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[3]

[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, otachi@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@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 JPMJCR1401. The authors thank Tatsuya Gima for helpful discussions. This paper is eligible for the best student paper award because Masaaki Kanzaki is a full-time student who contributed to the paper significantly.

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., [14,22,7,6,25,26]). 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.

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. [10] 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.

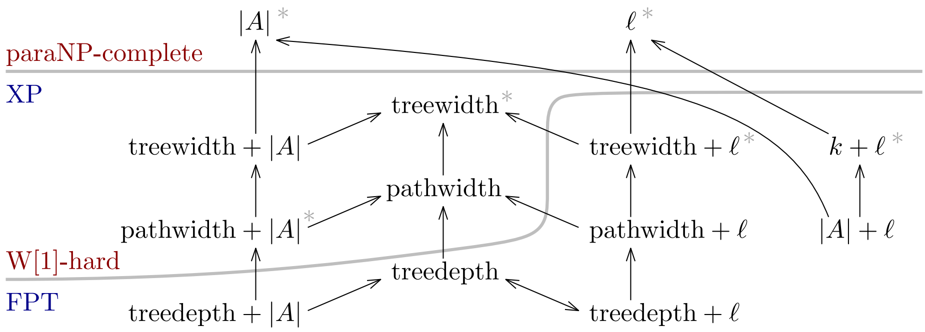


Fig. 1: Summary of the results. An arrow $\alpha \to \beta$ indicates that there is a function f such that $\alpha \geq f(\beta)$ for every instance of ALPP. Some possible arrows are omitted to keep the figure readable. The results on the parameters marked with $*$ are explicitly shown in this paper, and the other results follow by the hierarchy of the parameters. We have a bidirectional arrow treedepth $\leftrightarrow$ treedepth $+ \ell$ because the maximum length of a path in a graph is bounded by a function of treedepth [24, 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 subdividing some edges and attaching paths to some vertices do not change pathwidth significantly.

Corollary 2.1 (★[8]). *Let $G = (V, E)$ be a graph without isolated vertices. If G' is a graph obtained from G by subdividing a set of edges $F \subseteq E$ arbitrary 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 [15]. This observation is easily extended to every fixed even $|A|$.

Observation 3.1 (★). *For every fixed even $\alpha \geq 2$, 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 Section 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 $G[A]$, which can be done in polynomial time [11].

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

[8] A star ★ means that the proof is omitted and moved to the appendix.

are not included in any $(A, 2)$-path. New instance $(G', A, k, 3)$ is constructed by 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.

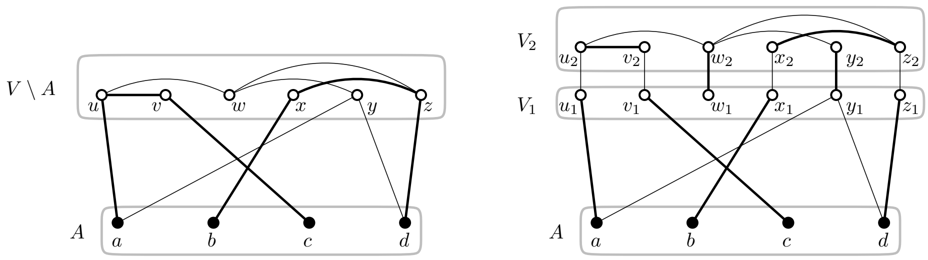


Fig. 2: The construction of G' (right) from G (left).

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 [15].

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. $\square$

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 [5] and a standard dynamic-programming over *nice tree decompositions* [20]. The FPT algorithm parameterized $\mathsf{tw} + \ell$ is achieved by expressing the problem in the *monadic second-order logic* (MSO_2) of graphs [2,9,3]. 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) [13], 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 $i \in [r]$[9]. 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.

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

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, 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) $\mathsf{pw}(G') \leq \mathsf{pw}(G) + 2$.*

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 $|V(G_1)| \leq n^2 W$. 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 $n_1^5 \leq n^{10} \cdot W^5$ 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 $(i + j) \cdot n_1^2 + 1 \leq n_1^3 \leq (i + j) \cdot n_1^2 + n_1 - 1$. The first inequality implies $i + j \leq n_1 - 1$, but then this implies $(i + j) \cdot n_1^2 + n_1 - 1 \leq n_1^3 - n_1^2 + n_1 - 1 < n_1^3$, 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).

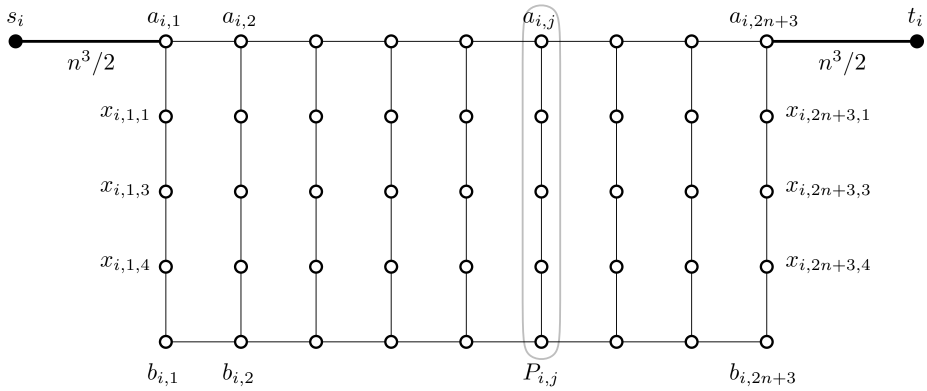


Fig. 3: An example of the vertex-selection gadget for $n = 3$, $k = 4$, and $i = 2$.

