Parameterized Complexity of (A, ℓ)-Path Packing

Rémy Belmonte[1], Tesshu Hanaka[2], Masaaki Kanzaki[3], Masashi Kiyomi[4], Yasuaki Kobayashi[5], Yusuke Kobayashi[5], Michael Lampis[6], Hirotaka Ono[7], and Yota Otachi[7](✉)

[1] The University of Electro-Communications, Chofu, Tokyo, Japan
remybelmonte@gmail.com
[2] Chuo University, Bunkyo-ku, Tokyo, Japan
hanaka.91t@g.chuo-u.ac.jp
[3] Kumamoto University, Kumamoto 860-8555, Japan
c5744@st.cs.kumamoto-u.ac.jp
[4] Yokohama City University, Yokohama, Japan
masashi@yokohama-cu.ac.jp
[5] Kyoto University, Kyoto, Japan
kobayashi@iip.ist.i.kyoto-u.ac.jp, yusuke@kurims.kyoto-u.ac.jp
[6] Université Paris-Dauphine, PSL University, CNRS, LAMSADE,
75016 Paris, France
michail.lampis@lamsade.dauphine.fr
[7] Nagoya University, Nagoya 464-8601, Japan
{ono,otachi}@nagoya-u.jp

Abstract. Given a graph $G = (V, E)$, $A \subseteq V$, and integers k and ℓ, the (A, ℓ)-Path Packing problem asks to find k vertex-disjoint paths of length ℓ that have endpoints in A and internal points in $V \setminus A$. We study the parameterized complexity of this problem with parameters $|A|$, ℓ, k, treewidth, pathwidth, and their combinations. We present sharp complexity contrasts with respect to these parameters. Among other results, we show that the problem is polynomial-time solvable when $\ell \leq 3$, while it is NP-complete for constant $\ell \geq 4$. We also show that the problem is W[1]-hard parameterized by pathwidth $+ |A|$, while it is fixed-parameter tractable parameterized by treewidth $+ \ell$.

Keywords: A-path packing · Fixed-parameter tractability · Treewidth

1 Introduction

Let $G = (V, E)$ be a graph and $A \subseteq V$. A path P in G is an *A-path* if the first and the last vertices of P belong to A and all other vertices of P belong to $V \setminus A$. Given

Partially supported by PRC CNRS JSPS project PARAGA, by JSPS KAKENHI Grant Numbers JP16K16010, JP17H01698, JP17H01788, JP18H05291, JP18K11157, JP18K11168, JP18K11169, JP18H04091, JP19K21537, and by JST CREST JPMJCR 1401. The authors thank Tatsuya Gima for helpful discussions.

L. Gąsieniec et al. (Eds.): IWOCA 2020, LNCS 12126, pp. 43–55, 2020.
https://doi.org/10.1007/978-3-030-48966-3_4

G and A, A-Path Packing is the problem of finding the maximum number of vertex-disjoint A-paths in G. The A-Path Packing problem is well studied and even some generalized versions are known to be polynomial-time solvable (see e.g., [5,6,11,15,18,19]). Note that A-Path Packing is a generalization of Maximum Matching since they are equivalent when $A = V$.

In this paper, we study a variant of A-Path Packing that also generalizes Maximum Matching. An A-path of length ℓ is an *(A, ℓ)-path*, where the length of a path is the number of edges in the path. Now our problem is defined as follows:

(A, ℓ)-Path Packing (ALPP)
Input: A tuple (G, A, k, ℓ), where $G = (V, E)$ is a graph, $A \subseteq V$, and k and ℓ are positive integers.
Question: Does G contain k vertex-disjoint (A, ℓ)-paths?

To the best of our knowledge, this natural variant of A-Path Packing was not studied in the literature. Our main motivation of studying ALPP is to see theoretical differences from the original A-Path Packing, but practical motivations of having the length constraint may come from some physical restrictions or some fairness requirements. Note that if $\ell = 1$, then ALPP is equivalent to Maximum Matching. Another related problem is ℓ-Path Partition [16,20,21], which asks for vertex-disjoint paths of length ℓ (without specific endpoints).

In the rest of paper, we assume that $k \leq |A|/2$ in every instance as otherwise the instance is a trivial no-instance. The restricted version of the problem where the equality $k = |A|/2$ is forced is also of our interest as that version corresponds to a “full” packing of A-paths. We call this version Full (A, ℓ)-Path Packing (Full-ALPP, for short). In this paper, all our positive results showing tractability of some cases will be on the general ALPP, while all our negative (or hardness) results will be on the possibly easier Full-ALPP.

We assume that the reader is familiar with terminologies in the parameterized complexity theory. See the textbook by Cygan et al. [8] for standard definitions.

Our Results

In summary, we show that ALPP is intractable even on very restricted inputs, while it has some nontrivial cases that admit efficient algorithms. (See Fig. 1.)

We call $|A|$, k, and ℓ the *standard parameters* of ALPP as they naturally arise from the definition of the problem. We determine the complexity of ALPP with respect to all standard parameters and their combinations. We first observe that Full-ALPP is NP-complete for any constant $|A| \geq 2$ (Observation 3.1) and for any constant $\ell \geq 4$ (Observation 3.2), while it is polynomial-time solvable when $\ell \leq 3$ (Theorem 3.3). On the other hand, ALPP is fixed-parameter tractable when parameterized by $k + \ell$ and thus by $|A| + \ell$ as well (Theorem 3.5). We later strengthen Observation 3.2 by showing that NP-complete for every fixed $\ell \geq 4$ even on grid graphs (Theorem 5.1).

