Parameterizing Path Partitions

Henning Fernau, Florent Foucaud, Kevin Mann, Utkarsh Padariya, K. N. Rajath Rao

Parameterizing Path Partitions

Henning Fernau[1][0000−0002−4444−3220], Florent Foucaud[2][0000−0001−8198−693X]⋆, Kevin Mann[1][0000−0002−0880−2513], Utkarsh Padariya[3][0000−0003−3422−6940], and Rajath Rao K.N.[3][0000−0003−3422−6940] ⋆⋆

[1] Universität Trier, Fachbereich IV, Informatikwissenschaften, Germany
{fernau,mann}@uni-trier.de
[2] Université Clermont-Auvergne, CNRS, Mines de Saint-Étienne, Clermont-Auvergne-INP, LIMOS, 63000 Clermont-Ferrand, France
florent.foucaud@uca.fr
[3] International Institute of Information Technology Bangalore, India
{utkarsh.prafulchandra,rajath.rao}@iiitb.ac.in

Abstract. We study the algorithmic complexity of partitioning the vertex set of a given (di)graph into a small number of paths. The Path Partition problem (PP) has been studied extensively, as it includes Hamiltonian Path as a special case. The natural variants where the paths are required to be either *induced* (Induced Path Partition, IPP) or *shortest* (Shortest Path Partition, SPP), have received much less attention. Both problems are known to be NP-complete on undirected graphs; we strengthen this by showing that they remain so even on planar bipartite directed acyclic graphs (DAGs), and that SPP remains NP-hard on undirected bipartite graphs. When parameterized by the natural parameter "number of paths", both problems are shown to be W[1]-hard on DAGs. We also show that SPP is in XP both for DAGs and undirected graphs for the same parameter (IPP is known to be NP-hard on undirected graphs, even for two paths). On the positive side, we show that for undirected graphs, both problems are in FPT, parameterized by neighborhood diversity. When considering the dual parameterization (graph order minus number of paths), all three variants, IPP, SPP and PP, are shown to be in FPT for undirected graphs.

Keywords: Path Partitions · NP-hardness · Parameterized Complexity · Neighborhood Diversity · Vertex Cover Parameterization

1 Introduction

Graph partitioning and graph covering problems are among the most studied problems in graph theory and algorithms. There are several types of graph partitioning and covering problems including covering the vertex set by stars (Dominating Set), covering the vertex set by cliques (Clique Covering),

⋆ Research financed by the French government IDEX-ISITE initiative 16-IDEX-0001 (CAP 20-25) and by the ANR project GRALMECO (ANR-21-CE48-0004).
⋆⋆ Research funded by a DAAD-WISE Scholarship 2022.

partitioning the vertex set by independent sets (Coloring), and covering the vertex set by paths or cycles [22]. In recent years, partitioning and covering problems by paths have received considerable attention in the literature because of their connections with well-known graph-theoretic theorems and conjectures like the Gallai-Milgram theorem and Berge's path partition conjecture. Also, these studies are motivated by applications in diverse areas such as code optimization [1], machine learning / AI [32], transportation networks [33], parallel computing [26], and program testing [25]. There are several types of paths that can be considered: unrestricted paths, induced paths, shortest paths, or directed paths (in a directed graph). A path P is an *induced* path in G if the subgraph induced by the vertices of P is a path. An induced path is also called a *chordless* path; *isometric* path and *geodesic* are other names for shortest path. Various questions related to the complexity of these path problems, even in standard graph classes, remained open for a long time (even though they have uses in various fields), a good motivation for this project.

In this paper, we mainly study the problem of partitioning the vertex set of a graph (undirected or directed) into the minimum number of disjoint *paths*, focussing on three problems, Path Partition (PP), Induced Path Partition (IPP) and Shortest Path Partition (SPP) — formal definitions are given in section 2. A *path partition* (pp) of a graph G is a partitioning of the vertex set into unrestricted paths. The *path partition number* of G is the smallest size of a pp of G. Similar definitions apply to ipp and spp. PP is studied extensively under the names of Path Cover and also Hamiltonian Completion on many graph classes [4, 6, 16]. We give a wide range of results in this paper including complexity (NP- or W[1]-hardness) and algorithms (polynomial time or FPT); see Table 1. The types of graphs we consider are general directed, directed acyclic (DAG), general undirected and bipartite undirected graphs. We also consider some structural parameters like neighborhood diversity and vertex cover number.

The three problems considered are all NP-hard. PP can be seen as an extension of Hamiltonian Path and is thus NP-hard, even for one path. IPP is NP-hard, even for two paths [20]. Recently, SPP was proved to be NP-hard [23]. On trees, the three problems are equivalent, and are solvable in polynomial time. For a detailed survey on these types of problems (both partitioning and covering versions), see [22]. The covering version of SPP (where the paths need not necessarily be disjoint) was recently studied, see [8] for an XP algorithm, [2] for NP-hardness on chordal graphs and approximation algorithms for chordal graphs and other classes, and [32] for a $\log n$-factor approximation algorithm.

The versions of these problems where covering is not required (and the endpoints of the solution paths are prescribed in the input) are studied as Disjoint Paths (DP) [27], Disjoint Induced Paths (DIP) and Disjoint Shortest Paths (DSP) [21]. DP in particular has been extensively studied, due to its connections to the Graph Minor theorem [27]. Robertson and Seymour showed that DP is in FPT, parameterized by the number of paths [27], contrasting (D)PP. Recently, DSP was shown to have an XP algorithm and to be W[1]-hard when parameterized by the number of paths [21], solving a 40-year-old open problem.

parameter	PP	SPP	IPP
none (UG)	NP-c. [15]	NP-c. [23]	NP-c. [20]
none (bipartite UG)	NP-c. [18, 20]	**NP-c.**	open
solution size k (UG)	paraNP-h. [15]	**in XP**	paraNP-h. [20]
solution size k (DAG)	polynomial [3, Problem 26-2]	**NP-c. W[1]-h. in XP**	**NP-c. W[1]-h.**
neighborhood diversity (UG)	FPT [14]	**FPT**	**FPT**
dual $n - k$ (UG)	**FPT**	**FPT**	**FPT**

Table 1: Summary of known results concerning path partitioning problems. The abbreviations c. and h. refer to completeness and hardness, respectively. In parentheses, we put further input specifications, with UG referring to undirected graphs. Our results are highlighted in bold face.