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 $i_1 < i_2$.

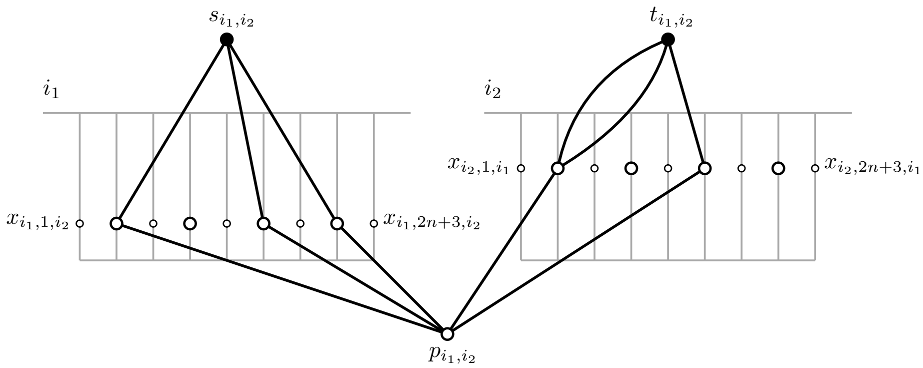


Fig. 4: An example of the edge-verification gadget for V_{i_1} and V_{i_2} ($i_1 < i_2$). In this example, there are exactly three edges between V_{i_1} and V_{i_2}.

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.

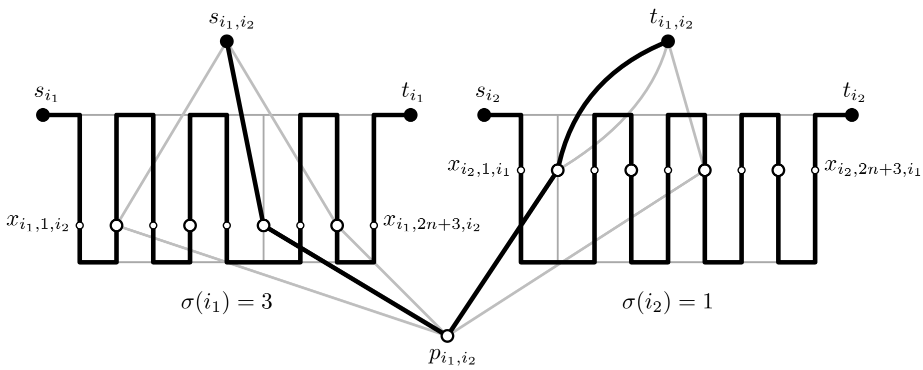


Fig. 5: Construction of paths from σ.

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 $a_{i,j} \to b_{i,j}$ 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 $s_i \to t_i$ 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 is 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 [16]). 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. Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995. doi:10.1145/210332.210337.
2. Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *J. Algorithms*, 12(2):308–340, 1991. doi:10.1016/0196-6774(91)90006-K.

3. Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996. doi:10.1137/S0097539793251219.
4. Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1):1–45, 1998. doi:10.1016/S0304-3975(97)00228-4.
5. Hans L. Bodlaender, Pål Grønås Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michał Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM J. Comput.*, 45(2):317–378, 2016. doi:10.1137/130947374.
6. Maria Chudnovsky, William H. Cunningham, and Jim Geelen. An algorithm for packing non-zero a-paths in group-labelled graphs. *Combinatorica*, 28(2):145–161, 2008. doi:10.1007/s00493-008-2157-8.
7. Maria Chudnovsky, Jim Geelen, Bert Gerards, Luis A. Goddyn, Michael Lohman, and Paul D. Seymour. Packing non-zero a-paths in group-labelled graphs. *Combinatorica*, 26(5):521–532, 2006. doi:10.1007/s00493-006-0030-1.
8. Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC 1971*, pages 151–158, 1971. doi:10.1145/800157.805047.
9. Bruno Courcelle. The monadic second-order logic of graphs III: tree-decompositions, minor and complexity issues. *Theor. Inform. Appl.*, 26:257–286, 1992. doi:10.1051/ita/1992260302571.
10. Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
11. Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965. doi:10.4153/CJM-1965-045-4.
12. John A. Ellis, Ivan Hal Sudborough, and J. Turner. The vertex separation and search number of a graph. *Inform. Comput.*, 113:50–79, 1994. doi:10.1006/inco.1994.1064.
13. Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theor. Comput. Sci.*, 410(1):53–61, 2009. doi:10.1016/j.tcs.2008.09.065.
14. Tibor Gallai. Maximum-minimum sätze und verallgemeinerte faktoren von graphen. *Acta Mathematica Academiae Scientarim Hungaricae*, 12:131–137, 1961. doi:10.1007/BF02066678.
15. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
16. Petr Hliněný, Sang-il Oum, Detlef Seese, and Georg Gottlob. Width parameters beyond tree-width and their applications. *Comput. J.*, 51(3):326–362, 2008. doi:10.1093/comjnl/bxm052.
17. Alon Itai, Christos H. Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM J. Comput.*, 11(4):676–686, 1982. doi:10.1137/0211056.
18. David G. Kirkpatrick and Pavol Hell. On the complexity of general graph factor problems. *SIAM J. Comput.*, 12(3):601–609, 1983. doi:10.1137/0212040.
19. Lefteris M. Kirousis and Christos H. Papadimitriou. Interval graphs and searching. *Discrete Mathematics*, 55:181–184, 1985. doi:10.1016/0012-365X(85)90046-9.
20. Ton Kloks. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer, 1994. doi:10.1007/BFb0045375.
21. Y. Liu, A. Morgana, and B. Simeone. A linear algorithm for 2-bend embeddings of planar graphs in the two-dimensional grid. *Discrete Applied Mathematics*, 81(1):69–91, 1998. doi:10.1016/S0166-218X(97)00076-0.