We then study structural parameters such as treewidth and pathwidth in combination with the standard parameters. We first observe that ALPP can be solved in time $n^{O(\mathsf{tw})}$ (Theorem 4.1), where n and tw are the number of vertices and the treewidth of the input graph, respectively. Furthermore, we show that ALPP parameterized by $\mathsf{tw} + \ell$ is fixed-parameter tractable (Theorem 4.2). We finally show that Full-ALPP parameterized by $\mathsf{pw} + |A|$ is W[1]-hard (Theorem 4.5), where pw is the pathwidth of the input graph.

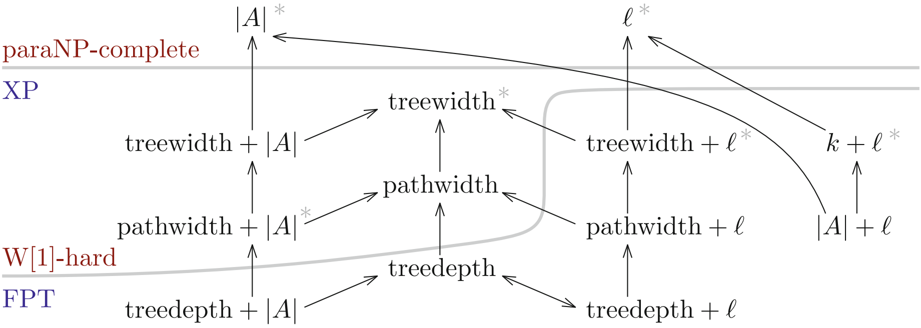

Fig. 1. Summary of the results. An arrow $\alpha \to \beta$ indicates that there is a function f such that $\alpha \geq f(\beta)$ for every instance of ALPP. Some possible arrows are omitted to keep the figure readable. The results on the parameters marked with $*$ are explicitly shown in this paper, and the other results follow by the hierarchy of the parameters. We have a bidirectional arrow treedepth ↔ treedepth + ℓ because the maximum length of a path in a graph is bounded by a function of treedepth [17, Section 6.2].

2 Preliminaries

A graph $G = (V, E)$ is a *grid graph* if V is a finite subset of $\mathbb{Z}^2$ and $E = \{\{(r, c), (r', c')\} \mid |r - r'| + |c - c'| = 1\}$. From the definition, all grid graphs are planar, bipartite, and of maximum degree at most 4. To understand the intractability of a graph problem, it is preferable to show hardness on a very restricted graph class. The class of grid graphs is one of such target classes.

A *tree decomposition* of a graph $G = (V, E)$ is a pair $(\{X_i \mid i \in I\}, T = (I, F))$, where $X_i \subseteq V$ for each i and T is a tree such that

- for each vertex $v \in V$, there is $i \in I$ with $v \in X_i$;
- for each edge $\{u, v\} \in E$, there is $i \in I$ with $u, v \in X_i$;
- for each vertex $v \in V$, the induced subgraph $T[\{i \mid v \in X_i\}]$ is connected.

The *width* of a tree decomposition $(\{X_i \mid i \in I\}, T)$ is $\max_{i \in I} |X_i| - 1$, and the *treewidth* of a graph G, denoted $\mathsf{tw}(G)$, is the minimum width over all tree decompositions of G.

The *pathwidth* of a graph G, denoted $\mathsf{pw}(G)$, is defined by restricting the trees T in tree decompositions to be paths. We call such decompositions *path decompositions*. It is easy to observe that pathwidth does not change significantly by subdividing some edges and attaching paths to some vertices.

Corollary 2.1 (★[1]). *Let $G = (V, E)$ be a graph without isolated vertices. If G' is a graph obtained from G by subdividing a set of edges $F \subseteq E$ an arbitrary number of times, and attaching a path of arbitrary length to each vertex in a set $U \subseteq V$, then $\mathsf{pw}(G') \leq \mathsf{pw}(G) + 2$.*

3 Standard Parameterizations of ALPP

In this section, we completely determine the complexity of ALPP with respect to the standard parameters $|A|$, k, ℓ, and their combinations. (Recall that $k \leq |A|/2$.) We first observe that using one of them as a parameter does not make the problem tractable. That is, we show that the problem remains NP-complete even if one of $|A|$, k, ℓ is a constant. We then show that the problem is tractable when $\ell \leq 3$ or when $k + \ell$ is the parameter.

3.1 Intractable Cases

The first observation is that Full-ALPP is NP-complete even if $|A| = 2$ (and thus $k = 1$). This can be shown by an easy reduction from HAMILTONIAN CYCLE [12]. This observation is easily extended to every fixed even $|A|$.

Observation 3.1 (★). *For every fixed even number $\alpha \geq 2$, Full-ALPP on grid graphs is NP-complete even if $|A| = \alpha$.*

The NP-hardness of Full-ALPP for fixed ℓ can be shown also by an easy reduction from a known NP-hard problem, but in this case only for $\ell \geq 4$. This is actually tight as we see later that the problem is polynomial-time solvable when $\ell \leq 3$ (see Theorem 3.3).

Observation 3.2 (★). *For every fixed $\ell \geq 4$, Full-ALPP is NP-complete.*

We can strengthen Observation 3.2 to hold on grid graphs by constructing an involved reduction from scratch. As the proof is long and the theorem does not really fit the theme of this section, we postpone it to Sect. 5.

3.2 Tractable Cases

Theorem 3.3. *If $\ell \leq 3$, then ALPP can be solved in polynomial time.*

Proof. Let (G, A, k, ℓ) with $G = (V, E)$ be an instance of ALPP with $\ell \leq 3$.

If $\ell = 1$, then the problem can be solved by finding a maximum matching in $G[A]$. Since a maximum matching can be found in polynomial time [9], this case is polynomial-time solvable.