Our contribution. Table 1 summarizes known results about the three problems, with our results are highlighted in bold. We fill in most of the hiterto open questions concerning variations of PP, SPP and IPP, e.g., we show that SPP has a poly-time algorithm for a fixed number of paths (in undirected, DAGs, and planar-directed graphs). This is surprising as both PP and IPP are NP-hard for $k = 1$ and for $k = 2$, respectively, see [20]. Many of our results concern our problems restricted to DAGs. Notice that PP has a polynomial time algorithm when restricted to DAGs (using Maximum Matching), but the complexity for IPP and SPP was open for such inputs. We show that IPP and SPP are NP-hard even when restricted to planar DAGs whose underlying graph is bipartite. We strengthen this result using a similar construction as in the classic proof of DP being W[1]-hard on DAGs by Slivkins [29], to show that IPP and SPP are W[1]-hard on DAGs. The complexity of these problems has not been studied when parameterized by structural parameters. We show that IPP and SPP both belong to FPT when parameterized by standard structural parameters like vertex cover and neighborhood diversity using ILP-techniques. Moreover, when considering the dual parameterization (graph order minus number of paths), all three variants, IPP, SPP and PP, are shown to be in FPT for undirected graphs.

It is interesting to note the differences in the complexities of the three problems on different input classes. For example, SPP can be solved in XP-time on undirected graphs, while this is not possible for the other two problems. For DAGs, PP is polynomial-time solvable, but the other two problems are NP-hard. In a way, IPP can be seen as an intermediate problem between PP and SPP. Thus it is not too surprising that, when the complexity of the three problems differs, IPP sometimes behaves like PP, and sometimes, like SPP.

The following combinatorial properties are helpful to connect the problems: (1) Every shortest path is also an induced path. (2) Every induced path of length at most two is also a shortest path. (3) In bipartite graphs, an induced path of length three is a shortest path. (4) If $G = (V, E)$ is a subgraph of H and p is a shortest path in H with only vertices of V, then p is also a shortest path in G. Proofs of statements marked with $(*)$ are omitted due to space constraints; [10].

2 Definitions

We are using standard terminology concerning graphs, classical and parameterized complexity and we will not iterate this standard terminology here. In particular, a *path* P can be described by a sequence of non-repeated vertices such that there is an edge between vertices that are neighbors in this sequence. Sometimes, it is convenient to consider P as a set of vertices. We are next defining the problems considered in this paper. All problems can be considered on undirected or directed graphs, or also on directed acyclic graphs (DAG). We will specify this by prefixing U, D, or DAG to our problem name abbreviations.

Disjoint Paths (DP for short)

Input: A graph G, pairs of terminal vertices $\{(s_1, t_1), \ldots, (s_k, t_k)\}$
Problem: Are there pairwise vertex-disjoint paths $P_1, \ldots, P_k$ such that, for $1 \leq i \leq k$, the end-points of P_i are s_i and t_i?

A path P is an *induced path* in G if the induced graph $G[P]$ is a path; here, P is considered as a vertex set. Analogously to DP, we can define the problem Disjoint Induced Paths or DIP for short. A *shortest path* is a path with end-points u, v that is shortest among all paths from u to v. The greatest length of any shortest path in a graph G is also known as its *diameter*, written as $\mathrm{diam}(G)$. Hence, we can define the problem Disjoint Shortest Paths or DSP for short.

Remark 2.1. $(*)$ *The problems* DIP *and* DP *are* equivalent *when the inputs are undirected graphs or DAGs, but not for general directed graphs.*

Next, we define the corresponding partition problems. The induced and non-induced versions do no longer coincide as seen above for the set of DP problems. We say that a sub-graph G' of $G = (V, E)$ *spans* G if its vertex set is V.

Path Partition (PP for short)

Input: A graph G, a non-negative integer k
Problem: Are there pairwise vertex-disjoint paths $P_1, \ldots, P_{k'}$, with $k' \leq k$, such that, together, these paths span G?

Asking for shortest or induced paths in the partition similarly gives the problems Shortest Path Partition, or SPP, and Induced Path Partition, or IPP, respectively. Contrasting Remark 2.1, the complexities of DAGPP and DAGIPP differ drastically, see Table 1. As also sketched in [12], these problems are tightly linked to Hamiltonian Completion, asking to add at most k (directed) edges to a (di)graph to guarantee the existence of a Hamiltonian path.

Omitting the vertex-disjointness condition, we arrive at *covers* instead:

Path Cover (PC for short)

Input: A graph G, a non-negative integer k
Problem: Are there paths $P_1, \ldots, P_{k'}$, with $k' \leq k$, such that, together, these paths span G?

Similarly, we can define the problems Shortest Path Cover (SPC) and Induced Path Cover (IPC).

The *neighborhood diversity* [19] of a graph G (or just $\mathsf{nd}(G)$) is the number of equivalency classes of the following equivalence: two vertices $u, v \in V$ are equivalent (we also say that they have the *same type*) if they have the same neighborhoods except for possibly themselves, i.e., if $N(v) \setminus \{u\} = N(u) \setminus \{v\}$. The equivalence classes are all cliques, possibly of size one, but one class that collects all isolated vertices.