22. Wolfgang Mader. Über die maximalzahl kreuzungsfreier H-wege. *Archiv der Mathematik (Basel)*, 31:387–402, 1978. doi:10.1007/BF01226465.
23. William F. McColl. Planar crossovers. *IEEE Trans. Computers*, 30(3):223–225, 1981. doi:10.1109/TC.1981.1675758.
24. Jaroslav Nesetril and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012. doi:10.1007/978-3-642-27875-4.
25. Gyula Pap. Packing non-returning a-paths. *Combinatorica*, 27(2):247–251, 2007. doi:10.1007/s00493-007-0056-z.
26. Gyula Pap. Packing non-returning a-paths algorithmically. *Discrete Mathematics*, 308(8):1472–1488, 2008. doi:10.1016/j.disc.2007.07.073.

A Omitted proofs in Section 2

A.1 Proof of Corollary 2.1

A useful characterization of pathwidth is via the following search game. We are given a graph $G = (V, E)$ with all edges *contaminated*. The goal in this game is to clear all edges. In each turn, we can place a *searcher* on a vertex or delete a searcher from a vertex. An edge is *cleared* by having searchers on both endpoints. A cleared edge is immediately *recontaminated* when a deletion of a searcher results in a path not passing through any searchers from the edge to a contaminated edge. The minimum number of searchers needed to clear all edges of G is the *node search number*, and we denote it by $\mathsf{ns}(G)$. It is known that $\mathsf{ns}(G) = \mathsf{pw}(G) + 1$ for every graph G [19,12,4].

Lemma A.1 (Folklore). *Let $G = (V, E)$ be a graph. If G' is a graph obtained from G by subdividing a set of edges $F \subseteq E$ arbitrary times, then $\mathsf{pw}(G') \leq \mathsf{pw}(G) + 2$.*

Proof. Let $p = \mathsf{pw}(G) + 1$. Since $\mathsf{ns}(G) \leq p$, there is a sequence S of placements and deletions of searchers to clear all edges of G using at most p searchers. To clear all edges of G', we extend S as follows. For each placement of a searcher on a vertex $v \in V$, we insert, right after this placement, a subsequence that clears all paths corresponding to the edges between v and its neighbors having searchers on them at this point. This can be done with two extra searchers that clear the paths one-by-one The extra searchers are deleted in the end of the subsequence, and thus we only need two extra searchers in total. This implies that $\mathsf{pw}(G') = \mathsf{ns}(G') - 1 \leq \mathsf{ns}(G) + 1 = \mathsf{pw}(G) + 2$. □

Lemma A.2 (Folklore). *Let $G = (V, E)$ be a graph without isolated vertices. If G' is a graph obtained from G by attaching a path of arbitrary length to each vertex in a set $U \subseteq V$, then $\mathsf{pw}(G') \leq \mathsf{pw}(G) + 1$.*

Proof. There is a sequence S of placements and deletions of searchers to clear all edges of G using at most $p = \mathsf{pw}(G) + 1$ searchers. To clear all edges of G', we extend S as follows. We replace each placement of a searcher on a vertex

$u \in U$ with a subsequence that clears the path $P = (u, p_1, p_2, \dots, p_q)$ attached to u. This can be done with two searchers by first placing a searcher on p_q, then placing a searcher on p_{q-1}, deleting a searcher on p_q, placing a searcher on p_{q-2}, and so on. At the end of the subsequence, u has a searcher on it and all other vertices in P do not have searchers. We need only one extra searcher in total. Hence, $\mathsf{pw}(G') = \mathsf{ns}(G') - 1 \leq \mathsf{ns}(G) = \mathsf{pw}(G) + 1$. ☐

In the proofs of Lemmas A.1 and A.2, we show that search sequences for the original graph can be "locally" extended for the new graph by using one or two temporal searchers. Thus we can have the following combined version of the lemmas as Corollary 2.1.

B Omitted proofs in Section 3

B.1 Proof of Observation 3.1

Proof. Let $G = (V, E)$ be an instance of Hamiltonian Cycle on grid graphs, which is known to be NP-complete [17]. Since Full-ALPP is clearly in NP, it suffices to construct an equivalent instance of Full-ALPP in polynomial time.

Observe that the minimum degree $\delta(G)$ of G is at most 2. If $\delta(G) < 2$, then G is a no-instance of Hamiltonian Cycle, and thus we can construct trivial no-instance of Full-ALPP. Assume that $\delta(G) = 2$, and let v be a vertex of degree 2 in G with the neighbors u and w. Let G' be the graph obtained from G by removing v and adding a (possibly empty) set of $\alpha/2 - 1$ new paths of length $|V| - 2$. Let Q be the set of endpoints of the new paths. Then, $(G', \{u, w\} \cup Q, \alpha/2, |V| - 2)$ is a yes-instance of Full-ALPP if and only if G has a Hamiltonian cycle. This can be seen by observing that each Hamiltonian cycle of G includes edges $\{u, v\}$ and $\{v, w\}$ and that each $(\{u, w\}, |V| - 2)$-path in G' can be extended to a Hamiltonian cycle of G by using v and the edges $\{u, v\}$ and $\{v, w\}$. ☐

B.2 Proof of Observation 3.2

Proof. Given a graph $G = (V, E)$, the λ-Path Partition problem asks whether G contains $k := |V|/(\lambda + 1)$ vertex-disjoint paths of length λ. For every fixed $\lambda \geq 2$, λ-Path Partition is NP-complete [18]. We construct G' from G by adding a set A of $2k$ new vertices, where A is an independent set in G' and G' has all possible edges between A and V. We can see that G is a yes-instance of λ-Path Partition if and only if (G', A, k, ℓ) is a yes-instance of Full-ALPP, where $\ell = \lambda + 2 \geq 4$. ☐