Consider the case where $\ell = 2$. We reduce this case to the case of $\ell = 3$. We can assume that $G[A]$ and $G[V \setminus A]$ do not contain any edges as such edges are not included in any $(A, 2)$-path. New instance $(G', A, k, 3)$ is constructed by

[1] A star ★ means that the proof is omitted.

adding a true twin v' to each vertex $v \in V \setminus A$; i.e., $V(G') = V \cup \{v' \mid v \in V \setminus A\}$ and $E(G') = E \cup \{\{v, v'\} \mid v \in V \setminus A\} \cup \{\{u, v'\} \mid u \in A, v \in V \setminus A, \{u, v\} \in E\}$. Clearly, $(G, A, k, 2)$ is a yes-instance if and only if so is $(G', A, k, 3)$.

For the case of $\ell = 3$, we construct an auxiliary graph $G' = (A \cup V_1 \cup V_2, E_{A,1} \cup E_{1,2} \cup E_{2,2})$ as follows (see Fig. 2):

$$\begin{aligned} V_i &= \{v_i \mid v \in V \setminus A\} \text{ for } i \in \{1, 2\}, \\ E_{A,1} &= \{\{u, v_1\} \mid u \in A,\ v \in V \setminus A,\ \{u, v\} \in E\}, \\ E_{1,2} &= \{\{v_1, v_2\} \mid v \in V\}, \\ E_{2,2} &= \{\{u_2, v_2\} \mid u, v \in V \setminus A,\ \{u, v\} \in E\}. \end{aligned}$$

We show that $(G, A, k, 3)$ is a yes-instance if and only if G' has a matching of size $k + |V \setminus A|$, which implies that the problem can be solved in polynomial time.

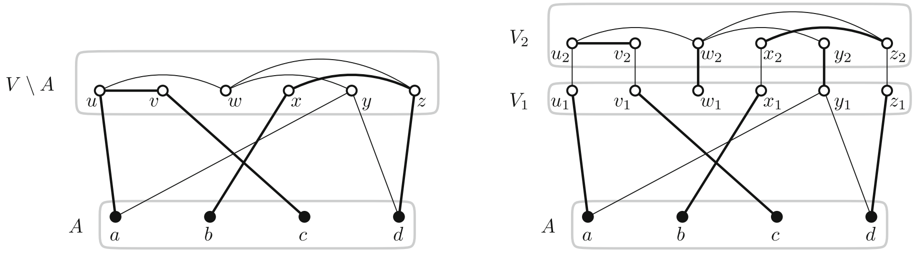

Fig. 2. The construction of G' (right) from G (left).

To prove the only-if direction, let $P_1, \ldots, P_k$ be k vertex-disjoint $(A, 3)$-path in G. We set $M = M_{A,1} \cup M_{1,2} \cup M_{2,2}$, where

$$\begin{aligned} M_{A,1} &= \{\{u, v_1\} \in E_{A,1} \mid \text{edge } \{u, v\} \text{ appears in some } P_i\}, \\ M_{1,2} &= \{\{v_1, v_2\} \in E_{1,2} \mid \text{vertex } v \text{ does not appear in any } P_i\}, \\ M_{2,2} &= \{\{u_2, v_2\} \in E_{2,2} \mid \text{edge } \{u, v\} \text{ appears in some } P_i\}. \end{aligned}$$

Since the $(A, 3)$-paths $P_1, \ldots, P_k$ are pairwise vertex-disjoint, M is a matching. We can see that $|M| = k + |V \setminus A|$ as $|M_{2,2}| = k$ and $|M_{A,1}| + |M_{1,2}| = |V_1| = |V \setminus A|$.

To prove the if direction, assume that G' has a matching of size $k + |V \setminus A|$. Let M be a maximum matching of G' that includes the maximum number of vertices in $V_1 \cup V_2$ among all maximum matchings of G'. We claim that M actually includes all vertices in $V_1 \cup V_2$. Suppose to the contrary that v_1 or v_2 is not included in M for some $v \in V \setminus A$. Now, since M is maximum, exactly one of v_1 and v_2 is included in M.

Case 1: $v_1 \in V(M)$ and $v_2 \notin V(M)$. There is a vertex $u \in A$ such that $\{u, v_1\} \in M$. The set $M - \{u, v_1\} + \{v_1, v_2\}$ is a maximum matching that uses more vertices in $V_1 \cup V_2$ than M. This contradicts how M was selected.

Case 2: $v_1 \notin V(M)$ and $v_2 \in V(M)$. There is a vertex $w_2 \in V_2$ such that $\{v_2, w_2\} \in M$. The edge set $M' := M - \{v_2, w_2\} + \{v_1, v_2\}$ is a maximum matching that uses the same number of vertices in $V_1 \cup V_2$ as M. Since M' is maximum and w_2 is not included in M', the vertex w_1 has to be included in M', but such a case leads to a contradiction as we saw in Case 1.

Now we construct k vertex-disjoint $(A, 3)$-paths from M as follows. Let $\{u_2, v_2\} \in M \cap E_{2,2}$. Since M includes all vertices in V_1, it includes edges $\{u_1, x\}$ and $\{v_1, y\}$ for some $x, y \in A$. This implies that G has an $(A, 3)$-path (x, u, v, y). Let (x', u', v', y') be the $(A, 3)$-path constructed in the same way from a different edge in $M \cap E_{2,2}$. Since M is a matching, these eight vertices are pairwise distinct, and thus (x, u, v, y) and (x', u', v', y') are vertex-disjoint $(A, 3)$-paths. Since $|M| \geq k + |V \setminus A|$ and each edge in $E_{A,1} \cup E_{1,2}$ uses one vertex of V_1, M includes at least k edges in $E_{2,2}$. By constructing an $(A, 3)$-path for each edge in $M \cap E_{2,2}$, we obtain a desired set of k vertex-disjoint $(A, 3)$-paths. □

In their celebrated paper on *Color-Coding* [1], Alon, Yuster, and Zwick showed the following result.