3 NP-hardness results

A well-known result related to PC in DAGs is Dilworth's theorem: the minimal size of PC equals the maximal cardinality of an anti-chain [7]. Fulkerson [13] gave a constructive proof of this theorem. Hence, the PC problem in DAGs can be reduced to a maximum matching problem in a bipartite graph. Even PP can be solved in polynomial time by reducing it to a matching problem in a bipartite graph [3, Problem 26-2]. We show that DAGSPP, DAGSPC, DAGIPP and DAGIPC are NP-hard even when restricted to planar bipartite DAGs.

Theorem 3.1. DAGSPP *is NP-hard even when the inputs are restricted to planar bipartite DAGs of maximum degree 3.*

Proof (sketch). Our reduction is adapted from [24, 30]. We reduce from the Planar 3-Dimensional Matching problem, or Planar 3-DM, which is NP-complete (see [9]), even when each element occurs in either two or three triples. A 3-DM instance consists of three disjoint sets X, Y, Z of equal cardinality p and a set T of triples from $X \times Y \times Z$. Let $q = |T|$. The question is if there are p triples which contain all elements of X, Y and Z. We associate a bipartite graph with this instance. We assume that the four sets T, X, Y and Z are pairwise disjoint. We also assume that each element of $X \cup Y \cup Z$ belongs to at most three triples. We have a vertex for each element in X, Y, Z and each triple in T. There is an edge connecting triples to elements if and only if the element belongs to the triple. This graph G is bipartite with vertex bipartition of $T, X \cup Y \cup Z$, and has maximum degree 3. We say the instance is planar if G is planar. Given an instance of Planar 3-DM, $G = (T, X \cup Y \cup Z, E)$, and a planar embedding of it, we build an instance $G' = (V', E')$ of DAGSPP.

Construction: We replace each $v_i = (x, y, z) \in T$, where $x \in X$, $y \in Y$, $z \in Z$, with a gadget $H(v_i)$ that consists of 9 vertices named l^i_{jk} where $1 \leq j, k \leq 3$ and with edges as shown in Figure 1; if the planar embedding has x, y, z in clockwise order seen as neighbors of v_i, then we add the arcs (l^i_{12}, x), (l^i_{22}, z) and (l^i_{32}, y), otherwise, we add the arcs (l^i_{12}, x), (l^i_{22}, y) and (l^i_{32}, z).

We observe the following two properties of G'.

Claim 3.2. $(*)$ G' *is a planar DAG with maximum degree 3 in which every shortest/induced path is of length at most* 3*, and the underlying undirected graph of* G' *is bipartite.*

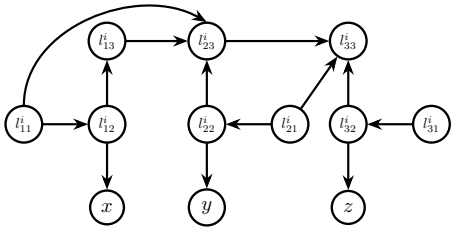

Fig. 1: The vertex gadget as defined in the proof of Theorem 3.1

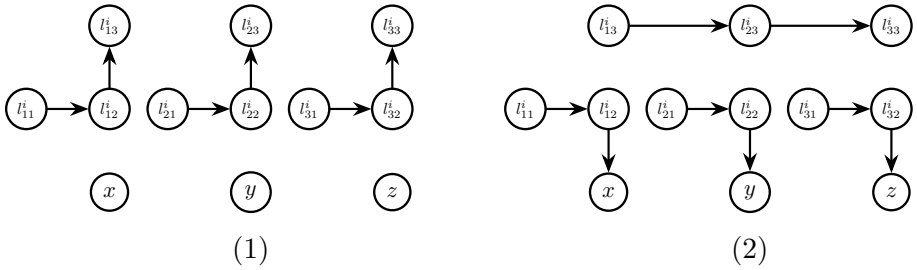

Fig. 2: Two different vertex partitions of a $H(v_i)$ gadget into 3-vertex paths, corresponding to different triple selections in the construction of Theorem 3.1.

Claim 3.3. $(*)$ *The* Planar 3-DM *instance has a solution if and only if G' can be partitioned into $p + 3q$ shortest paths.*

The intuition behind the proof of Claim 3.3 is that each shortest or induced path in a solution must contain exactly three vertices, and each gadget $H(v_i)$ is partitioned into P_3-paths in one of the two ways shown in Figure 2. □

The proof above can also be adapted to DAGSPC, DAGIPP and DAGIPC.

Corollary 3.4. DAGSPC, DAGIPP, *and* DAGIPC *are NP-hard even when restricted to planar bipartite DAGs of maximum degree 3.*

Next, we prove that SPP is NP-hard even when the input graph is restricted to bipartite 5-degenerate graphs with diameter at most 4. To prove this, we reduce from 4-SPP on bipartite graphs to SPP on bipartite graphs. 4-SPP asks, given $G = (V, E)$, if there exists a partition $\mathbb{P}$ of V such that each set in $\mathbb{P}$ induces a shortest path of length 3 in G. First, we show that 4-SPP is NP-hard on bipartite graphs (Lemma 3.6) by a reduction from 4-IPP (also known as Induced P_4-Partition) on bipartite graphs. 4-IPP asks if there exists a partition $\mathbb{P}$ of V such that each set in $\mathbb{P}$ induces a path of length 3 in G.

Lemma 3.5. *[24]* 4-IPP *is NP-hard for bipartite graphs of maximum degree* 3.

Lemma 3.6. $(*)$ 4-SPP *is NP-hard for bipartite graphs of maximum degree* 3.

Theorem 3.7. SPP *is NP-hard, even for bipartite 5-degenerate graphs with diameter 4.*

Proof. To prove this claim, we use Lemma 3.6. Given an instance of 4-SPP, say, $G = (V, E)$, with bipartition $V = A \cup B$ and $|V| = 4k$, as the number of vertices must be divisible by 4, we create an instance $G' = (V', E')$ of SPP.