C Omitted proofs in Section 4

C.1 Proof of Theorem 4.1

Proof. Let G be an n-vertex graph of treewidth at most tw. We compute the maximum number of vertex-disjoint (A, ℓ)-paths by a standard dynamic programming algorithm over a tree decomposition. To this end, it is helpful to use so called nice tree decompositions [20]. A tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ is *nice* if T is a rooted tree, each node of T has at most two children, and

- if $i \in I$ has no children, then $|X_i| = 1$;
- if $i \in I$ has exactly one child j, then $X_i = X_j \cup \{u\}$ for some $u \notin X_j$ or $X_i = X_j \setminus \{v\}$ for some $v \in X_j$;
- if $i \in I$ has exactly two children j and h, then $X_i = X_j = X_h$.

We compute a tree decomposition of width at most $w = 5\mathsf{tw} + 4$ in time $2^{O(\mathsf{tw})}n$ [5], and then convert it in linear time to a nice tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ of the same width having $O(n)$ nodes in the tree T [20].

Let $V_i = \bigcup_j X_j$, where the union is taken over all descendants j of i in T (including i itself). For each $i \in I$, we define the DP table $\mathsf{dp}_i(\alpha, \lambda, \delta, \kappa) \in \{\mathsf{true}, \mathsf{false}\}$ with the indices $\alpha \colon X_i \to \{B \subseteq A \mid |B| \leq 2\}$, $\lambda \colon X_i \to \{0, \ldots, \ell\}$, $\delta \colon X_i \to \{0, 1, 2\}$, $\kappa \in \{0, \ldots, |A|/2\}$ such that $\mathsf{dp}_i(\alpha, \lambda, \delta, \kappa) = \mathsf{true}$ if and only if there exists a spanning subgraph H of $G[V_i]$ such that all the following conditions are satisfied.

- all connected components of H are paths, and κ of them are (A, ℓ)-paths;
- each vertex in $A \cap V_i$ has degree at most 1 in H;
- for each $v \in X_i$, $\alpha(v) = V(C(v)) \cap A$, where $C(v)$ is the connected component of H containing v;
- $C(v)$ contains exactly $\lambda(v)$ edges for each $v \in X_i$;
- each $v \in X_i$ has degree $\delta(v)$ in H.

The size of the table dp_i is $O(|A|^{2w} \cdot (\ell + 1)^w \cdot 3^w \cdot |A|/2)$. Since $|A|$ and ℓ are at most n, this table size can be bounded by $n^{O(\mathsf{tw})}$. If we know all table entries for the root r of T, then we can find the maximum number of vertex-disjoint (A, ℓ)-paths in G in time $n^{O(\mathsf{tw})}$, by just finding the maximum number κ such that there exist α, λ, and δ with $\mathsf{dp}_i(\alpha, \lambda, \delta, \kappa) = \mathsf{true}$.

It is trivial to compute all the entries for a node i with no children in time $n^{O(\mathsf{tw})}$. For a node i with one or two children, if the table entries for the children are already computed, then it is straightforward to compute the table entries for i in time polynomial in the total table size of the children. This running time is again $n^{O(\mathsf{tw})}$. Since there are $O(n)$ nodes in T, the total running times is $n^{O(\mathsf{tw})}$. □

C.2 Proof of Theorem 4.2

Proof. To show the fixed-parameter tractability of ALPP parameterized by $\mathsf{tw}+\ell$, we use the *monadic second-order logic* (MSO_2) of graphs. In an MSO_2 formula, we can use (i) the logical connectives $\vee$, $\wedge$, $\neg$, $\Leftrightarrow$, $\Rightarrow$, (ii) variables for vertices, edges, vertex sets, and edge sets, (iii) the quantifiers $\forall$ and $\exists$ applicable to these variables, and (iv) the following binary relations:

- $u \in U$ for a vertex variable u and a vertex set variable U;
- $d \in D$ for an edge variable d and an edge set variable D;

– $\mathsf{inc}(d, u)$ for an edge variable d and a vertex variable u, where the interpretation is that d is incident with u;
– equality of variables.

In the following expressions, we use some syntax sugars, such as $\neq$ and $\notin$, obviously obtained from the definition of MSO_2 for ease of presentation. Also, we follow the convention that in MSO_2 formulas, the set variables V and E denote the vertex and edges sets of the input graph, respectively.

Let φ be a fixed MSO_2 formula. It is known that given an n-vertex graph of treewidth w and assignments to some free variables of φ, one can find in time $O(f(|\varphi| + w) \cdot n)$, where f is some computable function, assignments to the rest of free variables that satisfies φ and maximizes a given linear function in the sizes of the free variables of φ [2,9,3].

We can express a formula (A, ℓ)-$\mathsf{paths}(F)$ that is true if and only if F is the edge set of a set of (A, ℓ)-paths as follows:

$$\begin{aligned}(A, \ell)\text{-}\mathsf{paths}(F) := {} & \mathsf{paths}(F) \wedge \ell\text{-}\mathsf{components}(F) \\ & \wedge (\forall v \in V \ (\mathsf{deg}_{=1}(v, F) \implies v \in A)),\end{aligned}$$

where $\mathsf{paths}(F)$ is true if and only if F is the edge set of a set of paths, ℓ-$\mathsf{components}(F)$ is true if and only if F is the edge set of a graph that only has size-ℓ components, and $\mathsf{deg}_{=1}(v, F)$ is true if and only if exactly one edge in F has v as an endpoint. We can easily express these three subformulas in such a way that the length of the formula (A, ℓ)-$\mathsf{paths}(F)$ depends only on ℓ. (See Appendix C.3.)