Proposition 3.4 ([1, Theorem 6.3]). *Let H be a graph on h vertices with treewidth t. Let G be a graph on n vertices. A subgraph of G isomorphic to H, if one exists, can be found in time $O(2^{O(h)} \cdot n^{t+1} \log n)$.*

By using Proposition 3.4 as a black box, we can show that ALPP parameterized by $k + \ell$ is fixed-parameter tractable.

Theorem 3.5. *ALPP on n-vertex graphs can be solved in $O(2^{O(k\ell)} n^6 \log n)$ time.*

Proof. Let (G, A, k, ℓ) be an instance of ALPP. Observe that the problem ALPP can be seen as a variant of the Subgraph Isomorphism problem as we search for $H = kP_{\ell+1}$ in G as a subgraph with the restriction that each endpoint of $P_{\ell+1}$ in H has to be mapped to a vertex in A, where $P_{\ell+1}$ denotes an $(\ell+1)$-vertex path (which has length ℓ) and $kP_{\ell+1}$ denotes the disjoint union of k copies of $P_{\ell+1}$. We reduce this problem to the standard Subgraph Isomorphism problem [12].

Let G' and H' be the graphs obtained from G and H, respectively, by subdividing each edge once. The graphs G' and $H' = kP_{2\ell+1}$ are bipartite. We then construct G'' from G' by attaching a triangle to each vertex in A; that is, for each vertex $u \in A$ we add two new vertices v, w and edges $\{u, v\}$, $\{v, w\}$, and $\{w, u\}$. Similarly, we construct H'' from H' by attaching a triangle to each endpoint of each $P_{2\ell+1}$. Note that $|V(G'')| \in O(n^2)$, $|V(H'')| = k(2\ell + 1)$, and $\mathsf{tw}(H'') = 2$. Thus, by Proposition 3.4, it suffices to show that (G, A, k, ℓ) is a yes-instance of ALPP if and only if G'' has a subgraph isomorphic to H''.

To show the only-if direction, assume that G has k vertex-disjoint (A, ℓ)-paths $P_1, \ldots, P_k$. In G'', for each P_i, there is a unique path Q_i of length 2ℓ plus triangles attached to the endpoints; that is, Q_i consists of the vertices of P_i, the new vertices and edges introduced by subdividing the edges in P_i, and the triangles attached to the endpoints of the subdivided path. Furthermore, since the paths P_i are pairwise vertex-disjoint, the subgraphs Q_i of G'' are pairwise vertex-disjoint. Thus, G'' has a subgraph isomorphic to $H'' = \bigcup_{1 \le i \le k} Q_i$.

To prove the if direction, assume that G has a subgraph H' isomorphic to H. Let $R_1, \dots, R_k$ be the connected components of H'. Each R_i is isomorphic to a path of length 2ℓ with a triangle attached to each endpoint. Let $u, v \in V(R_i)$ be the degree-3 vertices of R_i. Since G'' is obtained from the triangle-free graph G' by attaching triangles at the vertices in A, we have $u, v \in A$. Since the u-v path of length 2ℓ in R_i is obtained from a u-v path of length ℓ in G by subdividing each edge once, the graph $G[V(R_i) \cap V(G)]$ contains an (A, ℓ)-path. Since $V(R_1), \dots, V(R_k)$ are pairwise disjoint, G contains k vertex-disjoint (A, ℓ)-paths. □

4 Structural Parameterizations

In this section, we study structural parameterizations of ALPP. First we present XP and FPT algorithms parameterized by tw and $\mathsf{tw} + \ell$, respectively.

The XP-time algorithm parameterized by tw is based on an efficient algorithm for computing a tree decomposition [4] and a standard dynamic-programming over *nice tree decompositions* [14]. The FPT algorithm parameterized $\mathsf{tw} + \ell$ is achieved by expressing the problem in the *monadic second-order logic* (MSO_2) of graphs [2,3,7]. The proofs of them are omitted.

Theorem 4.1 (★). *ALPP can be solved in time $n^{O(\mathsf{tw})}$.*

Theorem 4.2 (★). *ALPP parameterized by $\mathsf{tw} + \ell$ is fixed-parameter tractable.*

Now we show that Full-ALPP is W[1]-hard parameterized by pathwidth (and hence also by treewidth), even if we also consider $|A|$ as an additional parameter. We present a reduction from a W[1]-complete problem k-Multi-Colored Clique (k-MCC) [10], which goes through an intermediate version of our problem. Specifically, we will consider a version of Full-ALPP with the following modifications: the graph has (positive integer) edge weights, and the length of a path is the sum of the weights of its edges; the set A is given to us partitioned into pairs indicating the endpoints of the sought A-paths; for each such pair the value of ℓ may be different.

More formally, Extended-ALPP is the following problem: we are given a graph $G = (V, E)$, a weight function $w \colon E \to \mathbb{Z}^+$, and a sequence of r triples $(s_1, t_1, \ell_1), \dots, (s_r, t_r, \ell_r)$, where all the $s_i, t_i \in V$ are distinct vertices and $\ell_i \in \mathbb{Z}^+$ for all $i \in [r]$[2]. We are asked if there exists a set of r vertex-disjoint paths in G such that for all $i \in [r]$ the i-th path in this set has endpoints s_i, t_i and the sum of the weights of its edges is ℓ_i. We first show that establishing that this variation of the problem is hard implies also the hardness of Full-ALPP.