Construction: We add 10 new vertices to G, getting

$$V' = V \cup \{x_1, x_2, x_3, x_4, x_5, y_1, y_2, y_3, y_4, y_5\}.$$

We add edges from x_2 and x_4 to all vertices of $B \cup \{y_2, y_4\}$. Also, add edges from y_2 and y_4 to all vertices of A, add further edges to form paths $x_1x_2x_3x_4x_5$ and $y_1y_2y_3y_4y_5$. The remaining edges all stem from G. This describes E' of G'.

We have the following observations, due to the construction of G'.

Claim 3.8. $(*)$ *$G' = (V', E')$ is bipartite and 5-degenerate.*

We can make the claimed bipartition explicit by writing $V' = A' \cup B'$, where $A' = A \cup \{x_2, x_4, y_1, y_3, y_5\}$ and $B' = B \cup \{x_1, x_3, x_5, y_2, y_4\}$.

Claim 3.9. $(*)$ *Any shortest path of G', except for $x_1x_2x_3x_4x_5$, $x_1x_2y_2x_4x_5$, $x_1x_2y_4x_4x_5$ and $y_1y_2y_3y_4y_5$, $y_1y_2x_2y_4y_5$, $y_1y_2x_4y_4y_5$, contains at most four vertices. Hence, G' has a diameter at most 4.*

The arguments leading to the previous claim also give raise to the following one.

Claim 3.10. *The only two shortest paths in G' that have five vertices and that can simultaneously exist in a path partition are $x_1x_2x_3x_4x_5$ and $y_1y_2y_3y_4y_5$.* $\Diamond$

Notice that Claim 3.8 and Claim 3.9 together guarantee the additional properties of the constructed graph G' that have been claimed in Theorem 3.7.

Claim 3.11. $(*)$ *Let $u, v \in V$ have distance $d < 3$ in G. Then, they also have distance d in G'. Hence, if p is a shortest path on at most three vertices in G, then p is also a shortest path in G'.*

Now, we claim that G is a Yes-instance of 4-SPP if and only if G' has a shortest path partitioning of cardinality $k' = k + 2$, where $|V| = 4k$. For the forward direction, let D be any solution of 4-SPP for G, containing k shortest paths. To construct a solution D' of SPP for the instance (G', k'), we just need to add the two paths $x_1x_2x_3x_4x_5$ and $y_1y_2y_3y_4y_5$ to D. By Claim 3.11, every path $p \in D$ is in fact a shortest path in G'. Hence, D' is a set of shortest paths with cardinality $k' = k + 2$ that covers all vertices of G'.

For the backward direction, assume G' has a solution D', where $|D'| = k' = k + 2$. As $|V'| = 4k + 10$ by construction, we know by Claim 3.9 that D' contains k paths of length three and two paths of length four. By Claim 3.10, $\{x_1x_2x_3x_4x_5, y_1y_2y_3y_4y_5\} \subseteq D'$ and the rest of the paths of D' are of length three and consists of vertices from V only. Let $D = D' \setminus \{x_1x_2x_3x_4x_5, y_1y_2y_3y_4y_5\}$. As the k paths of D are each of length three also in G (by Table 1) and as they cover V completely, D provides a solution to the 4-SPP instance G. $\square$

The reduction above can also be used for proving the following result. (A graph is *d-degenerate* if every induced subgraph has a vertex of degree at most d.)

Corollary 3.12. SPC *is NP-hard, even for bipartite 5-degenerate graphs with diameter 4.*

4 W[1]-hardness results

The natural or standard parameterization of a parameterized problem stemming from an optimization problem is its solution size. We will study this type of parameterization in this section for path partitioning problems. More technically speaking, we are parameterizing these problems by an upper bound on the number of paths in the partitioning. Unfortunately, our results show that for none of the variations that we consider, we can expect FPT-results.

Theorem 4.1. SPP *(parameterized by solution size) is W[1]-hard on DAGs.*

The following reduction is non-trivially adapted from [28].

Proof. We define a parameterized reduction from Clique to SPP on DAGs (both parameterized by solution size). Let (G, k) be an instance of Clique, where $k \in \mathbb{N}$ and $G = (V, E)$. We construct an equivalent instance (G', k') of the SPP problem, where G' is a DAG and $k' = \frac{k\cdot(k-1)}{2} + 3k$. Let $V = [n]$.

Overview of the construction: We create an array of $k \times n$ identical gadgets for the construction, with each gadget representing a vertex in the original graph. We can visualize this array as having k rows and n columns. If the SPP instance (G', k') is a Yes-instance, then we show that each row has a so-called *selector* in the solution. Here, a selector is a path that traverses all but one gadget in a row, hence skipping exactly one of the gadgets. The vertices in G corresponding to the skipped gadgets form a clique of size k in G. To ensure that all selected vertices form a clique in G, we have so-called *verifiers* in SPP. Verifiers are the paths that are used for each pair of rows to ensure that the vertices corresponding to the selected gadgets in these rows are adjacent in G. This way, we also do not have to check separately that the selected vertices are distinct.

Construction details: The array of gadgets is drawn with row numbers increasing downward and column numbers increasing to the right. Arcs between columns go down and arcs within the same row go to the right. To each row i, with $i \in [k]$, we add a *start terminal*, an arc (s_i, s'_i), and an *end terminal*, an arc (t'_i, t_i). Next, add arcs starting from s_i, s'_i to t_l and t'_l, with $l > i$. Also, for each row, we have $k - i$ *column start terminals*, arcs $(s_{i,j}, s'_{i,j})$ with $i < j$, and $i - 1$ *column end terminals*, arcs $(t_{j,i}, t'_{j,i})$ with $j \in [i - 1]$. Also, add arcs from vertices $s_{i,j}$ and $s'_{i,j}$ to all vertices $t'_{j,p}$ with $p < j$ if $j > i$, and to t_l, t'_l if $l \geq i$.