The formula (A, ℓ)-$\mathsf{paths}(F)$ has two free variables A and F. We assign (or identify) the terminal vertex set A in the input of ALPP to the variable A, and maximize the size of F. As mentioned above, this can be done in time $O(f(|\varphi| + w) \cdot n)$ for n-vertex graphs of treewidth at most w, where f is some computable function. □

C.3 Proof of Theorem 4.2

We present the three missing expressions in the proof of Theorem 4.2. Recall that $\mathsf{paths}(F)$ is true if and only if F is the edge set of a set of paths, ℓ-$\mathsf{components}(F)$ is true if and only if F is the edge set of a graph and $\mathsf{deg}_{=1}(v, F)$ is true if and only if exactly one edge in F has v as an endpoint.

The following formula $\mathsf{deg}_{\leq d}(v, D)$ has length depending only on d and is true if and only if at most d edges in D has v as an endpoint.

$$\mathsf{deg}_{\leq d}(v, D) := \nexists e_1, \ldots, e_{d+1} \in D \left(\bigwedge_{1 \leq i \leq d+1} \mathsf{inc}(e_i, v) \right).$$

Now it is straightforward to express $\mathsf{deg}_{=1}(v, D)$ and $\mathsf{paths}(F)$:

$$\begin{aligned}\mathsf{deg}_{=1}(v, D) &:= \mathsf{deg}_{\leq 1}(v, D) \wedge \neg \mathsf{deg}_{\leq 0}(v, D), \\ \mathsf{paths}(F) &:= \big(\forall v \in V(\mathsf{deg}_{\leq 2}(v, F))\big) \wedge \big(\forall C \subseteq F, \exists v \in V(\mathsf{deg}_{\leq 1}(v, C))\big).\end{aligned}$$

We can express ℓ-$\mathsf{components}(F)$ as follows

$$\ell\text{-}\mathsf{components}(F) := \forall C \subseteq F(\mathsf{component}(C, F) \implies \mathsf{size}_{=\ell}(C)),$$

where $\mathsf{component}(C, F)$ is true if and only if C is the edge set of a connected component (i.e., an inclusion-wise maximal connected subgraph) of the graph induced by F, and $\mathsf{size}_{=\ell}(C))$ is true if and only if C includes exactly ℓ edges.

As we allow the expression of $\mathsf{size}_{=\ell}(C)$ to have length depending on ℓ, it is trivially expressible, e.g., as follows:

$$\begin{aligned}\mathsf{size}_{=\ell}(C) := {} & \exists e_1, \dots, e_\ell \in C \\ & \left(\bigwedge_{1 \le i < j \le \ell} (e_i \neq e_j) \wedge \left(\forall e' \left(\bigwedge_{1 \le i \le \ell} (e_i \neq e') \implies e' \notin C \right) \right) \right).\end{aligned}$$

Expressing the connectivity of the graph induced by an edge set C is a nice exercise and well known to have the following solution:

$$\begin{aligned}\mathsf{connected}(C) := {} & \exists U \subseteq V(\forall v \in V(v \in U \iff \exists e \in C(\mathsf{inc}(e, v)))) \\ & \wedge (\forall W \subseteq U(W = U \vee (\exists w \in W, \exists z \notin W, \mathsf{adj}(w, z)))),\end{aligned}$$

where $\mathsf{adj}(w, z) := \exists e \in E(\mathsf{inc}(e, w) \wedge \mathsf{inc}(e, z))$. Using this expression, $\mathsf{component}(C, F)$ can be presented as follows:

$$\begin{aligned}\mathsf{component}(C, F) := {} & \mathsf{connected}(C) \\ & \wedge \forall e \in F(e \notin C \implies \neg\mathsf{connected}(C \cup \{e\})).\end{aligned}$$

D Omitted proofs in Section 5

The input of Circuit SAT is a Boolean circuit with a number of inputs and one output. The question is whether the circuit can output true by appropriately setting its inputs. Circuit SAT is NP-complete since CNF SAT [8] can be see as a special case. When the underlying graph of the circuit is planar, the problem is called Planar Circuit SAT. Using planar crossover gadgets [23], we can show that Planar Circuit SAT is NP-complete. Furthermore, since NOR gates can replace other gates such as AND, OR, NOT, NAND, and XOR without introducing any new crossing, we can conclude that Planar Circuit SAT having NOR gates only is NP-complete.

Let $I = (G, A, \ell)$ be an instance of Full-ALPP with $G = (V, E)$. Let ψ be a mapping that assigns each $e \in E$ an (A, ℓ)-path in G, and $\psi(E) = \{\psi(e) \mid e \in E\}$. We say that ψ is a *guide* to I if every set of $|A|/2$ vertex-disjoint (A, ℓ)-paths, if any exists, is a subset of $\psi(E)$. When a guide is given additionally to an instance of Full-ALPP, we call the problem Guided Full-ALPP. Observe that a guide to an instance is not a restriction but just additional information.

Lemma D.1. *For every fixed $\ell \geq 4$, Guided Full-ALPP is NP-complete on planar bipartite graphs of maximum degree at most 4.*

Proof. Given a planar circuit with only NOR gates, we construct an equivalent instance of Guided Full-ALPP with the fixed ℓ. We only need input gadget, output gadget, split gadget, NOR gadget, and a way to connect the gadgets. See Fig. 6 for the high-level idea of the reduction.