Lemma 4.3. *There exists an algorithm which, given an instance of Extended-ALPP on an n-vertex graph G with r triples and maximum edge weight W,*

[2] For a positive integer r, we denote the set $\{1, 2, \dots, r\}$ by $[r]$.

constructs in time polynomial in $n+W$ an equivalent instance $(G', A, |A|/2, \ell)$ of Full-ALPP with the properties: (i) $|A| = 2r$, (ii) $\mathsf{pw}(G') \le \mathsf{pw}(G) + 2$.

Proof. First, we simplify the given instance of Extended-ALPP by removing edge weights: for every edge $e = \{u, v\} \in E(G)$ with $w(e) > 1$, we remove this edge and replace it with a path from u to v with length $w(e)$ going through new vertices (in other words we subdivide e $w(e) - 1$ times). It is not hard to see that we have an equivalent instance of Extended-ALPP on the new graph, which we call G_1, where the weight of all edges is 1 and $|V(G_1)| \le n^2W$. We now give a polynomial-time reduction from this new instance of Extended-ALPP to Full-ALPP.

Let $n_1 = |V(G_1)|$ and $\ell = n_1^3$. For each $i \in [r]$ we do the following: we construct a new vertex s'_i and connect it to s_i using a path of length $i \cdot n_1^2$ going through new vertices; we construct a new vertex t'_i and connect it to t_i using a path of length $(n_1 - i) \cdot n_1^2 - \ell_i$ through new vertices. We set A to contain all the vertices s'_i, t'_i for $i \in [r]$. This completes the construction and it is clear that $|A| = 2r$ (because the s_i, t_i vertices are distinct), the new graph G' has order at most $n_1^5 \le n^{10} \cdot W^5$ and can be constructed in time polynomial in $n + W$.

We claim that the new graph G' has $|A|/2$ vertex-disjoint (A, ℓ)-paths if and only if the Extended-ALPP instance of G_1 has a positive answer. Indeed, if there exists a collection of r vertex-disjoint paths in G_1 such that the i-th path has endpoints s_i, t_i and length ℓ_i, we add to this path the paths from s'_i to s_i and from t_i to t'_i and this gives a path of length $\ell = n_1^3$ with endpoints in A. Observe that all these paths are vertex-disjoint, so we obtain a yes-certificate of Full-ALPP. For the converse direction, suppose that G' has a set $\mathcal{A}$ of $|A|/2$ vertex-disjoint (A, ℓ)-paths. If $\mathcal{A}$ contains a path P with endpoints s'_i and s'_j, then considering the length of P we get $(i + j) \cdot n_1^2 + 1 \le n_1^3 \le (i + j) \cdot n_1^2 + n_1 - 1$. The first inequality implies $i + j \le n_1 - 1$, but then this implies $(i + j) \cdot n_1^2 + n_1 - 1 \le n_1^3 - n_1^2 + n_1 - 1 < n_1^3$, a contradiction. Also, there cannot be a path in $\mathcal{A}$ with endpoints t'_i and t'_j, since existence of such a path implies, by the pigeon hole principle, that there is a path in $\mathcal{A}$ with endpoints s'_p and s'_q. Assume that $\mathcal{A}$ contains a path with endpoints s'_i and t'_j. Then the length of this path is at least $i \cdot n_1^2 + (n_1 - j) \cdot n_1^2 - \ell_j + 1$ and at most $i \cdot n_1^2 + (n_1 - j) \cdot n_1^2 - \ell_j + n_1 - 1$. Therefore, if this path has length exactly $\ell = n_1^3$, it must be the case that $i = j$. Furthermore, if $i = j$ we infer that the length of the part of the path from s_i to t_i is exactly ℓ_i. We therefore obtain a solution to the Extended-ALPP instance.

Finally, observe that the only modifications we have done on G is to subdivide some edges and to attach paths to some vertices. By Corollary 2.1, the pathwidth is increased only by at most 2. □

We can now reduce the k-MCC problem to Extended-ALPP.

Lemma 4.4. *There exists a polynomial-time algorithm which, given an instance of k-MCC on a graph G with n vertices, produces an equivalent instance of Extended-ALPP on a graph G', with $r \in O(k^2)$ triples, $\mathsf{pw}(G') \in O(k^2)$, and maximum edge weight $W \in n^{O(1)}$.*

Proof. We are given a graph $G = (V, E)$ with V partitioned into k sets $V_1, \ldots, V_k$, and are asked for a clique of size k that contains one vertex from each set. To ease notation, we will assume that n is even and $|V_i| = n$ for $i \in [k]$ (so the graph has kn vertices in total) and that the vertices of V_i are numbered $1, \ldots, n$. We define two lengths $L_1 = n^3 + (k+1)(2n+2)$ and $L_2 = n^6$.

For $i \in [k]$ we construct a vertex-selection gadget as follows (see Fig. 3): we make $2n+3$ paths of length k, call them $P_{i,j}$, where $j \in [2n+3]$. Let $a_{i,j}, b_{i,j}$ be the first and last vertex of path $P_{i,j}$ respectively. We label the remaining vertices of the path $P_{i,j}$ as $x_{i,j,i'}$ for $i' \in \{1, \ldots, k\} \setminus \{i\}$ in some arbitrary order. Then for each $j \in [2n+2]$ we connect $a_{i,j}$ to $a_{i,j+1}$ and $b_{i,j}$ to $b_{i,j+1}$. All edges constructed so far have weight 1. We add two vertices s_i, t_i, connect s_i to $a_{i,1}$ with an edge of weight $n^3/2$ and t_i to $a_{i,2n+3}$ also with an edge of weight $n^3/2$. We add to the instance the triple (s_i, t_i, L_1).