Gadgets are denoted by $G_{i,u}$, $i \in [k]$, corresponding to $u \in V$. Each gadget $G_{i,u}$ consists of $k - 1$ arcs $(a_r^{i,u}, b_r^{i,u})$, with $r \in [k] \setminus \{i\}$. To each row $i \in [k]$, we also add *dummy gadgets* $G_{i,0}$ and $G_{i,n+1}$. $G_{i,0}$ consists of two arcs, $(a_1^{i,0}, a_2^{i,0})$ and $(b_1^{i,0}, b_2^{i,0})$, also add arcs from $b_1^{i,0}$ and $b_2^{i,0}$ to t_m, t'_m and $t_{m,h}$ where $m < h$ and $i \leq$

h. $G_{i,n+1}$ consists of a directed P_{k+2} and an arc $(b_1^{i,n+1}, b_2^{i,n+1})$; the P_{k+2}-vertices are named $a_j^{i,n+1}$, with $j \in [k+2]$, where $a_{k+2}^{i,n+1}$ has out-degree zero. We can speak of $T^{i,u} := \{a_r^{i,u} \mid r \in [k] \setminus \{i\}\} \cup \{a_1^{i,0}, a_2^{i,0}, a_j^{i,n+1} \mid j \in [k+2]\}$ as being on the *top level*, while the vertices $B^{i,u} := \{b_r^{i,u} \mid r \in [k] \setminus \{i\}\} \cup \{b_1^{i,0}, b_2^{i,0}, b_1^{i,n+1}, b_2^{i,n+1}\}$ form the *bottom level* of gadget $G_{i,u}$.

Due to the natural ordering of the wires within a gadget, we can also speak of the first vertex on the top level of a gadget or that last vertex on the bottom level of a gadget. On both levels, the vertices are connected following their natural wire ordering. More technically speaking, this means that within gadget $G_{i,u}$, with $u \in V$, there is an arc from the first vertex of the upper level to the second vertex of the upper level, from the second vertex of the upper level to the third vertex of the upper level etc., up to an arc from the penultimate vertex of the upper level to the last vertex of the upper level. Moreover, there is an arc from the last vertex of the upper level of $G_{i,u-1}$ to the first vertex of the upper level of $G_{i,u}$ and from the last vertex of the upper level of $G_{i,u}$ to the first vertex of the upper level of $G_{i,u+1}$. Analogously, the vertices of the lower level of the gadgets are connected. These notions are illustrated in Figure 3. In the figure, we omit the superscripts $(a_r, b_r) = (a_r^{i,u}, b_r^{i,u})$, where $r \in [k-i]$.

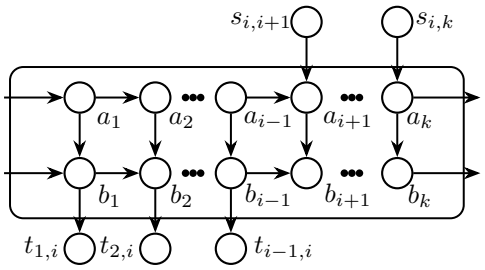

Fig. 3: Gadget $G_{i,u}$, $0 < u \leq n$, $i \in [k]$

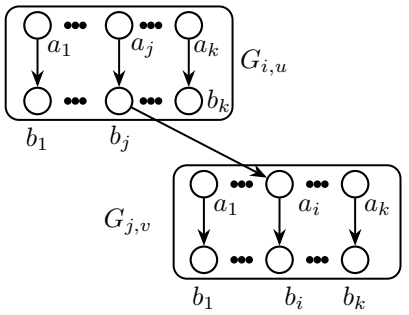

Fig. 4: $G_{i,u}$ connected to $G_{j,v}$, $i < j$

A **selector** is a path that starts at s_i, enters its row at the top level, and exits it at the bottom level, ending at t_i and skipping exactly one gadget $G_{i,u}$, with $i \in [k]$ and $u \in V$. In order to implement this, we add the arcs $(s'_i, a_1^{i,0})$ and $(b_2^{i,n+1}, t'_i)$ for row i, as well as *skipping arcs* that allow to skip a gadget $G_{i,u}$ for $u \in V$. These connect the top level of the last wire of $G_{i,u-1}$ to the bottom level of the first wire of $G_{i,u+1}$.

A **verifier** is a path that routes through one of the wires of the skipped gadget and connects column terminals $s_{i,j}$ to $t_{i,j}$. In order to implement this, we add the arcs $(s'_{i,j}, a_j^{i,u})$ and $(b_i^{i,u}, t'_{i,j})$ to every gadget in rows i and j. To connect the gadget in row i and j, for every edge uv in G, we add an arc between the wires $(b_j^{i,u}, a_i^{j,v})$ for each $i < j$, see Figure 4.