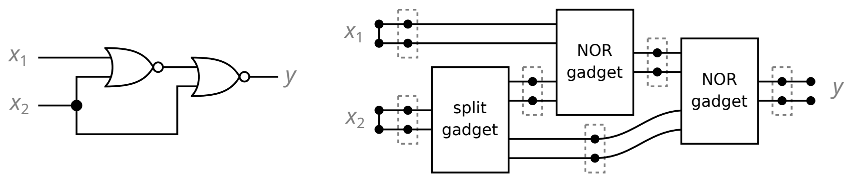


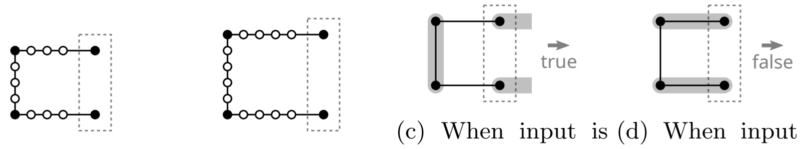
Fig. 6: A planar circuit and the corresponding ALPP instance (simplified). The vertices in $V \setminus A$ are omitted. The connection pairs are marked with dashed rectangles.

We explicitly present the gadgets for the cases $\ell = 4$ and $\ell = 5$. For even (odd) $\ell > 5$, the gadgets can be obtained from the one for $\ell = 4$ ($\ell = 5$, resp.) by subdividing $\lfloor \ell/2 \rfloor - 2$ times each edge incident to a vertex in A.

Connections between gadgets. We first explain how the gadgets are connected. Each gadget has one, two, or three pairs of vertices that are shared with other gadgets. We call them *connection pairs*. All those vertices belong to the terminal set A. In the figures, we draw each connection pair so that the two vertices are next to each other vertically and mark them with a dashed rectangle. If the (A, ℓ)-paths using the vertices of a connection pair are going to the positive direction, then we interpret it as that a true signal is sent via the connection pair. If the paths are going to the negative direction, then the connection pair is carrying a false signal. (See Fig. 7.) Note that our reduction below forces the paths at each connection pair to proceed in the same direction.

Fig. 7: (A, ℓ)-paths at a connection pair. We draw an (A, ℓ)-path as a gray bar.

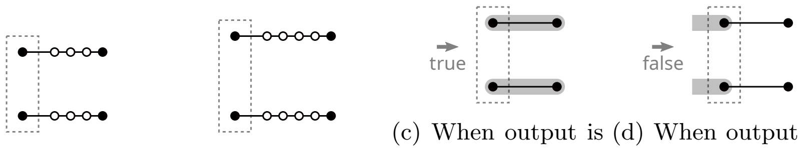
Input gadgets. The input gadget is simply a path of length 3ℓ, where the endpoints form its unique connection pair. See Fig. 8. For a full (A, ℓ)-path packing, we only have two options. One corresponds to true input (Fig. 8c) and the other to false input (Fig. 8d).



(a) $\ell = 4$. (b) $\ell = 5$. (c) When input is true. (d) When input is false.

Fig. 8: The input gadgets. The black vertices belong to A and the white vertices belong to $V \setminus A$.

Output gadgets. The output gadget consists of two paths of length ℓ, where its unique connection pair includes one endpoint from each path. See Fig. 9. To have a full packing, the input to this gadget has to be true.



(a) $\ell = 4$ (b) $\ell = 5$ (c) When output is true. (d) When output is false.

Fig. 9: The output gadgets.

Split gadgets. To simulate the split of a wire depicted in Fig. 10, the split gadget consists of three paths of length 3ℓ, each of which is identical to the input gadget, and a cycle of length 10ℓ that synchronizes the three paths. See Fig. 11. To have a full (A, ℓ)-path packing, there are only two ways to pack (A, ℓ)-paths into a split gadget. Fig. 12 shows the two ways: one on the left corresponds to a split of a true signal, and the other a split of a false signal.

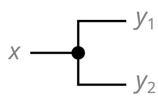
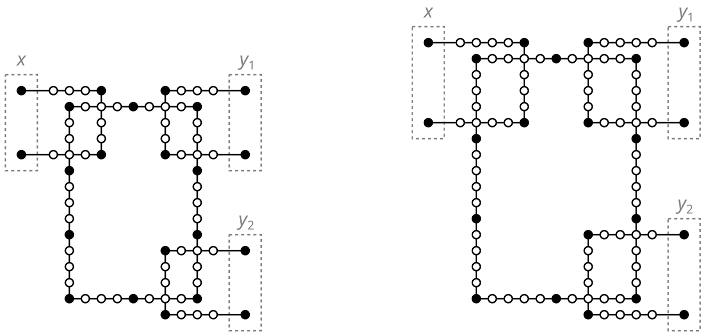


Fig. 10: Splitting a wire in a circuit.

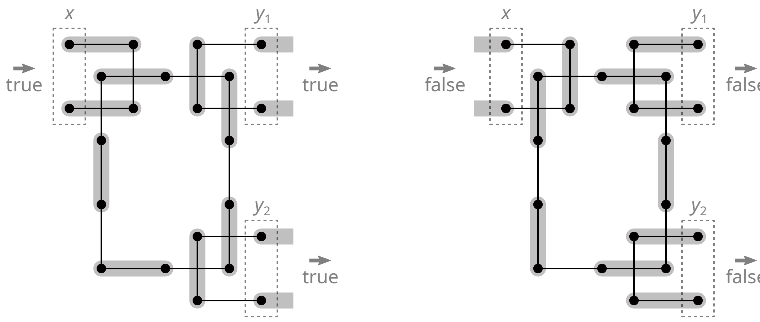
NOR gadgets. Recall that NOR stands for "NOT OR" and that the output y of a NOR gate is true if and only if both inputs x_1 and x_2 are false. The NOR gadgets are given in Fig. 13. The structure of the gadget is rather involved. It has three connection pairs, two for the inputs and one for the output, and the endpoints of each pair are connected by a path of length 5ℓ. Additionally, there is a long self-intersecting cycle that somehow entangles the inputs and the output.