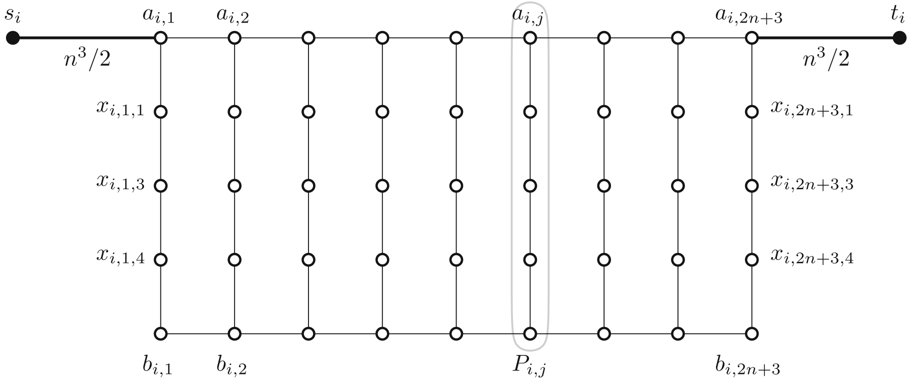

Fig. 3. An example of the vertex-selection gadget for $n = 3$, $k = 4$, and $i = 2$.

We now need to construct an edge-verification gadget as follows (see Fig. 4): for each $i_1, i_2 \in [k]$ with $i_1 < i_2$ we construct three vertices $s_{i_1,i_2}, t_{i_1,i_2}, p_{i_1,i_2}$. For each edge e of G between V_{i_1} and V_{i_2} we do the following: suppose e connects vertex j_1 of V_{i_1} to vertex j_2 of V_{i_2}. We add the following four edges:

1. An edge from s_{i_1,i_2} to $x_{i_1,2j_1,i_2}$. This edge has weight $L_2/4 + j_1 n^4 + j_2 n^2$.
2. An edge from $x_{i_1,2j_1,i_2}$ to p_{i_1,i_2}. This edge has weight $L_2/4$.
3. An edge from p_{i_1,i_2} to $x_{i_2,2j_2,i_1}$. This edge has weight $L_2/4$.
4. An edge from $x_{i_2,2j_2,i_1}$ to t_{i_1,i_2}. This edge has weight $L_2/4 - j_1 n^4 - j_2 n^2$.

We call the edges constructed in the above step heavy edges, since their weight is close to $L_2/4$. We add the $k(k-1)/2$ triples $(s_{i_1,i_2}, t_{i_1,i_2}, L_2)$ to the instance, for all $i_1, i_2 \in [k]$, with $i_1 < i_2$.

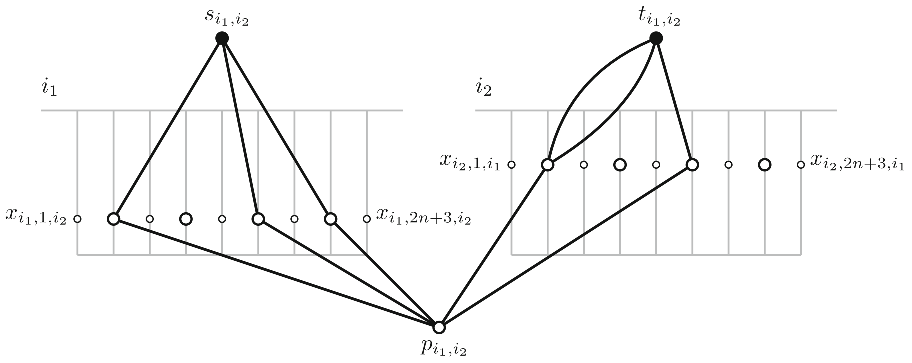

Fig. 4. An example of the edge-verification gadget for V_{i_1} and V_{i_2} ($i_1 < i_2$). In this example, there are exactly three edges between V_{i_1} and V_{i_2}.

Note that in the above description we have created some parallel edges, for example from s_{i_1,i_2} to $x_{i_1,2j_1,i_2}$ (if the vertex j_1 of V_{i_1} has several neighbors in V_{i_2}). This can be avoided by subdividing such edges once and assigning weights to the new edges so that the total weight stays the same. For simplicity we ignore this detail in the remainder since it does not significantly affect the pathwidth of the graph (see Corollary 2.1). This completes the construction.

Let us now prove correctness. First assume that we have a k-multicolored clique in G, encoded by a function $\sigma\colon [k] \to [n]$, that is, $\sigma(i)$ is the vertex of the clique that belongs in V_i. For the i-th vertex-selection gadget we have the triple (s_i, t_i, L_1). We construct a path from s_i to t_i as follows: we take the edge $(s_i, a_{i,1})$, then for each $j < 2\sigma(i)$ we follow the path $P_{i,j}$ from $a_{i,j}$ to $b_{i,j}$ if j is odd, and in the reverse direction if j is even. We thus arrive to the vertex $b_{i,2\sigma(i)-1}$. We then skip the path $P_{i,\sigma(i)}$, proceed through $b_{i,2\sigma(i)}$ to the vertex $b_{i,2\sigma(i)+1}$ and traverse the paths by reversing our parity rule: for $j > 2\sigma(i)$ we traverse $P_{i,j}$ from $b_{i,j}$ to $a_{i,j}$ if j is odd, and in the reverse direction otherwise. Hence, the last vertex of this traversal is $a_{i,2n+3}$, after which we reach t_i. The first and last edge of this path have total cost n^3; we have traversed $2n+2$ paths $P_{i,j}$, each of which has k edges; we have also traversed $2n+2$ edges connecting adjacent paths. The total length is therefore, $n^3 + (2n+2)k + 2n + 2 = L_1$. In this way we have satisfied all the k triples (s_i, t_i, L_1) and have not used the vertices $x_{i,2\sigma(i),i'}$ for any $i' \neq i$.