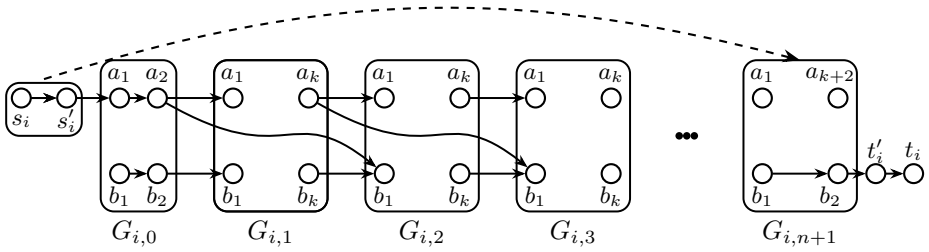

Fig. 5: The i^{th} row (only the first two skipping arcs are shown), the dashed line indicates arcs from s_i, s'_i to $a_j^{i,n+1}$, with $j \in [k+2]$

To force the start and end vertices of paths in the spp of G' of size k', we add the following arcs, called *shortest paths enforcers*. For every row i, from all the vertices of gadget $G_{j,u}$, where $j < i$ and $0 \leq u \leq n$, from every vertex s_l and s'_l of a start terminal and from every vertex $s_{l,m}$, $s'_{l,m}$ of a column start terminal, where $l \leq i$ and $l < m$, add arcs to $a_j^{i,n+1}$, with $j \in [k+2]$.
We are now going to show a number of properties of our construction.
Observe that G' is a DAG because all arcs either go from left to right or from top to bottom. Next, using Claim 4.2 and Claim 4.3, we can deduce that, if G' has an spp of size k', then the start vertex of each path in the solution is fixed.

Claim 4.2. $(*)$ *G' has $k' - k$ vertices with in-degree zero and out-degree zero.*

Hence, we know $k' - k$ start and end vertices of the solution are fixed. The next claim shows that each row i has one more start vertex of some path, hence fixing all the starting vertex of all the paths in the solution.

Claim 4.3. $(*)$ *If a solution $\mathbb{P}$ of the created* SPP *instance G' is of size k' then, for each row $i \in [k]$, there is a path in $\mathbb{P}$ starting from $v \in T^{i,u}$ which covers at least the vertex $a_1^{i,n+1}$.*

Hence, if G' has an spp of size k', then the paths in the solution must start at: for $i \in [k]$, s_i, $b_1^{i,0}$, $v, v \in T^{i,u}$ and $s_{i,j}$, with $i < j$. Next, we will show observations about these paths' ending vertices.

Claim 4.4. $(*)$ *If G' has an spp of size k', then a path (in this* SPP *solution) starting at s_i has to end at t_i, with $i \in [k]$.*

With arguments along similar lines, we can show:

Claim 4.5. *If G' has an spp of size k', then any path in this partition starting at $s_{i,j}$ has to end at $t_{i,j}, i < j$.*

Claim 4.6. *If G' has an spp of size k', then any path in this partition starting at $v \in T^{i,u}$ has to end at $a_i^{i,n+1}, i \in [k]$.*

Also, if G' has an spp of size k', then for each row i, with $i \in [k]$, any path starting at s_i has to skip exactly one gadget and end at t_i. If we do not skip any gadget, $s_{i,j}$ cannot be connected to $t_{i,j}, j > i$. Once we skip a gadget, we are at the bottom level of the row and hence we cannot skip again. To conclude if G has a shortest path partition of size k' (then in the solution) the path starting at s_i has to end at t_i and this path has to skip exactly one gadget. Through this skipped gadget, $s_{i,j}$ is connected to $t_{i,j}$. The bottom of the row is covered by a path starting from $b^{i,0}$ and the top of the row is covered by a path starting at $v \in T^{i,u}$ after the skipped gadget and ending at $a_{k+2}^{i,n+1}$.

Finally, we have the following claim about G' that follows by construction.

Claim 4.7. *There exists a path between $s_{i,j}$ and $t_{i,j}$ through two gadgets $G_{i,u}$ and $G_{j,v}$ where $i > j$ if and only if there is an edge uv in the graph G.*

Claim 4.8. $(*)$ *G has a k-clique iff G' has a shortest path partition of size k'.*

As the construction is polynomial-time, W[1]-hardness follows. □

5 XP Algorithms

We now present our XP algorithms to prove the following result. We note that the result for USPP also follows from [8] (with a different proof).

Proposition 5.1. USPP *and* DAGSPP *are in XP.*

For our XP algorithms, the following combinatorial result is crucial.

Lemma 5.2. $(*)$ *Let $G = (V, E)$ be a directed graph. Then, V can be partitioned into k vertex-disjoint shortest paths if and only if there are k vertex-disjoint paths between some s_i and t_i, for $1 \leq i \leq k$, such that $\sum_{i=1}^{k} d(s_i, t_i) = |V| - k$. A similar characterization is true for the undirected case.*

This lemma allows us to prove Proposition 5.1 by cycling through all possible $X := \{(s_1, t_1), \ldots, (s_k, t_k)\}$, resulting in an instance of DP. Now, either apply [11, Theorem 3] for DAGs or [17,27] for undirected graphs to get the XP-result. Notice that these algorithmic results rule out paraNP-hardness results.

6 Neighborhood Diversity Parameterization

One of the standard structural parameters studied within parameterized complexity is the *vertex cover number*. As graphs with bounded vertex cover number are highly restricted, less restrictive parameters that generalize vertex cover are interesting, as *neighborhood diversity* is, introduced by Lampis [19].

Crucial to our FPT-results is the following interesting combinatorial fact.

Proposition 6.1. $(*)$ *If G is connected, then* $\mathrm{diam}(G) \leq \mathsf{nd}(G)$.

Similar arguments show that the number of neighborhood diversity equivalence classes that a shortest path P intersects equals its number of vertices if P contains at least four vertices. In shortest paths on two or three vertices, however, their endpoints might have the same type. Now, call two shortest paths P_i and P_j (viewed as sets of vertices) *equivalent*, denoted by $P_i \equiv P_j$, in a graph G with $d = \mathsf{nd}(G)$ if $|P_i \cap C_l| = |P_j \cap C_l|$ for all l with $1 \leq l \leq d$, where $C_1, \ldots, C_d$ denote the nd-equivalence classes. Any two equivalent shortest paths have the same length. Proposition 6.1 and our discussions imply that there are $\mathcal{O}(2^{\mathsf{nd}(G)})$ many equivalence classes of shortest paths (+).

Theorem 6.2. USPP *is FPT when parameterized by neighborhood diversity.*

Proof. As we can solve SPP separately on each connected component, we can assume in the following that the input graph is connected.

