $S \leftarrow CPCP(G', k - (i - 2))$ ($i \ge 4$ is the number of neighbors of $v$).
         if $S$ is not 'NO', return $S+$ all neighbors of $v$ other than $w, z$
 3   pick an edge $e = (u, v)$ such that both $u$ and $v$ are of degree 3.
   3.1   $u$ and $v$ have no common neighbor. let $u_1, u_2$ ($v_1, v_2$) be the other
two neighbors of $u$ ($v$).
         3.1.1   $S \leftarrow CPCP(G - u, k - 1)$   if $S$ is not 'NO', return $S + u$
         3.1.2   $S \leftarrow CPCP(G - v, k - 1)$   if $S$ is not 'NO', return $S + v$
         3.1.3   $S \leftarrow CPCP(G - u_i - v_j, k - 2)$, for each $u_i, v_j$ ($i, j \in \{1, 2\}$),
if $S$ is not 'NO', return $S + u_i - v_j$
   3.2   $u$ and $v$ have only one common neighbor $w$. let $x$ ($y$) be the other
neighbors of $u$ ($v$).
         3.2.1.   if $d(w) = 2$
            3.2.1.1   $S \leftarrow CPCP(G - u, k - 1)$   if $S$ is not 'NO', return $S + u$
            3.2.1.2   $S \leftarrow CPCP(G - v, k - 1)$   if $S$ is not 'NO', return $S + v$
            3.2.1.3   $S \leftarrow CPCP(G - x - y, k - 2)$   if $S$ is not 'NO',
return $S + x + y$
         3.2.2.   if $d(w) = 3$, let $z$ be the other neighbors of $w$ and $z \ne x, z \ne y$
            3.2.2.1   $S \leftarrow CPCP(G - u, k - 1)$   if $S$ is not 'NO', return $S + u$
            3.2.2.2   $S \leftarrow CPCP(G - v, k - 1)$   if $S$ is not 'NO', return $S + v$
            3.2.2.3   $S \leftarrow CPCP(G - w, k - 1)$   if $S$ is not 'NO', return $S + w$
            3.2.2.4   $S \leftarrow CPCP(G - x - y - z, k - 3)$   if $S$ is not 'NO',
return $S + x + y + z$
         3.2.3   if $d(w) = 3$, let $z$ be the other neighbors of $w$ and $z \in \{x, y\}$
            3.2.3.1   if $z = x$, goto step3 with $e = (u, w)$.
            3.2.3.2   if $z = y$, goto step3 with $e = (v, w)$.
   3.3   $u$ and $v$ have two neighbors $w_1, w_2$in common, let $x$ ($y$) be the other
neighbor of $w_1$ ($w_2$)
         3.3.1   $S \leftarrow CPCP(G - u, k - 1)$   if $S$ is not 'NO', return $S + u$
         3.3.2   $S \leftarrow CPCP(G - w_1, k - 1)$   if $S$ is not 'NO', return $S + w_1$
         3.3.3   $S \leftarrow CPCP(G - w_2, k - 1)$   if $S$ is not 'NO', return $S + w_2$
 4   pick a degree-3 vertex $v$ with neighbors $v_1, v_2, v_3$, for each $v_i$ ($i \in \{1, 2, 3\}$)
       $S \leftarrow CPCP(G - v_i, k - 1)$, if $S$ is not 'NO', return $S + v_i$.
 5   return 'NO'
```

References

1. Abu-Khzam, F.N., Fellows, M.R., Langston, M.A., Suters, W.H.: Crown structures for vertex cover kernelization. Theory Comput. Syst. 41(3), 411–430 (2007)
2. Chen, J., Kanj, I.A., Jia, W.: Vertex cover: Further observations and further improvements. J. Algorithms 41(2), 280–301 (2001)
3. Chauve, C., Tannier, E.: A methodological framework for the reconstruction of contiguous regions of ancestral genomes and its application to mammalian genome. PLoS Comput. Biol. 4, e1000234 (2008)
4. Downey, R., Fellows, M.: Parameterized Complexity. Springer, Heidelberg (1999)
5. Fujito, T.: Approximating node-deletion problems for matroidal properties. J. Algorithms 31, 211–227 (1999)
6. Fellows, M., Guo, J., Moser, H., Niedermeier, R.: A generalization of Nemhausser and Trotter's local optimization theorem. In: Proc. STACS 2009, pp. 409–420 (2009)
7. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Heidelberg (2006)
8. Fernau, H., Raible, D.: Search trees: an essay. In: Proc. TAMC 2009, pp. 59–70 (2009)
9. Jiang, H., Chauve, C., Zhu, B.: Breakpoint distance and PQ-trees. In: Javed, A. (ed.) CPM 2010. LNCS, vol. 6129, pp. 112–124. Springer, Heidelberg (2010)
10. Khot, S., Regev, O.: Vertex cover might be hard to approximate to within $2 - \epsilon$. J. Comput. System Sci. 74, 335–349 (2008)
11. Lewis, J., Yannakakis, M.: The node-deletion problem for hereditary properties is NP-complete. J. Comput. System Sci. 20, 425–440 (1980)
12. Niedermeier, R.: Invitation to fixed-parameter algorithms. Oxford University Press, Oxford (2006)
13. Tannier, E., Zheng, C., Sankoff, D.: Multichromosomal median and halving problems under different genomic distances. BMC Bioinformatics 10, 120 (2009)