Consider now a triple $(s_{i_1,i_2}, t_{i_1,i_2}, L_2)$, for $i_1 < i_2$. Because we have selected a clique, there exists an edge between vertex $\sigma(i_1)$ of V_{i_1} and $\sigma(i_2)$ of V_{i_2}. For this edge we have constructed four edges in our new instance, linking s_{i_1,i_2} to t_{i_1,i_2} with a total weight of L_2. We use these paths to satisfy the $\binom{k}{2}$ triples $(s_{i_1,i_2}, t_{i_1,i_2}, L_2)$. These paths are disjoint from each other: when $i_1 < i_2$, $x_{i_1,2\sigma(i_1),i_2}$ is only used in the path from s_{i_1,i_2} to t_{i_1,i_2} and when $i_1 > i_2$, $x_{i_1,2\sigma(i_1),i_2}$ is only used in the path from s_{i_2,i_1} to t_{i_2,i_1}. Furthermore, these paths are disjoint from the paths in the vertex-selection gadgets, as we observed that

$x_{i,2\sigma(i),i'}$ are not used by the path connecting s_i to t_i. We thus have a valid solution. See Fig. 5.

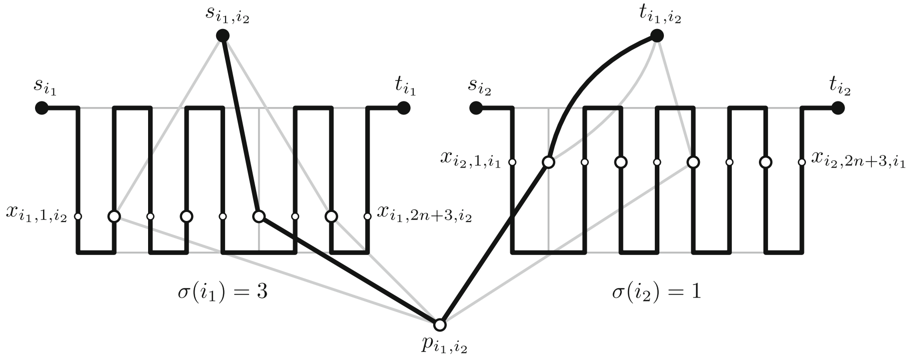

Fig. 5. Construction of paths from σ.

For the converse direction, suppose we have a valid solution for the Extended-ALPP instance. First, consider the path connecting s_i to t_i. This path has length L_1, therefore it cannot be using any heavy edges, since these edges have cost at least $L_2/4 - n^5 - n^3 > L_1$. Inside the vertex-selection gadget, the path may use either all of the edges of a path $P_{i,j}$ or none. Let us now see how many $P_{i,j}$ are unused. First, a simple parity argument shows that, because s_i, t_i are both connected to an $a_{i,j}$ vertex, the number of paths traversed in the $a_{i,j} \rightarrow b_{i,j}$ direction is equal to those traversed in the opposite direction, so the total number of used paths is even. Since we have an odd number of paths in total, at least one path is not used. We conclude that exactly one $P_{i,j}$ is not used, otherwise the path from s_i to t_i would be too short. Let $\sigma(i)$ be defined as the index j such that the internal vertices of $P_{i,j}$ are not used in the $s_i \rightarrow t_i$ path of the solution. We define a clique in G by selecting for each i the vertex $\lfloor \sigma(i)/2 \rfloor$.

Let us argue why this set induces a clique. Let j_1, j_2 be the vertices selected in V_{i_1}, V_{i_2} respectively, with $i_1 < i_2$, and consider the triple $(s_{i_1,i_2}, t_{i_1,i_2}, L_2)$. This triple must be satisfied by a path that uses exactly four heavy edges, since each heavy edge has weight strictly larger than $L_2/5$ and strictly smaller than $L_2/3$ and all other edges together are either incident on another terminal or have weight smaller than $L_2/5n^2$. Hence, every such path is using at least two internal vertices of some $P_{i,j}$ because every heavy edge is incident on such a vertex. But, by our previous reasoning, the paths that satisfy the (s_i, t_i, L_1) triples have used all such vertices except for one path $P_{i,j}$ for each i. There exist therefore exactly $k(k-1)$ such vertices available, so each of the $k(k-1)/2$ triples $(s_{i_1,i_2}, t_{i_1,i_2}, L_2)$ has a path using exactly two of these vertices. Hence, each such path consists of four heavy edges and no other edges.

Such a path must therefore be using one edge incident on s_{i_1,i_2}, one edge incident on t_{i_1,i_2} and two edges incident on p_{i_1,i_2}. The used edge incident on

s_{i_1,i_2} must have as other endpoint $x_{i_1,2j_1,i_2}$, which implies that its weight is $L_2/4 + j_1 n^4 + j'_2 n^2$, for some j'_2. Similarly, the edge incident on t_{i_1,i_2} must have weight $L_2/4 - j'_1 n^4 - j_2 n^2$, as its other endpoint is necessarily $x_{i_2,2j_2,i_1}$. We conclude that the only way that the length of this path is L_2 is if $j_1 = j'_1$ and $j_2 = j'_2$. Therefore, we have an edge between the two selected vertices, and as a result a k-clique.

To conclude we observe that deleting the $O(k^2)$ vertices $s_{i_1,i_2}, p_{i_1,i_2}, t_{i_1,i_2}$ disconnects the graph into components that correspond to the vertex gadgets. Each vertex gadget has pathwidth at most 4 as it can be seen as a subgraph of a subdivision of the $2 \times (2n+4)$ grid. As a result the whole graph has pathwidth $O(k^2)$. □

Theorem 4.5. *Full-ALPP is W[1]-hard parameterized by* $\mathsf{pw} + |A|$.