Given that the neighborhood diversity of a graph $G = (V, E)$ is bounded by an integer d, then there exists a partition of V into d nd-classes $C_1, \ldots, C_d$. Hence, each C_i for $i \in [d]$ either induces a clique or an independent set. Such a partition can be found in linear time with a fast modular decomposition algorithm [31].

Compute the set of all shortest path equivalence classes; this can be represented by a set $\mathcal{P}$ of shortest paths, in which any two shortest paths are not equivalent. By (+), we know that $|\mathcal{P}| \in \mathcal{O}(2^{\mathsf{nd}(G)})$. Construct and solve the following Integer Linear Program (ILP), with variables $z_p \geq 0$ corresponding to $p \in \mathcal{P}$. Each $p \in \mathcal{P}$ is characterized by a vector $(p^1, \ldots, p^d)$ with $p^j = |C_j \cap p|$.

$$\text{minimize} \sum_{p \in \mathcal{P}} z_p$$

$$\text{subject to} \sum_{p \in \mathcal{P}} z_p \cdot p^j = |C_j| \text{ for all } j \in [d]$$

The variable z_p encodes how many shortest paths equivalent to p are taken in the solution. Hence, the objective function expresses minimizing the number of shortest paths used in the partition. The constraints ensure the path partitioning.

Claim 6.3. (∗) *There exists a spp of G with k shortest paths if and only if the objective function attains the value k in the ILP described above.*

Notice that the number of variables of the ILP is exactly $|\mathcal{P}|$, which is $\mathcal{O}(2^d)$, as observed above. Now, we apply [5, Theorem 6.5] to prove our FPT claim. □

The arguments leading to Proposition 6.1 and the subsequent discussions are all proofs by contradiction that show shortcuts in shortest paths that are 'too long'. This also works for induced paths instead of shortest paths, leading to:

Proposition 6.4. *The length of a longest induced path in a graph G is upper-bounded by the neighborhood diversity $\mathsf{nd}(G)$.*

Theorem 6.5. UIPP *is FPT when parameterized by neighborhood diversity.*

By using results of Lampis [19], we can immediately infer the following.

Corollary 6.6. USPP, UIPP *are* FPT*, parameterized by vertex cover number.*

We have a similar result for UPP, either by a more general approach in [14], or based on a direct reasoning. It is open if one can get a polynomial-size kernel.

Proposition 6.7. $(*)$ UPP *is* FPT *when parameterized by vertex cover number, as it possesses a single-exponential size kernel.*

We can also obtain a direct FPT algorithm with running time $\mathcal{O}^*\left(\mathsf{vc}(G)! \cdot 2^{\mathsf{vc}(G)}\right)$.

7 Duals and Distance to Triviality

Given a typical graph problem that is (as a standard) parameterized by solution size k, it takes as input a graph G of order n and k, then its *dual parameter* is $k_d = n - k$. This applies in particular to our problems UPP, UIPP and USPP. As these problems always have as a trivial solution the number n of vertices (i.e., n trivial paths), we can also interpret this dual parameterization as a parameterization led by the idea of *distance from triviality*. Moreover, all our problems can be algorithmically solved for each connected component separately, so that we can assume, w.l.o.g., that we are dealing with connected graphs. Namely, if (G, k) with $G = (V, E)$ is a graph and $C \subsetneq V$ describes a connected component, then we can solve $(G[C], k')$ and $(G - C, k - k')$ independently for all $1 \leq k' < k$. We now prove that our problems, with dual parameterizations, are in FPT by providing a kernelization algorithm. The following claims are crucial.

Lemma 7.1. $(*)$ *Let G be a graph of order n. If G has a matching that covers $2k$ vertices, then $(G, n - k)$ is a* Yes*-instance of* UPP, UIPP *and* USPP.

The preceding lemma has the following interesting consequence.

Corollary 7.2. $(*)$ *If $G = (V, E)$ is a graph that possesses some $X \subseteq V$ with $|X| \geq 2k$ such that $\deg(v) \geq 2k$ for every $v \in X$, then G has a matching of size k and hence $(G, n - k)$ is a* Yes*-instance of* UPP, UIPP *and* USPP.

This consequence, as well as the following combinatorial observation, has no direct bearing on our algorithmic result, but may be of independent interest.

Lemma 7.3. $(*)$ *If G is a connected graph with $\mathrm{diam}(G) > k$, then $(G, n - k)$ is a* Yes*-Instance of* UPP, UIPP *and* USPP.

Our combinatorial thoughts, along with Corollary 6.6 and Proposition 6.7, allow us to show the following algorithmic result, the main result of this section.

Theorem 7.4. $(*)$ UPP, UIPP *and* USPP *can be solved in* FPT *time with dual parameterization.*

8 Conclusion

We have explored the algorithmic complexity of the three problems PP, IPP and SPP, and as witnessed by Table 1, our results show some interesting algorithmic differences between these three problems.

Many interesting questions remain to be investigated. For example, what is the parameterized complexity of SPP on undirected graphs, parameterized by the number of paths? Is it W[1]-hard, like for DAGs? This was asked in [8], and our W[1]-hardness result for DAGs can be seen as a first step towards an answer.

We have seen that PP, IPP and SPP admit FPT algorithms when parameterized by neighborhood diversity. Can we obtain such algorithms for other (e.g., more general) parameters (as was done for PP and modular-width in [14])?

Moreover, in the light of our FPT algorithms for the dual parameterizations of PP, IPP and SPP, we can ask whether they admit a polynomial kernel.