(a) $\ell = 4$.

(b) $\ell = 5$.

Fig. 11: The split gadgets.



(a) Splitting true signal.

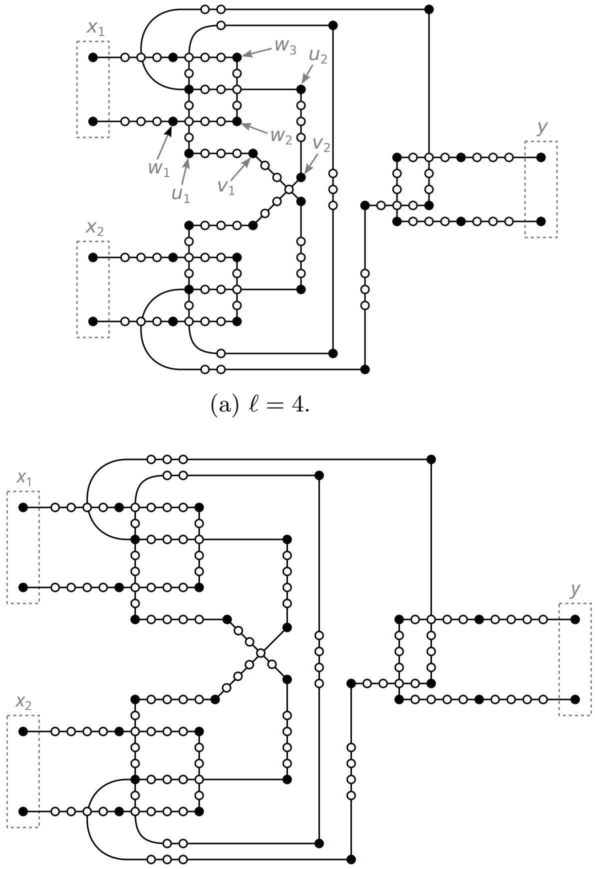
(b) Splitting false signal.

Fig. 12: The possible (A, ℓ)-path packings of the split gadget.

There are only four ways to fully pack (A, ℓ)-paths into a NOR gadget, and each packing corresponds to a correct behavior of a NOR gate (see Fig. 14). To see the correctness of Fig. 14, it is important to observe that in the NOR gadgets for even ℓ, there are some (A, ℓ)-paths that are never used in a full (A, ℓ)-path packing. For example, the (A, ℓ)-path with endpoints v_1 and v_2 in Fig. 13a is such a path. In a full (A, ℓ)-path packing, w_2 has to be an endpoint of an (A, ℓ)-path either with w_1 or w_3. Hence, if we use the (A, ℓ)-path with endpoints v_1 and v_2, then one of u_1 and u_2 cannot belong to any (A, ℓ)-path in the packing.

Guides. The guide $\psi(e)$ for each $e \in E$ can be easily set from Figures 8c, 8d, 9c, 12a, 12b, 14a, 14b, 14c, and 14d. For each edge e, the unique gray bar that includes the edge represents the (A, ℓ)-path $\psi(e)$.

Correctness. The correctness of each gadget implies the correctness of the whole reduction. Thus, it suffices to show that the output of the reduction is planar bipartite graph of maximum degree at most 4. The resultant graph clearly has maximum degree 4 and is planar. To see that the graph is bipartite, consider



(a) $\ell = 4$.

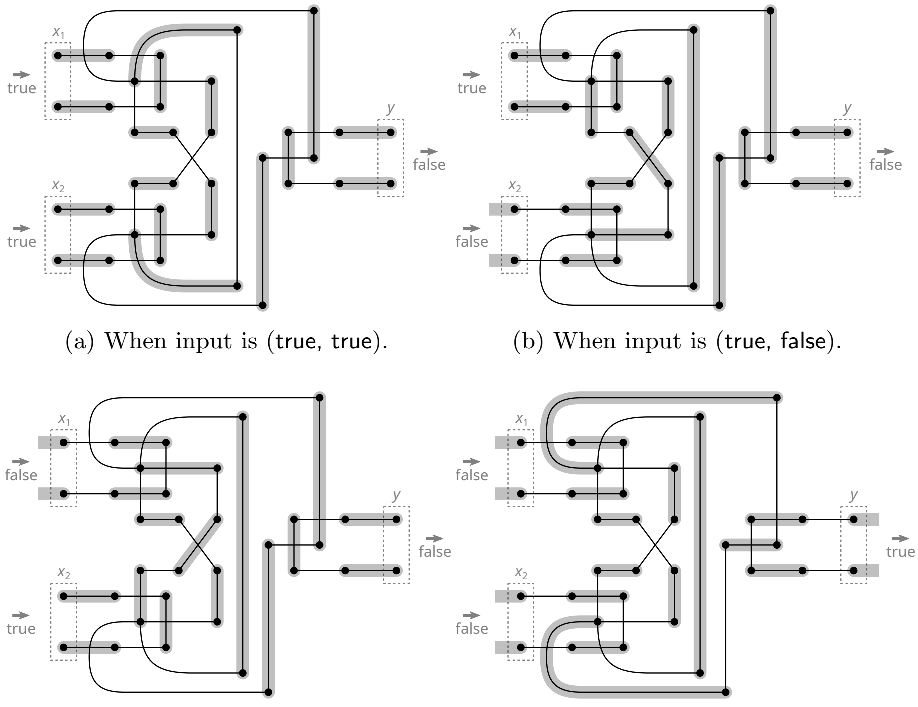
(b) $\ell = 5$.