Proof. We compose the reductions of Lemmas 4.3 and 4.4. Starting with an instance of k-MCC with n vertices this gives an instance of Full-ALPP with $n^{O(1)}$ vertices, $|A| = O(k^2)$, and pathwidth $O(k^2)$. □

5 Hardness on Grid Graphs

We first reduce PLANAR CIRCUIT SAT to Full-ALPP on planar bipartite graphs of maximum degree at most 4. We then modify the instance by subdividing edges and adding terminal vertices in a appropriate way, and have an equivalent instance on grid graphs. All proofs in this section are omitted.

Theorem 5.1 (★). *For every fixed* $\ell \geq 4$, *Full-ALPP is NP-complete on grid graphs.*

6 Concluding Remarks

In this paper, we have introduced a new problem (A, ℓ)-PATH PACKING and showed tight complexity results. One possible future direction would be the parameterization by clique-width cw, a generalization of treewidth (see [13]). In particular, we ask the following two questions.

- Does ALPP admit an algorithm of running time $O(n^{\mathsf{cw}})$?
- Is ALPP fixed-parameter tractable parameterized by $\mathsf{cw} + \ell$?

References

1. Alon, N., Yuster, R., Zwick, U.: Color-coding. J. ACM **42**(4), 844–856 (1995). https://doi.org/10.1145/210332.210337
2. Arnborg, S., Lagergren, J., Seese, D.: Easy problems for tree-decomposable graphs. J. Algorithms **12**(2), 308–340 (1991). https://doi.org/10.1016/0196-6774(91)90006-K

3. Bodlaender, H.L.: A linear-time algorithm for finding tree-decompositions of small treewidth. SIAM J. Comput. **25**(6), 1305–1317 (1996). https://doi.org/10.1137/S0097539793251219
4. Bodlaender, H.L., Drange, P.G., Dregi, M.S., Fomin, F.V., Lokshtanov, D., Pilipczuk, M.: A $c^k n$ 5-approximation algorithm for treewidth. SIAM J. Comput. **45**(2), 317–378 (2016). https://doi.org/10.1137/130947374
5. Chudnovsky, M., Cunningham, W.H., Geelen, J.: An algorithm for packing non-zero a-paths in group-labelled graphs. Combinatorica **28**(2), 145–161 (2008). https://doi.org/10.1007/s00493-008-2157-8
6. Chudnovsky, M., Geelen, J., Gerards, B., Goddyn, L.A., Lohman, M., Seymour, P.D.: Packing non-zero a-paths in group-labelled graphs. Combinatorica **26**(5), 521–532 (2006). https://doi.org/10.1007/s00493-006-0030-1
7. Courcelle, B.: The monadic second-order logic of graphs III: tree-decompositions, minor and complexity issues. Theor. Inf. Appl. **26**, 257–286 (1992). https://doi.org/10.1051/ita/1992260302571
8. Cygan, M., et al.: Parameterized Algorithms. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21275-3
9. Edmonds, J.: Paths, trees, and flowers. Can. J. Math. **17**, 449–467 (1965). https://doi.org/10.4153/CJM-1965-045-4
10. Fellows, M.R., Hermelin, D., Rosamond, F.A., Vialette, S.: On the parameterized complexity of multiple-interval graph problems. Theor. Comput. Sci. **410**(1), 53–61 (2009). https://doi.org/10.1016/j.tcs.2008.09.065
11. Gallai, T.: Maximum-Minimum Sätze und verallgemeinerte Faktoren von Graphen. Acta Mathematica Academiae Scientiarum Hungarica **12**, 131–173 (1961). https://doi.org/10.1007/BF02066678
12. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman W. H., New York (1979)
13. Hlinený, P., Oum, S., Seese, D., Gottlob, G.: Width parameters beyond tree-width and their applications. Comput. J. **51**(3), 326–362 (2008). https://doi.org/10.1093/comjnl/bxm052
14. Kloks, T. (ed.): Treewidth, Computations and Approximations. LNCS, vol. 842. Springer, Heidelberg (1994). https://doi.org/10.1007/BFb0045375
15. Mader, W.: Über die maximalzahl kreuzungsfreier H-wege. Archiv der Mathematik (Basel) **31**, 387–402 (1978). https://doi.org/10.1007/BF01226465
16. Monnot, J., Toulouse, S.: The path partition problem and related problems in bipartite graphs. Oper. Res. Lett. **35**(5), 677–684 (2007). https://doi.org/10.1016/j.orl.2006.12.004
17. Nešetřil, J., Ossona de Mendez, P.: Sparsity - Graphs, Structures, and Algorithms. AC, vol. 28. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27875-4
18. Pap, G.: Packing non-returning a-paths. Combinatorica **27**(2), 247–251 (2007). https://doi.org/10.1007/s00493-007-0056-z
19. Pap, G.: Packing non-returning a-paths algorithmically. Discrete Math. **308**(8), 1472–1488 (2008). https://doi.org/10.1016/j.disc.2007.07.073
20. Steiner, G.: On the k-path partition of graphs. Theor. Comput. Sci. **290**(3), 2147–2155 (2003). https://doi.org/10.1016/S0304-3975(02)00577-7
21. Yan, J.H., Chang, G.J., Hedetniemi, S.M., Hedetniemi, S.T.: k-path partitions in trees. Discret. Appl. Math. **78**(1–3), 227–233 (1997). https://doi.org/10.1016/S0166-218X(97)00012-7