References

1. Boesch, F.T., Gimpel, J.F.: Covering points of a digraph with point-disjoint paths and its application to code optimization. Journal of the ACM **24**(2), 192–198 (1977)
2. Chakraborty, D., Dailly, A., Das, S., Foucaud, F., Gahlawat, H., Ghosh, S.K.: Complexity and algorithms for isometric path cover on chordal graphs and beyond. In: Proceedings of the 33rd International Symposium on Algorithms and Computation, ISAAC 2022. LIPIcs, vol. 248, pp. 12:1–12:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022)
3. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, Third Edition. The MIT Press, 3rd edn. (2009)
4. Corneil, D.G., Dalton, B., Habib, M.: LDFS-based certifying algorithm for the minimum path cover problem on cocomparability graphs. SIAM Journal on Computing **42**(3), 792–807 (2013)
5. Cygan, M., Fomin, F., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms. Springer (2015)
6. Damaschke, P., Deogun, J.S., Kratsch, D., Steiner, G.: Finding Hamiltonian paths in cocomparability graphs using the bump number algorithm. Order **8**(4), 383–391 (1992)
7. Dilworth, R.P.: A decomposition theorem for partially ordered sets. In: Classic Papers in Combinatorics, pp. 139–144. Springer (2009)
8. Dumas, M., Foucaud, F., Perez, A., Todinca, I.: On graphs coverable by k shortest paths. In: Proceedings of the 33rd International Symposium on Algorithms and Computation, ISAAC 2022. LIPIcs, vol. 248, pp. 40:1–40:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022)
9. Dyer, M.E., Frieze, A.M.: Planar 3DM is NP-complete. Journal of Algorithms **7**(2), 174–184 (1986)
10. Fernau, H., Foucaud, F., Mann, K., Padariya, U., Rao, K.N.R.: Parameterizing path partitions. CoRR ArXiv preprint **arXiv:2212.11653** (2022)
11. Fortune, S., Hopcroft, J.E., Wyllie, J.: The directed subgraph homeomorphism problem. Theoretical Computer Science **10**, 111–121 (1980)
12. Franzblau, D.S., Raychaudhuri, A.: Optimal Hamiltonian completions and path covers for trees, and a reduction to maximum flow. ANZIAM Journal **44**, 193–204 (2002)

13. Fulkerson, D.R.: Note on Dilworth's decomposition theorem for partially ordered sets. Proceedings of the AMS **7**(4), 701–702 (1956)
14. Gajarský, J., Lampis, M., Ordyniak, S.: Parameterized algorithms for modular-width. In: Gutin, G.Z., Szeider, S. (eds.) Parameterized and Exact Computation - 8th Intern. Symposium, IPEC. LNCS, vol. 8246, pp. 163–176. Springer (2013)
15. Garey, M.R., Johnson, D.S.: Computers and Intractability. Freeman (1979)
16. Goodman, S., Hedetniemi, S.: On the Hamiltonian completion problem. In: Graphs and Combinatorics, pp. 262–272. Springer (1974)
17. Kawarabayashi, K.i., Kobayashi, Y., Reed, B.: The disjoint paths problem in quadratic time. Journal of Combinatorial Theory, Series B **102**(2), 424–435 (2012)
18. Krishnamoorthy, M.S.: An NP-hard problem in bipartite graphs. ACM SIGACT News **7**(1), 26–26 (1975)
19. Lampis, M.: Algorithmic meta-theorems for restrictions of treewidth. Algorithmica **64**(1), 19–37 (2012)
20. Le, H.O., Le, V.B., Müller, H.: Splitting a graph into disjoint induced paths or cycles. Discrete Applied Mathematics **131**(1), 199–212 (2003)
21. Lochet, W.: A polynomial time algorithm for the k-disjoint shortest paths problem. In: Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 169–178. SIAM (2021)
22. Manuel, P.D.: Revisiting path-type covering and partitioning problems. CoRR ArXiv preprint **arXiv:1807.10613** (2018)
23. Manuel, P.D.: On the isometric path partition problem. Discussiones Mathematicae Graph Theory **41**(4), 1077–1089 (2021)
24. Monnot, J., Toulouse, S.: The path partition problem and related problems in bipartite graphs. Operations Research Letters **35**(5), 677–684 (2007)
25. Ntafos, S., Hakimi, S.: On path cover problems in digraphs and applications to program testing. IEEE Transactions on Software Engineering **SE-5**(5), 520–529 (1979)
26. Pinter, S.S., Wolfstahl, Y.: On mapping processes to processors in distributed systems. International Journal of Parallel Programming **16**(1), 1–15 (1987)
27. Robertson, N., Seymour, P.: Graph minors XIII. The disjoint paths problem. Journal of Combinatorial Theory, Series B **63**(1), 65–110 (1995)
28. Schrijver, A.: Finding k disjoint paths in a directed planar graph. SIAM Journal on Computing **23**(4), 780–788 (1994)
29. Slivkins, A.: Parameterized tractability of edge-disjoint paths on directed acyclic graphs. SIAM Journal of Discrete Mathematics **24**(1), 146–157 (2010)
30. Steiner, G.: On the k-path partition of graphs. Theoretical Computer Science **290**(3), 2147–2155 (2003)
31. Tedder, M., Corneil, D., Habib, M., Paul, C.: Simpler linear-time modular decomposition via recursive factorizing permutations. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) International Colloquium on Automata, Languages, and Programming, ICALP, Part I: Tack A: Algorithms, Automata, Complexity, and Games. LNCS, vol. 5125, pp. 634–645. Springer (2008)
32. Thiessen, M., Gaertner, T.: Active learning of convex halfspaces on graphs. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Proceedings of the 35th Conference on Neural Information Processing Systems, NeurIPS 2021. vol. 34, pp. 23413–23425. Curran Associates, Inc. (2021)
33. Wang, C., Song, Y., Fan, G., Jin, H., Su, L., Zhang, F., Wang, X.: Optimizing cross-line dispatching for minimum electric bus fleet. IEEE Transactions on Mobile Computing (to appear)