Fig. 13: The NOR gadgets.

a 2-coloring of a gadget, which is not the output gadget. If ℓ is even, then all vertices in the connection pairs have the same color. If ℓ is odd, then each upper vertex of a connection pair in the figures has the same color, and the other vertices in the connection pairs have the other color. Therefore, the entire graph is 2-colorable. □

Let (G, A, ℓ, ψ) be an instance of Guided Full-ALPP with $G = (V, E)$. For $e = \{v, w\} \in E$, we denote by d_e the length of the subpath of $\psi(e)$ starting at v, passing w, and reaching an endpoint of $\psi(e)$. Let G_e be the graph obtained from G by subdividing e, 2ℓ times. Let P_e be the v-w path of length $2\ell + 1$ in G_e corresponding to e. We set $A_e = A \cup \{x_0, x_1\}$, where x_0 and x_1 are the vertices that have distance d_e and $d_e + \ell$ from v in P_e, respectively. (See Fig. 15.)

For each edge h of G_e, we set $\psi_e(h) = \psi(h)$ if h is not contained in the path $P_{\psi(e)}$ of length 3ℓ that corresponds to $\psi(e)$. If h is contained in $P_{\psi(e)}$, then we set



(a) When input is (true, true). (b) When input is (true, false).

(c) When input is (false, true). (d) When input is (false, false).

Fig. 14: The possible (A, ℓ)-path packings of the NOR gadget.

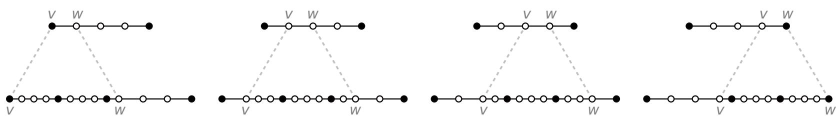


Fig. 15: Subdividing e and introducing two new terminals ($\ell = 4$).

$\psi_e(h)$ to be the unique (A, ℓ)-path in $P_{\psi(e)}$ that contains h. Observe that ψ_e is a guide to (G_e, A_e, ℓ). Furthermore, (G, A, ℓ, ψ) and (G_e, A_e, ℓ, ψ_e) are equivalent (see Fig. 16): if $\psi(e)$ is used in a full (A, ℓ)-path packing of G, then we use two (A, ℓ)-paths in $P_{\psi(e)}$; otherwise we use the middle (A, ℓ)-path in $P_{\psi(e)}$ connecting two new terminals.

Observation D.2. *(G, A, ℓ, ψ) and (G_e, A_e, ℓ, ψ_e) are equivalent instances of Guided Full-ALPP.*

Now we are ready to prove the main theorem of this section.

Theorem D.3 (Theorem 5.1). *For every fixed $\ell \geq 4$, Full-ALPP is NP-complete on grid graphs.*

Fig. 16: Equivalence of (G, A, ℓ, ψ) and (G_e, A_e, ℓ, ψ_e).

Proof. We reduce Guided Full-ALPP on planar bipartite graphs of maximum degree at most 4 for fixed $\ell \geq 4$ (which is NP-complete by Lemma D.1) to Full-ALPP on grid graphs for the same ℓ. Let (G, A, ℓ, ψ) be an instance of Guided Full-ALPP, where $G = (V, E)$ is a planar bipartite graph of maximum degree at most 4.

A *rectilinear embedding* of a graph is a planar embedding into the $\mathbb{Z}^2$ grid such that

- each vertex is mapped to a grid point;
- each edge $\{u, v\}$ is mapped to a rectilinear path between u and v consisting of vertical and horizontal segments connecting grid points;
- the rectilinear paths corresponding to two different edges may intersect only at their endpoints.

Every planar graph of maximum degree at most 4 has a rectilinear embedding, and a rectilinear embedding of area at most $(n+1)^2$ can be computed in linear time [21], where n is the number of vertices.

Let R_1 be a rectilinear embedding of G with area at most $(n+1)^2$. By multiplying each coordinate in the embedding by 2ℓ, we obtain an enlarged rectilinear embedding R_2 of G. Let U be one color class of a 2-coloring of G. Now, for each $v \in U$, we locally modify R_2 around the grid point (x_v, y_v) corresponding to v as illustrated in Fig. 17. We denote by R_3 the locally modified embedding.

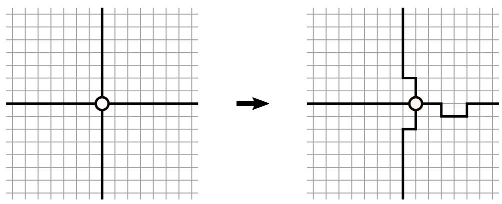


Fig. 17: Local modification around v. The grid point of v is moved to $(x_v + 1, y_v)$.

From R_3, we construct a new graph G' and its rectilinear embedding R' by inserting degree-2 vertices at each intersection point of a grid point and the inner part of a rectilinear path corresponding to an edge. Clearly, G' is a grid graph. Let $e \in E$ and λ_e be the (geometric) length of the rectilinear path in R_1 corresponding to e. Then the rectilinear path in R_3 corresponding to e has length $2\ell \cdot \lambda_e + 1$. Therefore, G' is the graph obtained from G by subdividing each edge e, $2\ell \cdot \lambda_e$ times. By Observation D.2, we can easily compute A' and ψ'

such that (G, A, ℓ, ψ) is equivalent to (G', A', ℓ, ψ'). Finally, from the definition of a guide to an instance of Full-ALPP, (G', A', ℓ, ψ') is equivalent to (G', A', ℓ). As everything in this reduction can be done in time polynomial, the theorem holds. $\square$